cadence®

PSpice A/D User Guide

Product Version 23.1
September 2023

© 2023 Cadence Design Systems, Inc. All rights reserved.
Portions © Apache Software Foundation, Sun Microsystems, Free Software Foundation, Inc., Regents of the University of California, Massachusetts Institute of Technology, University of Florida. Used by permission. Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Product PSpice contains technology licensed from, and copyrighted by: Apache Software Foundation, 1901 Munsey Drive Forest Hill, MD 21050, USA © 2000-2005, Apache Software Foundation. Sun Microsystems, 4150 Network Circle, Santa Clara, CA 95054 USA © 1994-2007, Sun Microsystems, Inc. Free Software Foundation, 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA © 1989, 1991, Free Software Foundation, Inc. Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, © 2001, Regents of the University of California. Daniel Stenberg, © 1996 - 2006, Daniel Stenberg. UMFPACK © 2005, Timothy A. Davis, University of Florida, (davis@cise.ulf.edu). Ken Martin, Will Schroeder, Bill Lorensen © 1993-2002, Ken Martin, Will Schroeder, Bill Lorensen. Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts, USA © 2003, the Board of Trustees of Massachusetts Institute of Technology. All rights reserved.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522.

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information. Cadence is committed to using respectful language in our code and communications. We are also active in the removal and/or replacement of inappropriate language from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related software can be updated without end-user impact.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

1
Introducing PSpice A/D 15
Analyses to Run with PSpice A/D 17
Files Required for Simulation 21
Files that Design Entry Tool Generates 21
Other Input Files to Configure 22
Preparing and Configuring Input Files 23

2
Preparing a Design for Simulation 25
PSpice Libraries 26
Standard PSpice Libraries 26
PSpice Advanced Analysis Libraries 30
Creating a PSpice-Enabled Project 32
Defining Stimuli 33
European Notation for Component Values 36
Checklist for Simulation Setup 37
Troubleshooting a Design 38

3
Creating and Editing Models 39
Models Defined as Model Parameter Sets 39
Organization of Models 40
Model libraries 40
Model Library Configuration 41
Global vs. Design vs. Profile Models and Libraries 41
Nested Model Libraries 42
PSpice-Provided Models 42
Model Library Data 42
Device characteristic curves-based models vs. Template-based models 44

@ 2023 All Rights Reserved.

Tools to Create and Edit Models . . . 46
Ways to Create and Edit Models . . . 47
Using the Model Editor . . . 50
Ways to use the Model Editor . . . 51
Running the Model Editor Alone . . . 51
Starting the Model Editor . . . 52
Creating Models Using the Model Editor . . . 52
Creating models based on device characteristic curves . . . 52
Creating models based on PSpice templates . . . 58
Importing an existing model . . . 60
Enabling and disabling automatic part creation . . . 61
Running the Model Editor from the schematic editor . . . 63
Model Creation Examples . . . 66
Example: Creating a PSpice model based on device characteristic curves . . . 66
Example: Creating template-based PSpice model . . . 75
Editing model text . . . 80
Example: Editing a Q2N2222 Instance Model . . . 82
Using the Create Subcircuit Format Netlist Command (Capture only) . . . 83
Changing the Model Reference to an Existing Model Definition . . . 85
Reusing instance models . . . 86
Reusing instance models in the same schematic . . . 86
Making instance models available to all designs . . . 87
Configuring Model Libraries . . . 88
The Configuration Files Tab . . . 88
How PSpice A/D uses model libraries . . . 90
Adding model libraries to the configuration . . . 92
Changing the Model Library Scope . . . 93
Changing model library search order . . . 94
Order of Access and Precedence for Models used in Simulation . . . 95
Changing the library search path . . . 98
Handling Smoke Information Using the Model Editor . . . 99
Adding smoke information to PSpice models . . . 99
Creating template-based PSpice models with smoke information . . . 100
Using the Model Editor to edit smoke information . . . 101
Examples: Smoke . . . 102
Adding smoke information to the D1 diode model . . . 102

@ 2023 All Rights Reserved.

Adding smoke information to the OPA_LOCAL operational amplifier model 104
Smoke Parameters 105
Diode 106
Bipolar Junction Transistors 107
Magnetic Core 108
Ins Gate Bipolar Transistor (IGBT) 109
Junction FET 111
Operational Amplifier 113
MOSFET 115
Voltage Regulator 117
Darlington Transistor 118

4
Creating Parts for Models 121

Ways to Create Parts for Models 121
Preparing Models for Part Creation 123
Starting the Model Editor 123
Using the Model Editor to Create Parts 125
Batch mode of part creation 125
Interactive Mode of Part Creation 125
Creating Design Entry Tool parts for all Models in a Library 126
Using Batch Mode 126
Using Interactive Mode 127
Setting up Automatic Part Creation 131
Part Creation Examples 132
Creating Parts in Batch Mode 133
Creating parts using interactive mode 138
Basing New Parts on a Custom Set of Parts 142
Editing Part Graphics (Capture only) 144
How Capture Places Parts 145
Defining Grid Spacing 146
Attaching Models to Parts 147
Defining Part Properties Needed for Simulation 149

@ 2023 All Rights Reserved.

5
Analog Behavioral Modeling 161
The ABM Part Library File 162
Placing and Specifying ABM Parts 163
Net names and device names in ABM expressions 163
Forcing the use of a global definition 164
ABM Part Templates 165
Control System Parts 166
Basic Components 169
Limiters 170
Chebyshev Filters 171
Integrator and differentiator 175
Table look-up parts 175
Laplace transform part 180
Math functions 183
ABM expression parts 184
An instantaneous device example: modeling a triode 188
PSpice-equivalent parts 191
Implementation of PSpice-equivalent parts 192
Modeling mathematical or instantaneous relationships 193
Lookup tables (ETABLE and GTABLE) 196
Frequency-domain device models 198
Laplace transforms (LAPLACE) 198
Frequency response tables 200
Cautions and Recommendations for Simulation and Analysis 203
Instantaneous Device Modeling 203
Frequency-domain Parts 204
Laplace Transforms 204
Trading Off Computer Resources for Accuracy 207
Basic Controlled Sources 208
Creating Custom ABM Parts 208

@ 2023 All Rights Reserved.

6
Digital Device Modeling . . . 209
Functional Behavior . . . 210
Timing Characteristics . . . 218
Timing Model . . . 218
Propagation Delay Calculation . . . 220
Inertial and Transport Delay . . . 221
Input/Output Characteristics . . . 223
Input/Output Model . . . 223
I/O Models for Digital Modeling Applications . . . 226
Defining Output Strengths . . . 229
Charge storage nets . . . 231
Creating your own interface subcircuits for additional technologies . . . 232
Creating a Digital Model using the PINDLY and LOGICEXP Primitives . . . 236
Digital primitives . . . 236
Logic Expression (LOGICEXP primitive) . . . 237
Pin-to-pin delay (PINDLY primitive) . . . 238
BOOLEAN . . . 239
PINDLY . . . 240
Constraint Checker (CONSTRAINT Primitive) . . . 241

7
Setting Up Analyses and Starting Simulation . . . 245
Output Variables . . . 247
Creating a New Simulation Profile . . . 254
Modifying an Existing Simulation Profile . . . 254
Using a Profile for Simulation . . . 255
Deleting a Simulation Profile . . . 256
General Simulation Settings . . . 256
Analysis Settings . . . 257
Configuration Files . . . 258
Options for Simulation Profiles . . . 262
Setting Up Auto Convergence . . . 264
Advanced Analog Options . . . 268

@ 2023 All Rights Reserved.

Setting up Solver Algorithms 274
Data Collection Options 282
Setting Data Collection Options 282
Probe Windows Options 284
Probe Window Display Options 284
Setting Up Probe Window Options 284
Starting a Simulation 285
Batch Simulations 286
Multiple Simulation Setups within One Circuit File 287
Running Simulations with Multiple Circuit Files 287
Viewing the Simulation Queue 287
PSpice A/D Simulation Window 288
Interaction with Simulation 291
Extending a Transient Analysis 291
Interrupting a Simulation 294
Schedule Changes to Runtime Parameters 295
Scheduling Runtime Parameters Changes 296
Using the Simulation Manager 297
Error Conditions During Simulation 301
Setting up Multiple Simulations 301
Starting, Stopping, and Pausing Simulations 302
Attaching PSpice A/D to a Simulation 302
Setting Options in the Simulation Manager 303

8
DC Analyses 307

DC Sweep 309
Setting up Simulation Profile to Run DC Sweep 311
Bias Point 317
Setting up Simulation Profile to Calculate Bias Point 319
Small-Signal DC Transfer 320
Setting up a Small-Signal DC Transfer Analysis 320
DC Sensitivity 322
Setting up a DC Sensitivity Analysis 322

@ 2023 All Rights Reserved.

9
AC Analysis 325
Setting up an AC Analysis 325
AC Sweep Analysis 328
Setting Up AC Sweep Analysis 329
Setting up an AC Stimulus 329
Noise Analysis with PSpice 334
Setting up and Running a Noise Analysis 335
Analyzing Noise in Probe Window 337

10
Time Domain Analysis 341
Setting Up a Transient Analysis 341
Check Point Options 342
Defining Checkpoints 343
Stimulus Editor Utility 347
Running the Stimulus Editor 349
Editing a Stimulus 355
Transient (time) Response 358
Internal Time Steps 360
Switching Circuits 360
Hysteresis Curves 361
Fourier Analysis 364

11
Parametric Analysis 365
Performance Analysis 366
Frequency Response vs. Arbitrary Parameter 371

12
Temperature Analysis 375
Setting up a Temperature Analysis 376

@ 2023 All Rights Reserved.

13 Statistical Analyses . . . 379

Monte Carlo History Support . . . 380
Temperature Considerations in Statistical Analyses . . . 388
Monte Carlo Analysis . . . 389
- Setting up a Simulation for Monte-Carlo Analysis . . . 389
- Monte Carlo Analysis of a Pressure Sensor . . . 395
- Monte Carlo Histograms . . . 403

Sensitivity Analysis . . . 410
- Worst-case Analysis Example . . . 413
- Tips and Other Useful Information . . . 418
- Gaussian Distributions . . . 419
- Sensitivity Analysis Results . . . 420

14 Digital Simulation . . . 423

Simulating Digital Circuits . . . 423
Concepts of Digital Simulation . . . 424
- States . . . 424
- Strengths . . . 425

Defining a Digital Stimulus . . . 426
- Using the DIGSTIMn part . . . 427
- Defining Input Signals using Stimulus Editor . . . 427
- Using the DIGCLOCK Part . . . 434
- Using STIM1, STIM4, STIM8 and STIM16 parts . . . 435
- Using the FILESTIMn parts . . . 436

Defining Simulation Time . . . 440
Adjusting Simulation Parameters . . . 441
- Selecting Propagation Delays . . . 442
- Initializing Flip-flops . . . 443

Starting a Simulation . . . 443
Analyzing Results . . . 444
- Adding digital signals to a plot . . . 445
- Adding buses to a waveform plot . . . 447

@ 2023 All Rights Reserved.

Tracking timing violations and hazards 450

15
Mixed Analog/Digital Simulation 457

Interface Subcircuit Selection by PSpice A/D 458
Level 1 interface 459
Level 2 interface 459
Setting a Default A/D Interface 461
Specifying Digital Power Supplies 462
Default Power Supply Selection by PSpice A/D 462
Creating Custom Digital Power Supplies 464
Interface Generation and Node Names 467

16
Digital Worst-case Timing Analysis 471

Digital Worst-case Analysis compared to Analog Worst-case Analysis 471
Starting Digital Worst-case Timing Analysis 473
Simulator Representation of Timing Ambiguity 473
Propagation of Timing Ambiguity 475
Identification of Timing Hazards 475
Convergence Hazard 476
Critical Hazard 477
Cumulative Ambiguity Hazard 478
Reconvergence Hazard 480
Glitch Suppression due to Inertial Delay 482
Timing Simulation Methodology 483

17
Analyzing Waveforms with PSpice 487

Elements of a Plot 488
Elements of a Probe Window 488
Managing Multiple Probe Windows 489
Setting up Waveform Analysis 492
Viewing Waveforms 497

@ 2023 All Rights Reserved.

Setting up a Waveform Display 497
Viewing Waveforms while Simulating 498
Using schematic page markers to add traces 501
Using Display Control 504
Using plot window templates 507
Limiting Waveform Data File Size 519
Viewing large data files 522
Using simulation data from multiple files 530
Comparing measured data with simulated data 534
Saving Simulation Results in ASCII Format 535
User Interface Features For Waveform Analysis 537
Tracking Digital Simulation Messages 550
Trace Expressions 553
Basic Output Variable Form 553
Analog Trace Expressions 561
Digital Trace Expressions 563

18
Measurement Expressions 567

Creating Measurement Expressions 567
Composing a Measurement Expression 568
Viewing Results of Measurement Evaluations 569
Example of Measurement Expression 569
Measurement Definitions Included in PSpice A/D 574
Creating Custom Measurement 578

19
Other Output Options 593

Viewing Analog Results in PSpice A/D Window 594
Writing Additional Results to PSpice Output File 595
Generating Plots of Voltage and Current Values 595
Generating Tables of Voltage and Current Values 596
Generating Tables of Digital State Changes 597
Creating Test Vector Files 599
Exporting Trace to CSV File 601

@ 2023 All Rights Reserved.

20
Bias Point Display 603

Bias Point Data for Multiple Analyses 604
Displaying Bias Point Values 605
Controlling the Display of Bias Points 607
Moving Bias Points 609
Updating Bias Point Values 609
Menu Commands for Bias Point Display 610
Toolbar Controls for Bias Point Display 611
Toggling a Specific Current Bias Display 612
Toggling a Specific Voltage Bias Display 612
Toggling a Specific Power Bias Display 613

A
Setting Initial State 615

Save and Load Bias Points 616
  Save Bias Point 616
  Load Bias Point 617
Setpoints 618
Setting Initial Conditions 620

B
Convergence and “time step too small errors” 621

Introduction 622
  Newton-Raphson Requirements 622
Diagnostics 625
Bias Point (DC) Convergence 626
DC Sweep Convergence 630
Transient Convergence 631

C
Importing Spice Models 637

Tasks for Vendor-Supplied Models 638

@ 2023 All Rights Reserved.

Importing Text Models 638
Generating Part Symbols 639
Creating New Symbols 639
Using symbols from an existing symbol library 640
Using Model Import wizard 643
Configuring New Model Library 644
Editing Model Editor Created Symbols 646

@ 2023 All Rights Reserved.

1

Introducing PSpice A/D

PSpice A/D is a simulation program that models the behavior of a circuit with both analog and digital devices. It simulates mixed-signal circuits with no performance degradation as both analog and digital simulation algorithms are built into the same program. You can use PSpice A/D to test and refine your design before you move to layout and fabrication.

PSpice analyses are used to test circuit response under different conditions as follows:

- DC, AC, and transient analyses are used to test the response of a circuit to different inputs.
- Parametric, Monte Carlo, and Sensitivity or worst-case analyses are used to test the variation in circuit behavior varies with changing component values.
- Digital worst-case timing analysis is used to detect timing problems that occur with only certain combinations of slow and fast signal transmissions.

The following table lists the models supported in PSpice A/D:

Cadence-supplied Models	Description
`Standard parts`	PSpice supports the range of models created using components such as resistors, inductors, capacitors.
`Bipolar transistors`	It includes general purpose, duel, NPN, PNP, RF transistors.
`Diode`	It includes current regulator, zener, PIN and more.
`Transmission line models`	It includes delay, reflection, loss, dispersion, and crosstalk.
`Nonlinear magnetic core models`	It includes saturation and hysteresis.
`MOSFET models`	It includes BSIM4 version 4.1, BSIM3 version 3.2, and EKV version 2.6.
`JFET`	It includes MOSFET, N-channel and thyristors.

@ 2023 All Rights Reserved.

`GaAsFET models`	It includes Parker-Skellern and TriQuint's TOM2 model.
`IGBTs`	The IGBT model parameters can be extracted from data sheet information by using the model editor.

Related Topics

- Creating and Editing Models
- Creating Parts for Models
- Analog Behavioral Modeling
- Digital Device Modeling

@ 2023 All Rights Reserved.

Analyses to Run with PSpice A/D

PSpice A/D supports both basic and advanced analyses. The following table explains what all analyses you can run with PSpice A/D and what it calculates for each analysis type:

Analysis Type	Analysis Option	Description
Basic Analysis		
AC Analysis		To evaluate circuit performance in response to a small-signal alternating current source, you can perform AC analysis.
	AC Sweep	Calculates small-signal response of the circuit (linearized around the bias point) when sweeping one or more sources over a range of frequencies. **Note:** Bias points are the node voltages, currents and power that PSpice calculates through the devices in a circuit. Outputs include voltages and currents with magnitude and phase.
	Noise	For each frequency specified in the AC analysis, PSpice A/D calculates and reports the following: ■ Propagated noise contributions at an output net from every noise generator in the circuit. ■ RMS sum of the noise contributions at the output. ■ Equivalent input noise. To run a noise analysis, you must also run an AC sweep analysis.

@ 2023 All Rights Reserved.

DC Analyses	DC Sweep	To evaluate circuit performance in response to a direct current source, you can perform DC analyses. Calculates steady-state voltages, currents, and digital states when sweeping a source, a model parameter, or temperature over a range of values.
Bias Point	Bias Point Details	Bias point data in addition to what is automatically computed in any simulation. PSpice also computes DC sensitivity of a net or part voltage as a function of bias point. It also computes small-signal DC gain, input resistance, and output resistance as a function of bias point.
	DC Sensitivity	Sensitivity of a net or part voltage as a function of bias point.
	Small Signal DC Transfer	Small-signal DC gain, input resistance, and output resistance as a function of bias point.

@ 2023 All Rights Reserved.

Time Domain Analysis	Transient	To evaluate circuit performance in response to time-varying sources, you can run these time-based analyses. Voltages, currents, and digital states tracked over time. For digital devices, set the propagation delays to minimum, typical, and maximum. If digital worst-case timing analysis is enabled, PSpice A/D considers all possible combinations of propagation delays within the minimum and maximum range.
	Fourier	DC and Fourier components of the transient analysis results.
Advanced Analysis		
Multi-Run Analyses	Parametric Sweep (Parametric Analysis)	To perform transient analysis on multiple parameters, perform Parametric analysis. Parametric analysis performs multiple iterations of a specified standard analysis while varying a global parameter, model parameter, component value, or operational temperature. The effect is the same as running the circuit several times, once for each value of the swept variable.
	Temperature (Sweep) (Temperature Analysis)	To perform multiple analyses for different temperatures, run Temperature analysis. PSpice A/D reruns standard analyses at different temperatures configured in the *Simulation Settings*. You require to step operational temperature to run Temperature analysis.

@ 2023 All Rights Reserved.

Statistical Analyses	Monte Carlo	To perform statistical analysis of the circuit, you can perform Monte Carlo analysis. It computes the parameters using random numbers. The Monte Carlo analysis calculates the circuit response to changes in part values by randomly varying all of the model parameters for which a tolerance is specified. For each simulation, PSpice A/D *randomly* varies all device model parameters for which you have defined a tolerance.
	Sensitivity/ worst-case	To perform a sensitivity and worst-case analysis of the circuit, you can run Sensitivity/worst-case analysis. It computes the parameters using the sensitivity data. PSpice A/D computes the probable worst-case response of the circuit in two steps: **1.** Computes component sensitivity to changes in the device model parameters. This means PSpice A/D *nonrandomly* varies device model parameters for which you have defined a tolerance, one at a time for each device and runs a simulation with each change. **2.** Sets *all* model parameters for *all* devices to their worst-case values (assumed to be at one of the tolerance limits) and runs a final simulation.

The multi-run analyses—parametric, temperature, Monte Carlo, and sensitivity/worst-case—result in a series of DC sweep, AC sweep, or transient analyses depending on which basic analyses you enabled.

@ 2023 All Rights Reserved.

Files Required for Simulation

To simulate your design, PSpice needs to know about the following:

- the parts in your circuit and how they are connected.
- what analyses to run.
- the simulation models that correspond to the parts in your circuit.
- the stimulus definitions to test with.

This information is provided in various data files. Some of these are generated by schematic editor, others come from libraries, and still others are user-defined.

Files that Design Entry Tool Generates

In a design entry tool, you need to prepare your circuit for simulation that includes the following:

- placing and connecting part symbols
- defining component values and other attributes
- defining input waveforms
- enabling one or more analyses
- marking the points in the circuit where you want to see simulation results

When you begin the simulation process, the design entry tool first generate files describing the parts and connections in your circuit. These files are the netlist file and the circuit file that PSpice reads before doing anything else.

Netlist File

The netlist file contains a list of the device names and their values, and the connections between the devices. The name that design entry programs generate for this file is `DESIGN_NAME.net`. The netlist file is located in the following directory:

```
<project_directory>\worklib\<design_name>sp_sim_1\
```


@ 2023 All Rights Reserved.

Circuit File

The circuit file contains commands describing how to run the simulation. This file also refers to other files that contain netlist, model, stimulus, and any other user-defined information that apply to the simulation. The name that the design entry programs generate for this file is `PROFILE_name.cir`.

Other Input Files to Configure

Before you start the simulation, PSpice needs to read other files that contain simulation information for your circuit. These are model files, and if required, stimulus files and include files.

Model Library Files

A model library is a file that contains the electrical definition of one or more parts. PSpice uses this information to determine how a part will respond to different electrical inputs.

These definitions take the form of either a:

- model parameter set, which defines the behavior of a part by fine-tuning the underlying model built into PSpice, or
- subcircuit netlist, which describes the structure and function of the part by interconnecting other parts and primitives.

The most commonly used models are available in the PSpice model libraries shipped with your programs. The model library names have a .LIB extension.

Stimulus Files

A stimulus file contains time-based definitions for analog and/or digital input waveforms. You can create a stimulus file using a standard text editor, to create the definition (a typical file extension is .STM).

Note: Not all stimulus definitions require a stimulus file. In some cases, like DC and AC sources, you must use a schematic symbol and set its properties.

Include Files

An include file is a user-defined file that contains:

@ 2023 All Rights Reserved.

- PSpice commands, or
- supplemental text comments that you want to appear in the PSpice output file

You can create an include file using any text editor, such as Notepad. Typically, include file names have a .INC extension.

Preparing and Configuring Input Files

You must first prepare the circuit design using Cadence schematic capture application. Entering the design in a schematic capture application is the most efficient way to draw up the circuit and define the various parameters required for simulation. Once the circuit is entered in the design entry programs, you can configure the input files for analysis by PSpice.

Along with the netlist and circuit files generated by the design entry programs, PSpice searches model libraries, stimulus files, and include files for any information it needs to complete the definition of a part or to run a simulation.

How you configure your model libraries and other files determines the way PSpice uses those files. Most of the configuration is set up for you automatically. However, you can do the following yourself:

- add and delete files from the configuration.
- change the scope of a file: that is, whether the file applies to one profile only, one design only (local) or to any design (global).
- change the search order.

@ 2023 All Rights Reserved.

@ 2023 All Rights Reserved.

2

Preparing a Design for Simulation

To simulate a design with PSpice, the design must have PSpice-compatible parts. This means that simulation information needs to be available for each part.

You can either use PSpice-compatible parts from Cadence-supplied library located under your installation directory or you can create models for parts that you want to simulate.

For any design to be simulated using PSpice, the following criteria need to be met:

- parts for which there are simulation models available and configured. Each part must have a `PSPICETEMPLATE` property in order to netlist correctly for use with PSpice.
- sources of stimulus to the circuit.
- ‘0’ (zero) ground part in designs intended to be simulated by PSpice. If you have used other ground parts, you can rename them to ‘0’ so that they are accepted by PSpice.

@ 2023 All Rights Reserved.

PSpice Libraries

The PSpice libraries contain parts that can be simulated.You can use Cadence-supplied PSpice libraries or PSpice advanced analysis libraries parts to analyze them with PSpice. Cadence-supplied parts are the ones that you install using the Cadence installation setup.

Note: To know more about a particular library (`.lib`), open it in a text editor and read the comments in the file.

PSpice libraries are located in the PSpice sub-folder under your main installation directory. For example, in Capture, you find parts at the following library location:
`<install_dir>\tools\capture\library\pspice`

The location contains the following Cadence-supplied libraries:

- Standard PSpice Libraries
- PSpice Advanced Analysis Libraries

  The PSpice Advanced Analysis Libraries are installed in the `<install_dir>\tools\capture\library\pspice\advanls` installation directory.

Note: All parts in the libraries are not set up for simulation. For example, connectors are parts destined for board layout only and do not have the simulation properties. The libraries contained in the PSpice sub-folder are the only ones set up for simulation.

Standard PSpice Libraries

Standard PSpice libraries contain parts from vendor libraries, special simulation-only parts including stimulus and ground parts, passive parts and breakout parts.

@ 2023 All Rights Reserved.

Parts from Vendor Libraries

The PSpice libraries provide an extensive selection of manufacturers' analog and digital parts. Typically, the library name reflects the kind of parts contained in the library and the vendor that provided the models. For example, `MOTOR_RF.OLB` contains parts and models, respectively, for Motorola-made RF bipolar transistors, and `INFINEON.OLB`, and, so on.

If a design component is not a Cadence-supplied PSpice compatible part, it can still be simulated in PSpice after associating a PSpice model to it.

The PSpice part libraries also include special parts under `special.olb` that you can use for simulation only. These include:

- stimulus parts to generate input signals to the circuit.
- ground parts required by all analog and mixed-signal circuits, which need reference to ground.
- simulation control parts to do things like set bias values.
- output control parts to do things like generate tables and line-printer plots to the PSpice output file.

Note: The parts that are to be used for simulation, but not for board layout, have a `PSPICEONLY=TRUE` property. You can add this (or any) property to your own custom parts to make them simulation-only.

PSpice contains some other parts such as passive parts, breakout parts, some behavioral parts that you can simulate using PSpice.

Passive Parts

The PSpice libraries supply several basic parts based on the passive device models built into PSpice. These parts `analog.olb` are available at this location:
`<install_dir>\tools\capture\library\pspice`

The passive parts are summarized in the following table:

Parts Available	Device Type	PSpice Device Letter
`C C_VAR`	capacitor	C
`L`	inductor	L
`R R_VAR`	resistor	R

@ 2023 All Rights Reserved.

XFRM_LINEAR K_LINEAR	transformer	K and L
T	ideal transmission line	T
TLOSSY	Lossy transmission line	T
TnCOUPLED TnCOUPLEDX KCOUPLE	coupled transmission line	T and K

Breakout Parts

The PSpice libraries supply passive and semiconductor parts with default model definitions that define a basic set of model parameters.

Breakout parts such as CBREAK, DBREAK can be accessed from breakout.olb library at this location: <install_dir>\tools\capture\library\pspice

This way, you can easily do the following:

- assign device and lot tolerances to model parameters for Monte Carlo and sensitivity/ worst-case analyses,
- define temperature coefficients, and
- define device-specific operating temperatures.

The breakout parts are summarized in the following table:

Parts Available	Device Type	PSpice Device Letter
BBREAK	GaAsFET	B
CBREAK	capacitor	C
DBREAKx	diode	D
JBREAKx*	JFET	J
KBREAK	inductor coupling	K
LBREAK	inductor	L
MBREAKx*	MOSFET	M
QBREAKx*	bipolar transistor	Q
RBREAK	resistor	R

@ 2023 All Rights Reserved.

SBREAK	voltage-controlled switch	S
TBREAK	transmission line	T
WBREAK	current-controlled switch	W
XFRM_NONLINEAR	transformer	K and L
ZBREAKN	IGBT	Z

@ 2023 All Rights Reserved.

PSpice Advanced Analysis Libraries

The PSpice Advanced Analysis libraries contain both parameterized and standard parts along with the behavioral models for electromechanical parts. The majority of the parts are parameterized. The parametrized parts have tolerance, distribution, optimizable and smoke parameters that are required by the PSpice Advanced Analysis tools.

Standard parts in the Advanced Analysis libraries are similar to parts in the standard PSpice libraries.

The parametrized parts are associated with template-based PSpice models. An important advantage of using the template-based PSpice models is that you can pass simulation parameters as properties from the schematic editor. There is no need to edit the model itself to change a parameter value. For example, if a template-based model is associated with a part, the simulation parameters that you specify on an instance of the part in your design will be passed to the model.

Standard PSpice parts are associated with device characteristic curve based PSpice models, where the model needs to be edited to change a simulation parameter.

You can use parametrized parts from advanced analysis libraries if you want to analyze the part with an advanced analysis option.

Behavioral Parts

Behavioral parts enable you to define how a block of circuitry should work without defining each discrete component. Behavioral parts are available at this library location:
`<install_dir>\tools\capture\library\pspice\advanls`

Analog Behavioral Parts

These parts use analog behavioral modeling (ABM) to define behavior as a mathematical expression or lookup table. The PSpice libraries provide ABM parts that operate as math functions, limiters, Chebyshev filters, integrators, differentiators, and others that you can customize for specific expressions and lookup tables. You can also create your own ABM parts.

Note: `abm.olb` is available at this location:
`<install_dir>\tools\capture\library\pspice`

@ 2023 All Rights Reserved.

Digital Behavioral Parts

These parts use special behavioral primitives to define each part's functional and timing behavior. These primitives are:

`LOGICEXP`	to define logic expressions
`PINDLY`	to define pin-to-pin delays
`CONSTRAINT`	to define constraint checks

Many of the digital parts provided in the PSpice libraries are modeled using these primitives. You can also create your own digital behavioral parts using these primitives.

Related Topics

- Behavioral Simulation Models

@ 2023 All Rights Reserved.

Creating a PSpice-Enabled Project

To simulate a design with PSpice A/D, you require to create a PSpice-enabled project for simulation in a schematic capture application.

For a project to be PSpice-enabled, you need to select either of the following options depending on your schematic capture application:

- Select the *Product option* as PSpice when launching the schematic capture application.
- Select *Enable PSpice Simulation* at the time of project creation.

Let us take an example to create a project in OrCAD X Capture.

To create a new project for simulation, do the following:

1. From the *File* menu in Capture, select *New – Project*.

   The *New Project* dialog box appears.

2. In the *Name* field, enter the name for the new project.
3. In the *Location* field, enter the path where you want the new project files to be stored, or use the browse button to locate the directory.
4. Select the *Enable PSpice Simulation* check box if you intend to have simulation capabilities in your PCB design.

5. Click *OK*.

   A new project is created and a schematic page is displayed.

   The PSpice toolbar options are enabled at the upper-right corner of the schematic capture application that contains icons to run simulations, edit simulation profiles, view simulation results, modeling applications and options to add markers.

@ 2023 All Rights Reserved.

Related Topics

- Simulating System Capture Designs

Defining Stimuli

To simulate your circuit, you need to connect one or more source parts that describe the input signal that the circuit must respond to.

The PSpice libraries supply several source parts that are described in the tables depend on:

- the kind of analysis you are running,
- whether you are connecting to the analog or digital portion of your circuit, and
- how you want to define the stimulus: using the Stimulus Editor, using a file specification, or by defining part property values.

Analog Stimuli

Analog stimuli include both voltage and current sources. The following table shows the part names for the corresponding voltage sources:

Input applied	Part used
`DC bias`	VDC or VSRC
`AC magnitude and phase`	VAC or VSRC
`exponential`	VEXP or VSTIM
`periodic pulse`	VPULSE or VSTIM
`piecewise-linear`	VPWL or VSTIM
`piecewise-linear that repeats forever`	VPWL_RE_FOREVER or VPWL_F_RE_FOREVER

@ 2023 All Rights Reserved.

`piecewise-linear that repeats n times`	VPWL_N_TIMES or VPWL_F_N_TIMES**
`frequency-modulated sine wave`	VSFFM or VSTIM
`sine wave`	VSIN or VSTIM

Note: ** VPWL_F_RE_FOREVER and VPWL_F_N_TIMES are file-based parts; the stimulus specification is saved in a file and adheres to PSpice netlist syntax.

➡ To determine the part name for an equivalent current source, replace the first V in the part name with I in the table of voltage source parts.

For example, the current source equivalent to VDC is IDC, to VAC is IAC, to VEXP is IEXP, and so on.

Using VSTIM and ISTIM

You can use VSTIM and ISTIM parts to define any kind of time-based input signal. To specify the input signal itself, you need to use the Stimulus Editor.

If you want to run more than one analysis type, including a transient analysis, you need to use either of the following:

- time-based stimulus parts with AC and DC properties
- VSRC or ISRC parts

Time-based stimulus parts with AC and DC properties

The time-based stimulus parts to be used to define a transient, DC, and/or AC input signal are listed in the following table:

VEXP	**IEXP**
`VPULSE`	IPULSE
`VPWL`	IPWL
`VPWL_F_RE_FOREVER`	IPWL_F_RE_FOREVER
`VPWL_F_N_TIMES`	IPWL_F_N_TIMES
`VPWL_RE_FOREVER`	IPWL_RE_FOREVER

@ 2023 All Rights Reserved.

VPWL_RE_N_TIMES	IPWL_RE_N_TIMES
VSFFM	ISFFM
VSIN	ISIN

In addition to the transient properties, each of these parts also has a DC and AC property. When you use one of these parts, you must define all the transient properties. However, it is common to leave DC or AC undefined (blank). When you give them a value, the syntax you need to use is as follows.

Property	Syntax
DC	DC_value[units]
AC	magnitude_value[units] [phase_value]

VSRC or ISRC parts

The VSRC and ISRC parts have one property for each analysis type: DC, AC, and TRAN. You can set any or all of them using PSpice netlist syntax. When you give them a value, the syntax you need to use is as follows:

Property	Syntax
DC	DC_value[units]
AC	magnitude_value[units] [phase_value]
TRAN	time-based_type (parameters)

where time-based_type is EXP, PULSE, PWL, SFFM, or SIN, and the parameters depend on the time-based_type.

Note: If you are running only a transient analysis, it is recommended that you use a VSTIM or ISTIM part if you have the standard package, or one of the other time-based source parts that has properties specific for a waveform shape.

Digital Stimuli

The table shows the parts used when the following input is applied:

Input applied transient analyses	Part used

@ 2023 All Rights Reserved.

`signal or bus (any width)`	DIGSTIMn
`clock signal`	DIGCLOCK
`1-bit signal`	STIM1
`4-bit bus`	STIM4
`8-bit bus`	STIM8
`16-bit bus`	STIM16
`file-based signal or bus (any width)`	FILESTIMn

European Notation for Component Values

You can use the European notation, such as 2K2, to assign values to resistors, capacitors, and inductors. Assigning values in this format reduces the possibility of errors while reading component values from screen or from a print out of the schematic. The alphabets that can be used in the 2K2 notation are listed in the following table:

Alphabet used	**Stands for**
F(f)	femto
P(p)	pico
N(n)	nano
U(u)	micro
M(m)	milli
K(k)	kilo
MEG(meg)	mega
G(g)	giga
T(t)	tera
Example	
2M2	2.2M
2MEG2	2.2MEG
4L5	inductor of 4.5 henry
2K2	2.2K

@ 2023 All Rights Reserved.

Alphabet used	Stands for
12C2	capacitor of 12.2 farads
5R4	resistor of 5.4 ohms
2p2	2.2 p ($2.2*10^{-12}$)

Checklist for Simulation Setup

Before you start simulating a design in PSpice, you need to verify if your circuit is set up for simulation.

Steps	Description	Know More
`Set component values and other properties`	An overview of vendor, passive, breakout, and behavioral parts.	Standard PSpice Libraries
	Things to consider when specifying values for part properties.	Specifying values for part properties
	How to define values using variable parameters, functional calls, and mathematical expressions.	Using global parameters and expressions for values.
`Define power supplies`	An overview of DC power for analog circuits and digital power for mixed-signal circuits.	Specifying Power Supply
`Define input waveforms.`	An overview of DC, AC, and time-based stimulus parts.	Defining Stimuli
`Set up one or more analyses.`	Procedures, general to all analysis types, to set up and start the simulation. The analyses include DC, AC, transient, parametric, temperature, Monte Carlo, sensitivity/worst-case, and digital analyses.	Analyses to Run with PSpice A/D
`Place markers.`	How to display results in PSpice by picking nets in your design.	Using Markers

@ 2023 All Rights Reserved.

`Limiting data file size`	How to limit the data file size.	Data collection options for simulation profiles

Troubleshooting a Design

The following tables list the most commonly encountered problems and where to find out more about what to do and what to check in a design.

Make sure that...	Know more
`The model libraries, stimulus files, and include files are configured.`	Configuring model libraries
`The parts you are using have models.`	Unmodeled parts and Defining part properties needed for simulation
`You are not using unmodeled pins.`	Unmodeled pins
`You have defined the grounds.`	Missing ground
`Every analog net has a DC path to ground.`	Missing DC path to ground
`The part template is correct.`	Defining part properties needed for simulation
`Hierarchical parts, if used, are properly defined.`	OrCAD X Capture User Guide
`Ports that connect to the same net have the same name.`	OrCAD X Capture User Guide

@ 2023 All Rights Reserved.

3

Creating and Editing Models

A model defines the electrical behavior of a part. On a schematic page, this correspondence is defined by the Implementation property on the part, which is assigned the model name.

Depending on the device type that it describes, a model is defined as one of the following:

- a model parameter set
- a subcircuit netlist

Both ways of defining a model are text-based, with specific rules of syntax.

Models Defined as Model Parameter Sets

PSpice A/D has built-in algorithms or models that describe the behavior of many device types. The behavior of these *built-in models* is described by a *set of model parameters*.

You can define the behavior for a device that is based on a built-in model by setting all or any of the corresponding model parameters to new values using the PSpice .MODEL syntax. For example:

```
.MODEL MLOAD NMOS
+ (LEVEL=1 VTO=0.7 CJ=0.02pF)
```

Note: In addition to the analog models built into PSpice, the .MODEL syntax applies to the timing and I/O characteristics of digital parts.

Models Defined as Subcircuit Netlists

For some devices, there are no PSpice built-in models that can describe their behavior fully. These types of devices are defined using the PSpice .SUBCKT/.ENDS or *subcircuit syntax* instead.

Subcircuit syntax includes:

- *Netlists* to describe the structure and function of the part.

@ 2023 All Rights Reserved.

- *Variable input parameters* to fine-tune the model.

For example:

```
* FIRST ORDER RC STAGE
.SUBCKT LIN/STG IN OUT AGND
+ PARAMS: C1VAL=1 C2VAL=1 R1VAL=1 R2VAL=1
+         GAIN=10000
C1 IN  N1   {C1VAL}
C2 N1  OUT  {C2VAL}
R1 IN  N1   {R1VAL}
R2 N1  OUT  {R2VAL}
EAMP1 OUT AGND VALUE={V(AGND,N1)*GAIN}
.ENDS
```

To find out more about PSpice A/D command and netlist syntax, refer to the *PSpice A/D Reference Guide*.

Organization of Models

The key concepts behind model organization are as follows:

- Model definitions are saved in files called model libraries.
- Model libraries must be configured so that PSpice A/D searches them for definitions.
- Depending on the configuration, model libraries are available either to a specific profile, to a specific design or to all (global) designs.

Model libraries

Device model and subcircuit definitions are organized into model libraries. Model libraries are text files that contain one or more model definitions. Typically, model library names have a `.LIB` extension.

Most model libraries contain models of similar type. For vendor-supplied models, libraries are also partitioned by manufacturer. For example, MOTOR_RF.LIB contains models for Motorola-made RF bipolar transistors.

To find out more about the models contained in a model library, read the comments in the file header.

Note: You can use the PSpice Model Editor, or any standard text editor, to view model definitions in libraries.

@ 2023 All Rights Reserved.

Model Library Configuration

PSpice A/D searches model libraries for the model names specified by the MODEL implementation for parts in your design. These are the model definitions that PSpice A/D uses to simulate your circuit.

For PSpice A/D to locate these model definitions, you must configure the libraries. This means:

- Specifying the directory path or paths to the model libraries.
- Naming each model library that PSpice A/D should search and listing them in the desired search order.
- Assigning global, design or profile scope to the model library.

To optimize the model library search, PSpice A/D uses indexes.

Global vs. Design vs. Profile Models and Libraries

Model libraries and the models they contain have either profile, design or global application to your designs.

Profile models

Profile models apply to one profile. You can create models using the Model Editor and then manually configure the new libraries for a specific profile.

Example usage: To set up device and lot tolerances on the model parameters for a particular part instance when running a Monte Carlo or sensitivity/worst-case analysis using a specific profile.

Design models

Design models apply to one design. The *schematic page editor* automatically creates a design model whenever you modify the model definition for a part instance on your schematic page. You can also create models externally and then manually configure the new libraries for a specific design.

Example usage: To set up device and lot tolerances on the model parameters for a particular part instance when running a Monte Carlo or sensitivity/worst-case analysis.

@ 2023 All Rights Reserved.

Global models

Global models are available to all designs you create. The *part editor* automatically creates a global model whenever you create a part with a new model definition. The Model Editor also creates global models. You can also create models externally and then manually configure the new libraries for use in all designs.

PSpice A/D searches profile libraries before design libraries and design libraries before global libraries.

Nested Model Libraries

Besides model and subcircuit definitions, model libraries can also contain references to other model libraries using the PSpice .LIB syntax. When searching model libraries for matches, PSpice A/D also scans these referenced libraries.

Example: Suppose you have two custom model libraries, `MYDIODES.LIB` and `MYOPAMPS.LIB`, that you want PSpice A/D to search any time you simulate a design. Then you can create a third model library, `MYMODELS.LIB`, that contains these two statements:

```
.LIB mydiodes.lib
.LIB myopamps.lib
```

and configure `MYMODELS.LIB` for global use. Because `MYDIODES.LIB` and `MYOPAMPS.LIB` are referenced from `MYMODELS.LIB`, they are automatically configured for global use as well.

PSpice-Provided Models

The model libraries that you initially install with your PSpice A/D programs are listed in `NOM.LIB`. This file demonstrates how you can nest references to other libraries and models.

If you click the Configuration Files tab in the Simulation Settings dialog box and view the Library files list immediately after installation, you see the `NOM.LIB` entry in the Library files list. The icon means that this model library and any of the model libraries it references contain global model definitions.

Model Library Data

Information contained in PSpice model libraries can be classified as:

- Simulation information
- Device information

@ 2023 All Rights Reserved.

Simulation information

Simulation information is also termed as model information and is used while simulating the models. Depending on the method of creation, PSpice simulation models are of two types:

- Device characteristic curves-based models
- Template-based models

Device characteristic curves-based models

Simulation models based on the device characteristic curves are the models that are historically being used in the PSpice A/D flow. You can extract device parameters based on the device characteristic curves in the data sheets.

Template-based models

These simulation models are based on PSpice-provided templates and are a new addition to the PSpice model library. Simulation models that are based on PSpice A/D provided templates are also referred to as parameterized models. Parameterized models are specified in terms of model parameters. Changing a parameter changes the behavior of the model. Template-based PSpice models describe the analog simulation behavior of a device in terms of parametric equations. These models are of the .SUBCKT type. The .SUBCKT wrapper enables symbol properties to be passed as parameters to the simulator.

The PSpice-provided templates are available in the `TEMPLATES.LIB` file. This is an encrypted file, and template-based models are wrappers to this file.

The main advantage of using template-based models is that simulation parameter values can be passed as properties from the design entry tool. Besides this, parameterized models are best suited to be used with PSpice Advanced Analysis, and also for performing statistical analysis for variations in model parameters.

Device information

Information that is specific to each device, such as simulation parameter tolerance and maximum operating conditions, is termed as device information. The device information is stored in the device property file and is required by PSpice Advanced Analysis. To know more about the device property file, see *Appendix A, Property Files* in the *PSpice Advanced Analysis User Guide*.

@ 2023 All Rights Reserved.

You can use the Model Editor to add device information to a model. For template-based simulation models, you can add smoke and tolerance information. For other simulation models, you can add only the smoke information.

Tolerance information is required to perform a Monte Carlo and Sensitivity/Worst-Case analysis. To know more about Monte Carlo and Sensitivity/Worst-Case analysis, see *PSpice Advanced Analysis User Guide*.

The tolerance information is added in the Simulation Parameters frame but smoke information can be added or modified only if you have Advanced Analysis installed.

Device characteristic curves-based models vs. Template-based models

The following table highlights the differences between two types of simulation models:

Table 3-1 Differences between device characteristic curves-based and template-based models

Device characteristic curves-based PSpice models	Template-based PSpice models
They are based on device characteristic curves. Device characteristic curves can be obtained from device datasheets.	They are based on PSpice-provided templates.
They cannot be used for Advanced Analysis Sensitivity, Monte Carlo, and Optimizer runs. They can be used for Advanced Analysis smoke test only if smoke information is explicitly added.	They can be used for all Advanced Analysis runs, such as Sensitivity, Smoke, Monte Carlo, and Optimizer.
All simulation information is contained in the model itself.	Models are wrappers to a template model file. Therefore, both the model and the template model are required for simulation.

@ 2023 All Rights Reserved.

Table 3-1 Differences between device characteristic curves-based and template-based models

Device characteristic curves-based PSpice models	Template-based PSpice models
The PSpiceTemplate property must be attached to the symbol for generating the PSpice netlist. To know more about the PSpiceTemplate property, see PSPICETEMPLATE.	The PSpiceTemplate property is not required on the symbol. The PORT_ORDER information present in the device property file is used for generating the PSpice netlist. **Note:** The model library and the device property file must have the same name and must be at the same location. To know more about the device property file, see *Appendix A, Property Files* in the *PSpice Advanced Analysis User Guide*.
To modify a simulation property, you need to edit the simulation parameter. This implies that you either update the values in the original model or create an instance model for the design. To know more, see Using the Model Editor to edit the D1 diode model.	Simulation parameter values can be passed as properties from the schematic editor. This implies that instance-specific values of simulation parameters can be passed from the schematic editor, without changing the original model. To know more, see Changing the level for CA1458.
By default, tolerance and smoke information is not available in the models created using the Model Editor.	Tolerance and smoke information for the device is available in the device property file associated with the model library.
Editing model text to add DEV and LOT information does not make the model compatible with Advanced Analysis Monte Carlo run.	If PSpice Advanced Analysis is installed, default values for the smoke parameters are visible through the Model Editor user interface.

Note: The shape and size of the part symbol generated by the Model Editor for the template-based models and device characteristic curves-based models may be different.

@ 2023 All Rights Reserved.

Tools to Create and Edit Models

There are two tools that you can use to create and edit model definitions.

- **The Model Editor**

  Use the Model Editor when you want to:

  - derive models from data sheet curves provided by manufacturers.
  - create models based on PSpice-provided templates.
  - modify the behavior of a Model Editor-supported model.
  - edit the PSpice command syntax (text) for .MODEL and .SUBCKT definitions.

  Note: For template-based models, the model text is read-only and cannot be edited using the Model Editor.

- **Capture**

  Use the Create Subcircuit Format Netlist command in Capture when you have a hierarchical level in your design that you want to set up as an equivalent part with behavior described as a subcircuit netlist (.SUBCKT syntax).

  Note: The Create Subcircuit Format Netlist command does not help you create a hierarchical design. You need to create this yourself before using the Create Subcircuit Format Netlist command. For information on hierarchical designs and how to create them, refer to the *OrCAD X Capture User Guide*.

  Note: If you created a subcircuit definition using the Create Subcircuit Format Netlist command and want to alter it, use the Model Editor to edit the definition, or modify the original hierarchical schematic and run Create Subcircuit Format Netlist again to replace the definition.

@ 2023 All Rights Reserved.

Ways to Create and Edit Models

This section is a roadmap to other information in this chapter. Find the task that you want to complete, then go to the referenced sections for more information.

@ 2023 All Rights Reserved.

Note: If the Model Editor does not support the device type for the model definition that you

If you want to...	Then do this...	To find out more, see this...
■ **Create a model from scratch** and automatically create a symbol for it to use in any schematic.	Start the Model Editor and enable/disable automatic symbol creation as needed. Then, create or view the model.	Running the Model Editor Alone on page 51.
■ **Create a model from scratch without a symbol** and have the model definition available to any design.		
■ **View model** characteristics for a part.		
■ **Create a new model by copying an existing model**	Copy the text of an existing model in a text editor and rename the file. or From the Model menu, choose the Copy From command.	Model Editor Help
■ **Create or edit the model for an existing symbol** and incorporate the changes in all schematics that use that symbol.	First, create or load the symbol in the design entry tool, and then edit the model using the Model Editor. You can edit models by: ■ changing parameter values in the Parameters window. ■ editing text in the Model Text window.	Running the Model Editor from the schematic editor. Editing model text

@ 2023 All Rights Reserved.

If you want to...	Then do this...	To find out more, see this...
■ **Edit a model** to add smoke information.	Start the Model Editor and open the library with the model, and then add the smoke information. **Note:** This feature is available only if you have Advanced Analysis installed on your machine.	
■ **Define tolerances** on model parameters for statistical analysis.	Select the part instance on your schematic and then edit the model text using the Model Editor. **Note:** For template-based PSpice models, tolerance information can be added in the Postol and Negtol columns of the Simulation Parameters window.	Editing model text.
■ **Test behavior variations** on a part. ■ **Refine a model** before making it available to all schematics.	Select the part instance on your schematic and then edit the model using either: ■ the Model Editor, or ■ editing the Model Text in a text editor.	Running the Model Editor from the schematic editor on page 63. Starting the Model Editor.
■ **Derive subcircuit definitions** from a hierarchical schematic.	In the Project Manager, select the .DSN file. From the Tools menu, choose Create Netlist, select the PSpice tab, and then check the Create Subcircuit Format Netlist check box.	Using the Create Subcircuit Format Netlist Command (Capture only).

want to create, then you can use a standard text editor to create a model definition using the PSpice .MODEL and .SUBCKT command syntax. Remember to configure the new model library (see Configuring Model Libraries).

@ 2023 All Rights Reserved.

Using the Model Editor

The Model Editor converts information that you enter from the device manufacturer's data sheet into either:

- model parameter sets using PSpice .MODEL syntax, or
- subcircuit netlists using PSpice .SUBCKT syntax.

  Note: The Extract Model view in the Model Editor does not support the following subcircuit constructs:

  - optional nodes construct, OPTIONAL:
  - variable parameters construct, PARAMS:
  - local .PARAM command
  - local .FUNC command

  To refine the subcircuit definition for these constructs, use the Model Text view in Model Editor, described in Editing model text.

The Model Editor then saves these definitions to model libraries that PSpice A/D can search when looking for simulation models.

Figure 3-1 Relationship of the Model Editor to design entry tool and PSpice A/D.

@ 2023
All Rights Reserved.

Note: By default, the Model Editor creates or updates model libraries. To create an exported model file, choose the Export command from the Model menu and configure it as an include file. For more information, see How PSpice A/D uses model libraries.

Ways to use the Model Editor

You can use the Model Editor in the following ways:

- **To define a new model, and then automatically create a part.** Any new models and parts are automatically available to any design. To find out more, see Running the Model Editor Alone on page 51.
- **To define a new model only (no part).** You can optionally turn off the part creation feature for new models. The model definition is available to any design, for example, by changing the model implementation for a part instance. To find out more, see Running the Model Editor Alone on page 51.
- **To edit a model definition for a part instance on your schematic.** This means you need to start the Model Editor from the schematic editor after selecting a part instance on your schematic. The schematic editor automatically attaches the new model implementation (that the Model Editor creates) to the selected part instance. To find out more, see Running the Model Editor from the schematic editor.
- **To examine or verify the electrical characteristics of a model without running PSpice A/D.** This means you can use the Model Editor alone to:
  - check characteristics of a model quickly, given a set of model parameter values, or
  - compare characteristic curves to data sheet information or measured data.

  To find out more, see Running the Model Editor Alone on page 51.
- **To add and modify a model definition for parameterized or template-based PSpice models.** This means you can create new parameterized models. You can also edit the existing models in the parameterized libraries to modify the values of simulation parameters.
- Adding and editing smoke data to the models supported by the Model Editor. If you have Advanced Analysis installed on your machine, you can use the Model Editor to add smoke information to device characteristic curves-based PSpice models. Editing of smoke information is possible for all types of PSpice models.

Running the Model Editor Alone

Run the Model Editor alone if you want to do any of the following:

@ 2023 All Rights Reserved.

- create a model and use the model in any design (and automatically create a part),
- create a model and have the model definition available to any design (without creating a part), or
- examine or verify the characteristics of a given model without using PSpice A/D.

Running the Model Editor alone means that the model you are creating or examining is not currently tied to a part instance on your schematic page or to a part editing session.

Note: You can edit models in the Edit Model View only for device types that the Model Editor supports.

Starting the Model Editor

To start the Model Editor alone

1. From the Start menu, point to Programs, installed OrCAD X release, choose *PSpice Accessories* and then choose *Model Editor*.
2. From the File menu, choose *New* or *Open*.

If you have already started the Model Editor from the design entry tool and want to continue working on new models, then:

1. Save the opened model library.
2. Open or create a different model library.
3. Get a model, or create a new one.

Creating Models Using the Model Editor

Using the Model Editor, you can create models from scratch. The Model Editor supports creation of PSpice models based on device characteristic curves as well as templates. This section covers:

- Creating models based on device characteristic curves
- Creating models based on PSpice templates

Creating models based on device characteristic curves

1. In the Model Editor, open a library.

@ 2023 All Rights Reserved.

2. From the Models menu, choose New.
3. Specify the name of the new model in the Model Name text box.
4. Select the Use Device Characteristic Curves option.
5. From the From Model drop-down list, select the device type and click OK.

   Note: Depending of the device type, you may have to provide some other details. For example, if the device type is Bipolar Transistor, you will also need to specify if the BJT will be of NPN or PNP type.

   All the device characteristic curves for the device and the simulation parameters are displayed. You can now characterize the models by either using data sheets or editing simulation parameter values.

@ 2023 All Rights Reserved.

Ways to characterize models

The following figure shows two ways to characterize PSpice models using the Model Editor.

Figure 3-2 Process and data flow for the Model Editor.

Testing and verifying models created with the Model Editor

Each curve in the Model Editor is defined only by the parameters being adjusted. For the diode, the forward current curve *only* shows the part of the current equation that is associated with the forward characteristic parameters (such as IS, N, Rs).

@ 2023 All Rights Reserved.

However, PSpice A/D uses the *full* equation for the diode model, which includes a term involving the reverse characteristic parameters (such as ISR, NR). These parameters could have a significant effect at low current.

This means that the curve displayed in the Model Editor does not exactly match what is displayed in PSpice A/D after a simulation. Be sure to test and verify models using PSpice A/D. If needed, fine-tune the models.

Creating models from data sheet information

The most common way to characterize models is to enter data sheet information for each device characteristic. After you are satisfied with the behavior of each characteristic, you can have the Model Editor estimate (or *extract*) the corresponding model parameters and generate a graph showing the behavior of the characteristic. This is called the fitting process.

You can repeat this process, and when you are satisfied with the results, save them; the Model Editor creates model libraries containing appropriate model and subcircuit definitions.

Note: When specifying operating characteristics for a model, you can use typical values found on data sheets effectively for most simulations. To verify your design, you may also want to use best- and worst-case values to create separate models, and then swap them into the circuit design.

Analyzing the effect of model parameters on device characteristics

You can also edit model parameters directly and see how changing their values affects a device characteristic. As you change model parameters, the Model Editor recalculates the behavior of the device characteristics and displays a new curve for each of the affected ones.

How to fit models

For a given model, the Model Editor displays a list of the device characteristics and a list of all model parameters and performance curves in the following figure.

@ 2023 All Rights Reserved.

For more information about the characteristics of devices supported by the Model Editor, refer to the *PSpice A/D Reference Guide*.

Figure 3-3 Model Editor workspace with data for a bipolar transistor.

To fit the model

1. For each device characteristic that you want to set up:
   - **a.** In the Spec Entry frame, click the tab of the device characteristic.
   - **b.** Enter the device information from the data sheet.
2. From the Tools menu, choose Extract Parameters to extract all relevant model parameters for the current specification.

@ 2023 All Rights Reserved.

A check mark appears in the Active column of the Parameters frame for each extracted model parameter. To keep a parameter value fixed, check the Fixed field corresponding to the parameter.

3. Repeat steps 1 and 2 until the model meets target behaviors.

To view updated performance curves

1. On the toolbar, click the Update Graph button .

Note: If you view performance curves before fitting, then your data points and the curve for the current model specification may not match.

Model Editor-supported device types based on device characteristic curves

Device types that the Model Editor models using the .MODEL statement are based on the models built into PSpice A/D.

The following table summarizes the device types for which you can create PSpice models based on characteristic curves.

Table 3-2 Device characteristic curves-based device types supported in Model Editor

This part type...	Uses this definition form...	And this name prefix...
diode	.MODEL	D
bipolar transistor	.MODEL	Q
bipolar transistor, Darlington model	.SUBCKT	X
IGBT	.MODEL	Z
JFET	.MODEL	J
MOSFET	.MODEL	M
operational amplifier	.SUBCKT	X
voltage comparator	.SUBCKT	X
voltage regulator	.SUBCKT	X
voltage reference	.SUBCKT	X

@ 2023 All Rights Reserved.

Table 3-2 Device characteristic curves-based device types supported in Model Editor

This part type...	Uses this definition form...	And this name prefix...
magnetic core	.MODEL	K

Note: The model parameter defaults used by the Model Editor are different from those used by the models built into PSpice A/D.

Creating models based on PSpice templates

An important advantage of using the template-based PSpice models is that you can pass simulation parameters as properties from the schematic editor. This implies that you can have instance-specific values for the model parameters. To know more about passing parameter values as properties from the schematic editor, see Changing the level for CA1458.

For a description of template-based models supported by the Model Editor, see Model Editor-supported device types based on PSpice templates.

To create a template-based PSpice model, complete the following steps.

1. In Model Editor, create a new library or open an existing library.
2. From the Model menu, choose New.
3. Specify the name of the new model in the Model Name text box.
4. Select the Use Templates option.
5. From the From Model drop-down list, select the device type.

   Depending of the device type, you may have to provide some other details. For example, if the device type is Bipolar Transistor, you will also need to specify if the BJT will be of NPN or PNP type.

6. Click OK.

   The Simulation parameters window appears with the default values of all simulation parameters. These values are editable and can be modified as required.

   Notice that the Model Text window of a template-based PSpice model is not editable, it is read-only. Also, the model text does not display the port information under the .SUBCKT statement.

@ 2023 All Rights Reserved.

Adding tolerance information

While creating template-based simulation models, you can add tolerance information using the Model Editor user interface. Tolerance information is required only for Advanced Analysis Monte Carlo and Sensitivity runs and not for simulating the models.

The Postol, Negtol, and the Distribution columns are used to specify the tolerance information. In the Postol and Negtol columns, specify the positive and negative tolerances, respectively, for each of the simulation parameters. Adding a tolerance value enables the Distribution field for the parameter.

PSpice Advanced Analysis Monte Carlo uses original (nominal) parameter values to calculate results for each specification's first Monte Carlo run. For the subsequent Monte Carlo runs, Advanced Analysis uses distribution functions to randomly pick new parameter values within the manufacturer's tolerance.

The possible distribution types are:

- FLAT - Use the flat distribution function if you want an equal probability of one parameter value being chosen over another.

  Use the flat distribution function if you want an equal probability that one parameter value will be chosen versus another.

- BSIMD.4.2 - Use the bimodal distribution function if you want to represent the probability of a manufactured component falling in the outer range of tolerance values.

  Use the bimodal distribution function if you want to represent the probability of a manufactured component failing in the outer range of tolerance values. For example, a resistor with a 10% tolerance might really only be available in the 5% to 10% range because the 1% to 4% resistors are picked out and sold as resistors with 5% tolerance.

- GAUSS0.4 - Use the Gaussian distribution function if you want a bell curve probability that one parameter value will be chosen versus another.

  Use the Gaussian distribution function if you want a bell curve probability that one parameter value will be chosen versus another.

- SKEW.4.8 - Use the skewed distribution function if you want to weigh the probability of one parameter value being chosen versus another.

  Use the skewed distribution function if you want to weigh the probability of one parameter value being chosen versus another.

Note:

@ 2023 All Rights Reserved.

By default, the distribution type is FLAT. The distribution type influences the Sensitivity and Monte Carlo analysis. To know more about Sensitivity and Monte Carlo analysis, see *PSpice Advanced Analysis User Guide*.

Model Editor-supported device types based on PSpice templates

The following table lists the device types for which template-based models can be created using the Model Editor:

Table 3-3 Template-based device types supported in Model Editor

This part type...	And this name prefix...
diode	X
bipolar transistor	X
IGBT	X
JFET	X
Power MOSFET	X
operational amplifier	X
voltage regulator	X
magnetic core	K

Importing an existing model

You can import third-party or vendor-provided Spice models into a format understood by the Model Editor. Importing a model enables editing the model using the Model Editor user interface.

1. Open the Model Editor.

   From the Start menu, point to installed OrCAD X release in the Programs folder, choose PSpice Accessories and then choose Model Editor.

2. Open a model library.

   From the File menu, choose New or Open.

3. From the Model menu, choose Import.

4. Select the file containing the model definition and select Open.

@ 2023 All Rights Reserved.

The imported model appears in the model library. Although, only the first model of the selected library file is imported to the Model Editor, it is recommended that the file selected in step 4 should contain only one model definition.

Caution

The device property file associated with the model is not imported.

Enabling and disabling automatic part creation

Part creation in the Model Editor is optional. By default, automatic part creation is enabled. However, if you previously disabled part creation, you will need to enable it before creating a new model and part.

Instead of using the PSpice A/D default part set for new models, you can have the Model Editor use your own set of standard parts. To find out more, see Basing New Parts on a Custom Set of Parts.

To automatically create parts for new models

1. From the Tools menu, choose Options.
2. Select the Always Create Part when Saving Model option if it is not already checked.
3. Under Schematic Editor, select the design entry tool name.
4. Under Save Part To, enter the name of the part library for the new part. Choose either:
   - Part Library Path Same As Model Library to create or open library file (*.olb) in Capture or a directory in Design Entry HDL that has the same name prefix as the currently open model library (*.LIB).

     Example: In Capture, if the model library is MYPARTS.LIB, then the Model Editor creates the part library MYPARTS.OLB. Similarly, in Design Entry HDL, if the model library is MYPARTS.LIB, then the Model Editor creates a part directory with the name MYPARTS.
   - User-Defined Part Library, and then enter a file name in the Part Library Name text box.

Note: If you select a user-defined Part library, the Model Editor saves all new parts to the specified file until you change it.

@ 2023 All Rights Reserved.

Saving global models (and parts)

When you save your changes, the Model Editor does the following for you:

- In capture:
  - Saves the model definition to the model library that you originally opened.
  - If you had the automatic part creation option enabled, saves the part definition to MODEL_LIBRARY_NAME.OLB.
- In Design Entry HDL
  - Saves the model definition to the model library that you originally opened.
  - If you had the automatic part creation option enabled, saves the part definition to a `MODEL_LIBRARY_NAME` directory that has subfolders for each part. In such cases the part name is same as the model name.
  - Creates a `CDS.LIB` file. The `CDS.LIB` file defines the location of the part directory created by the Model Editor.

If you want to save the open model library to a new library, then:

1. From the File menu, choose Save As.
2. Enter the name of the new model library.

If you want to save only the model definition that you are currently editing to a different library, then

1. From the Model menu, select Export.
2. Enter the name of the new file.

   Note: When you use the Export command, the model definition is saved with a `.MOD` extension. You cannot export multiple models to a single MOD file. Exporting a model to the same file overwrites the original contents of the MOD file.
3. If you want PSpice A/D to search this file automatically, configure it in the design entry tool (using the Library files list in the Configuration Files tab on the Simulation Settings dialog box).

You cannot export multiple models to a single MOD file. Exporting a model to the same file overwrites the original contents of the MOD file.

@ 2023 All Rights Reserved.

To save the new model (and part)

1. From the File menu, choose Save to update *MODEL_LIBRARY_NAME*.LIB, *MODEL_LIBRARY_NAME*.PRP (and, if you enabled part creation, *MODEL_LIBRARY_NAME* directory), and save them to disk.

Note: To simulate the model, add the Model Library (.LIB) to the project using the PSpice menu, Edit Profile, Simulation Settings dialog box. Click the Configuration Files tab, click Library in the Category field, browse to the Model Library, and click the Add to Design button.

Running the Model Editor from the schematic editor

Start the Model Editor from the schematic editor when you want to:

- in Capture to define tolerances on model parameters for statistical analysis (see Example: Editing a Q2N2222 Instance Model)
- test behavior variations on a part, or
- refine a model before making it available to all designs.

This means editing models for part instances on your schematic page. When you select a part instance and edit its model, the Model Editor automatically creates an *instance model* that you can then change.

Once you have started the Model Editor, you can proceed with entering data sheet information and model fitting as described in How to fit models.

You can also use the Model Editor to view the syntax for a model definition. When you have finished viewing, be sure to quit the Model Editor without saving the library, so that the schematic page editor does not create an instance model.

Note: When the Model Editor is invoked from a schematic editor, the part creation feature is disabled.

What is an instance model?

An instance model is a *copy* of the part's original model. The copied model is local to the design. You can customize the instance model without impacting any other design that uses the original part from the library.

Instance models are created only when you want to edit models from global libraries. If you open a model for editing from a local library, after editing, the model will be saved in the same local library. For more information on global and local libraries see Global vs. Design vs. Profile Models and Libraries.

@ 2023 All Rights Reserved.

When the schematic editor creates the copy, it saves a copy of the model in *DESIGN_NAME*.LIB.

For more information on instance models, see Reusing instance models.

Starting the Model Editor

To start editing an instance model

1. In the design entry tool, select one part on your schematic page.
2. In Capture, choose *Edit - PSpice Model*. In Design Entry HDL, choose *PSpice Simulator - Edit Model*.

   The schematic page editor searches the model libraries for the instance model. To find out how design entry tools searches the library, see Changing model library search order.

   - If found, the schematic page editor starts the Model Editor, which opens the design library and loads the instance model.
   - If not found, the schematic page editor assumes that this is a new instance model and does the following: makes a copy of the original model definition in the *DESIGN_NAME*.LIB and starts the Model Editor with the new model loaded.

   After you start the Model Editor, you can proceed to change the text as described in To display the model text.

Saving design models

When you save your edits, the following is done for you to make sure the instance model is linked to the selected part instances in your design:

- The Model Editor saves the model definition to *DESIGN_NAME*.LIB.
- If the library is new, the Model Editor configures *DESIGN_NAME*.LIB for local use.

The schematic page editor assigns the new model name to the Implementation property for each of the selected part instances (see What happens if you do not save the instance model).

To save instance models

1. From the File menu, choose Save to update *DESIGN_NAME*.LIB and save it to disk.

@ 2023 All Rights Reserved.

Actions that automatically configure the instance model library for global use

Instance model libraries are normally configured for design use. However, if you perform the following action, the Model Editor configures the library for global use instead:

- Save the model to a different library by typing a new file name in the Library text box in the Save To frame.

To save instance models

1. From the File menu, choose Save to update *DESIGN_NAME*.LIB and save it to disk.

What happens if you do not save the instance model

Before the schematic page editor starts the Model Editor, it does the following:

- Makes a copy of the original model and saves it as an instance model in *SCHEMATIC_NAME*.LIB.
- Configures *SCHEMATIC_NAME*.LIB for design use, if not already done.
- Attaches the new instance model name to the Implementation property for the selected part instance.

This means that if you:

- quit the Model Editor, or
- return to the design entry tool to simulate the design

without first saving the model you are editing, the part instance on your schematic page is still attached to the instance model implementation.

In this case, the instance model is identical to the original model. If you decide to edit this model later, be sure to do one of the following:

- If you want the changes to remain specific to the current design, edit the instance model in the design library using the Model Editor.
- If you want the change to be global, change the model implementation for the part instance in your design back to the original model name in the global library, and then edit the original model from within the part editor.

To find out how to change model references, see Changing the Model Reference to an Existing Model Definition.

@ 2023 All Rights Reserved.

Model Creation Examples

Examples covered in this section cover how to use the Model Editor to create simulation models based on:

- Device characteristic curves. See Example: Creating a PSpice model based on device characteristic curves.
- PSpice-provided templates. See Example: Creating template-based PSpice model on page 75.

Example: Creating a PSpice model based on device characteristic curves

In this example, you will model a simple diode device as follows:

- Create the schematic for a simple half-wave rectifier.
- Create a new model for a diode.
- Attach new model to the D1 diode.

Creating the half-wave rectifier design

Figure 3-4 Design for a half-wave rectifier.

@ 2023 All Rights Reserved.

To create the design in Capture

1. From the Project Manager, from the File menu point to New, then choose Project.
2. In the New Project dialog box, ensure that the Analog or Mixed A/D option is selected.
3. Enter the name of the new project (RECTFR) and click Create.
4. From the Capture Place menu, choose Part.
5. Place one each of the following parts (reference designator shown in parentheses) as shown in the figure.
   - ❑ Dbreak (D1 diode)
   - ❑ C (C1 capacitor)
   - ❑ R (R1 resistor)
   - ❑ VSIN (V1 sine wave source)
6. Click the Ground button on the tool palette and place the ‘0’ analog ground from the SOURCE.OLB part library.
7. From the Place menu, choose Wire, and draw the connections between parts.
8. From the File menu, choose Save.

Note: If you were to simulate this design using a transient analysis, you would also need to set up a transient specification for V1; most likely, this would mean defining the VOFF (offset voltage), VAMPL (amplitude), and FREQ (frequency) properties for V1. For this tutorial, however, you will not perform a simulation, so you can skip this step.

9. Invoke Project Manager.
10. In Project Manager, from the File menu, choose New. The New Project Wizard appears.
11. In the Name text box, enter the name of the project (`RECTFR`).
12. In the Location text box, enter the path to the directory where you want to create the project
13. Click Next. The Project Libraries dialog box appears.
14. Select `ANALOG` from the list of Available Libraries, and click Add. The `ANALOG` library is added to the Project Libraries list.
15. Add the following libraries as described in step 14:
    - ❑ `BREAKOUT`

@ 2023 All Rights Reserved.

- ❑ `source`

16. Click Next. The Design Name dialog box appears.
17. In the Design Name text box, enter the name of the top level design (`RECTFR`).
18. Click Next. The Summary dialog box displays the project details.
19. Click Finish to create the project.

To create the design in Design Entry HDL

Create a new design project:

1. Invoke Project Manager.
2. In Project Manager, choose *File – New*. The New Project Wizard appears.
3. In the Name text box, enter the name of the project (`RECTFR`).
4. In the Location text box, enter the path to the directory where you want to create the project
5. Click *Next*. The Project Libraries dialog box appears.
6. Select `ANALOG` from the list of Available Libraries, and click *Add*. The `ANALOG` library is added to the Project Libraries list.
7. Add the following libraries as described in step 14:
   - ❑ `BREAKOUT`
   - ❑ `source`
8. Click *Next*. The Design Name dialog box appears.
9. In the Design Name text box, enter the name of the top level design (`RECTFR`).
10. Click *Next*. The Summary dialog box displays the project details.
11. Click *Finish* to create the project.

To create the design, invoke the Design Entry HDL schematic editor.

1. In Project Manager, click on the Design Entry icon. Design Entry HDL appears.
2. From the Component menu, choose Add. The Component Browser appears.
3. Place one each of the following parts (reference designator shown in parentheses) as shown in the figure:

@ 2023 All Rights Reserved.

- Dbreak (D1 diode) from the BREAKOUT library
- C (C1 capacitor) from the ANALOG library
- R (R1 resistor) from the ANALOG library
- VSIN (V1 sine wave source) from the SOURCE library
- 0 analog ground from the SOURCE library.

Figure 3-5 Design for a half-wave rectifier.

4. From the Wire menu, choose Draw and draw the connections between parts as shown in the figure.
5. From the File menu, choose Save.

Note: If you were to simulate this design using a transient analysis, you would also need to set up a transient specification for V1; most likely, this would mean defining the VOFF (offset voltage), VAMPL (amplitude), and FREQ (frequency) properties for V1. For this tutorial, however, you will not perform a simulation, so you can skip this step.

Using the Model Editor to edit the D1 diode model

To create a new model and model library

1. Open the Model Editor.
2. From the File menu in the Model Editor, choose New.
3. From the Model menu, choose New.
4. In the New dialog box, do the following:

@ 2023 All Rights Reserved.

a. In the Model Name text box, type `DbreakX`.

b. Select Use Device Characteristic Curves.

c. From the From Model list, select Diode.

d. Click OK.

5. From the File menu, choose Save.

By default, the updated model is saved in the `RECTFR.LIB` library.

Entering data sheet information

As shown in the following figure**,** the Model Editor initially displays:

- diode model characteristics listed in the Models List window, and
- DbreakX model parameter values listed in the Parameters window.

Figure 3-6 Model characteristics and parameter values for DbreakX.

@ 2023 All Rights Reserved.

You can modify each model characteristic shown in the Model Spec frame with new values from data sheets. The Model Editor takes the new information and fits new model parameter values.

When updating the entered data, the Model Editor expects either:

- device curve data (point pairs) or
- single-valued data

depending on the device characteristic.

For the diode, Forward Current, Junction Capacitance, and Reverse Leakage require device curve data. Reverse Breakdown and Reverse Recovery require single-valued data.

The following table lists the data sheet information for the DbreakX model.

Table 3-4 Sample diode data sheet values

For this model characteristic...	Enter this...
forward current	(1.3, 0.2)
junction capacitance	(1m, 120p) (1, 73p) (3.75, 45p)
reverse leakage	(6, 20n)
reverse breakdown	(Vz=7.5, Iz=20m, Zz=5)
reverse recovery	no changes

To change the Forward Current characteristic

1. In the Spec Entry frame, click the Forward Current tab.

   This tab requires curve data.

2. In the Vfwd text box, type `1.3`.
3. Press *Tab* to move to the Ifwd text box, and then type `0.2`.

@ 2023 All Rights Reserved.

To change the values for Junction Capacitance and Reverse Leakage

Follow the same steps as for Forward Current, entering the data sheet information listed in the following table that corresponds to the current model characteristic.

To change the Reverse Breakdown characteristic

1. In the Spec Editing frame, click the Reverse Breakdown tab.

   This tab requires single-valued data.

2. In the Vz text box, type `7.5`.

   Note: The Model Editor accepts the same scale factors normally accepted by PSpice A/D.

3. Press *Tab* to move to the Iz text box, and then type `20m`.

4. Press *Tab* to move to the Zz text box, and then type `5`.

Extracting model parameters

To generate new model parameter values

1. From the Tools menu, choose Extract Parameters.

   A check mark appears in the Active column of the Parameters frame for each extracted model parameter.

To display the curves for the five diode characteristics

1. From the Window menu, choose Tile.

@ 2023 All Rights Reserved.

Some of the plots are shown in the following figure:

Figure 3-7 Assorted device characteristic curves for a diode.

You can also do the following with an active plot window:

- Pan and zoom within the plot using commands on the View menu.
- Rescale axes using the Axis Settings command on the Plot menu.

Adding curves for more than one temperature

By default, the Model Editor computes device curves at 27°C. For any characteristic, you can add curves to the plot at other temperatures.

To add curves for Forward Current at a different temperature

1. In the Spec Entry frame, click the Forward Current tab.
2. From the Plot menu, choose Add Trace.
3. Type `100` (in °C).
4. Click OK.

@ 2023 All Rights Reserved.

The Forward Current plot should appear as shown in the following figure:

Figure 3-8 Forward Current device curve at two temperatures.

Completing the model definition

You can refine the model definition by:

- modifying the entered data as described before, or
- editing model parameters directly.

You can update individual model parameters by editing them in the Parameters frame of the Model Editor workspace. When you save the model library, the Model Editor automatically updates the device curves.

For this tutorial, leave the model parameters at their current settings.

@ 2023 All Rights Reserved.

To save the model definition with the current parameter values and to make the model available to your design

1. From the File menu, select Save to update `RECTFR.LIB` and save the library to disk.

   The model definition is now complete. You can use this model definition in your design.

Attaching the DbreakX model to the D1 diode in Capture

1. In Capture, open the RECTFR project.
2. Select the D1 diode.
3. From the Edit menu, choose Properties.
4. In the Property Editor dialog box, change the value of Implementation property from Dbreak to DbreakX.

   To know more about the Implementation property, see MODEL.
5. Close the dialog box, and save the design.

   Your design is ready to simulate with the model definition you just created.

Example: Creating template-based PSpice model

In this example, you will create a template-based PSpice model for an operational amplifier, using the Model Editor.

The template-based OPAMP model is the only model created using the Model Editor that has multiple level support for simulation parameters. Tasks that will be covered in this example are:

- Creating a new template-based PSpice model for Operational Amplifier
- Multiple level support for template-based OPAMP models in the Model Editor

  Multiple level support implies that the number of simulation parameters used in the model varies with the model level. The higher the level, the more are the number of simulation parameters.

  The models with a higher number of simulation parameters are closer to the real life devices. Therefore, the simulation results are more accurate when high level models are used. Use lower level simulation models to minimize the simulation time.

@ 2023 All Rights Reserved.

Creating a new model

1. Start the Model Editor alone.
2. From the File menu, choose New to create a new library.
3. From the Model menu, choose New.
4. In the New dialog box, specify the name of the new model as `OPA_LOCAL`.
5. Select the Use Templates option.
6. To specify the device type, select Operational Amplifier from the From Model drop-down list.
7. Specify the type of OPAMP to be created. In this example, select options to create an internally compensated bipolar operational amplifier with PNP input.

Figure 3-9 The New dialog box

@ 2023
All Rights Reserved.

8. Click OK.

The Simulation Parameters and the Model Text windows appear. In the Models List, three models, `OPA_LOCAL_1`, `OPA_LOCAL_2`, and `OPA_LOCAL_3` appear.

Caution

The Model Editor creates multiple models only for Advanced Analysis OPAMP models. This is because, template-based OPAMP models support multiple levels of simulation parameters, and the Model Editor creates one model for each level.

The `OPA_LOCAL_3` model contains all the simulation parameters available in the `OPA_LOCAL_2` model plus some extra simulation parameters. The `OPA_LOCAL_2` model contains some additional simulation parameters besides the ones listed in the `OPA_LOCAL_1` model.

9. Select the `OPA_LOCAL_3` model, and edit the values of the simulation parameters listed in the table. The following table lists the simulation parameters along with the new values. For other simulation parameters not listed in the table, accept the default values.

Table 3-5 List of Simulation Parameters to be modified

Simulation Property Name	Value	Distribution	Postive Tolerance (Postol)	Negative Tolerance (Negtol)
VOS	$1e^{-7}$	FLAT	10%	10%
IB	default	FLAT	10%	10%
IBOS	default	FLAT	10%	10%
A0	1000000	FLAT	10%	10%
GBW	default	FLAT	10%	10%
SRP	$1.0e^{+6}$	FLAT	10%	10%
SRM	$1.0e^{+6}$	FLAT	10%	10%
CMRR	default	FLAT	10%	10%

10. To ensure that the values entered by you in the Simulation Parameters frame overwrite the default value of the simulation parameters, check the Editable check box for all the simulation properties.

Selecting the Editable check box ensures that the simulation parameter value:

@ 2023 All Rights Reserved.

- entered by users will overwrite the values in the template property file.
- appears in the device property file.
- can be directly passed from the schematic editor as properties attached to the symbol.

In such cases, the values of the simulation parameters can be picked up from following three locations in the decreasing order of priority:

- schematic editor
- device property file
- template property file

11. After making the simulation parameters editable, save the model. Specify the library name as LOCAL_LIB.

All the changes in the `OPA_LOCAL_3` model are reflected in the `OPA_LOCAL_2` and `OPA_LOCAL_1` models also. This is because all three models have a common section for the simulation and smoke parameters in the device property file, `LOCAL_LIB.PRP`.

Caution

It is recommended that multiple level models created using the Model Editor should be used cautiously especially when used in different designs.

Consider a situation where you use `OPA_LOCAL_1` from the `LOCAL_LIB` library in design A and `OPA_LOCAL_3` from the `LOCAL_LIB` library in design B. Any changes that you make to the simulation or smoke parameter values of the `OPA_LOCAL_1` model for design A will also be reflected in the `OPA_LOCAL_3` model used in design B.

After you have saved your changes in the Model Editor, the following files are generated:

- `LOCAL_LIB.LIB` - The library file containing model information.
- `LOCAL_LIB.PRP` - The device property file containing device specific information for all the models in the `LOCAL_LIB.LIB`.

Saving the `LOCAL_LIB` library completes the tasks of creating a template-based PSpice model for Operational Amplifier. Because of the multiple-level support for Advanced Analysis OPAMP models in the Model Editor, instead of one, three models are created.

Note: Deleting a Model Editor-created multiple-level model deletes the model only from the library. The model information is not deleted from the device property file. This is because, all

@ 2023 All Rights Reserved.

three OPAMP models are linked to the same section in the device property file. The device property section for the model will be deleted only after all the linked models are deleted.

Note: If you have Advanced Analysis installed, the Smoke tab also appears besides the Simulation tab, and you can also add smoke information to the model.

Example covered in the *Creating parts in the batch mode* section, in *Creating Parts for Models* is an extension of this example. It covers part creation using Model Editor and using Model Editor-created parts in a schematic.

@ 2023 All Rights Reserved.

Editing model text

Note: This section is valid only for PSpice models that are based on device characteristic curves.

The Model Text is editable only for PSpice models. For template-based PSpice models, the Model Text window is read-only.

Caution

If you edit the text of a model that was created by entering data sheet values, you may not be able to edit the model in the Extract Model view again.

For any model, you can edit model text in the Model Editor instead of using the Spec Entry and Parameter frames. However, there are two cases where you must edit the model text:

- When you want to edit models of device types not supported by the Model Editor. The model text is displayed automatically when you load one of these models.
- When you want to add DEV and LOT tolerances for Monte Carlo or sensitivity/worst-case analysis.

By typing PSpice commands and netlist entries, you can do the following:

- change definitions, and
- create new definitions

When you have finished, the Model Editor automatically configures the model definitions into the model libraries.

To display the model text

1. From the View menu, choose Edit Model.

   The Model Editor displays the PSpice syntax for model definitions:

   - .MODEL syntax for models defined as parameter sets
   - .SUBCKT syntax for models defined as netlist subcircuits

   You can edit the definition just as you would in any standard text editor.

To find out more about PSpice command and netlist syntax, refer to the *PSpice A/D Reference Guide*.

@ 2023 All Rights Reserved.

Editing .MODEL definitions

For definitions implemented as model parameter sets using the PSpice .MODEL syntax, the Model Editor lists one parameter per line. This makes it easier to add DEV/LOT tolerances to model parameters for Monte Carlo or sensitivity/worst-case analysis.

Editing .SUBCKT definitions

For definitions implemented as subcircuit netlists using the PSpice .SUBCKT syntax, the Model Editor displays the subcircuit syntax exactly as it appears in the model library. The Model Editor also includes all of the comments immediately before or after the subcircuit definition.

Changing the model name

You can change the model name directly in the PSpice .MODEL or .SUBCKT syntax, but double-check that the new name does not conflict with models already contained in the libraries. To find out more about instance model naming conventions, see What is an instance model.

Note: If you do create a model with the same name as another model and want PSpice A/D to always use your model, make sure the configured model libraries are ordered such that your definition precedes any other definitions. To find out more about search order in the model library, see Changing model library search order.

@ 2023 All Rights Reserved.

Example: Editing a Q2N2222 Instance Model

Suppose you have a design named `MY.OPJ` that contains several instances of a Q2N2222 bipolar transistor. If you want to see the effect of base resistance variation on one specific device, Q6, you need to do the following:

- Define a tolerance (in this example, 5%) on the Rb model parameter.

Important

Adding tolerance to a device characteristic curve-based model does not make it compatible for use with Advanced Analysis Monte Carlo.

- Set up and run PSpice Monte Carlo analysis.

The following example demonstrates how to set up the instance model for Q6.

Starting the Model Editor

To start the Model Editor,

1. In the schematic page editor, select Q6 on the schematic page.
2. From the Edit menu, choose PSpice Model.

   The Model Editor automatically creates a copy of the Q2N2222 base model definition.
3. In the Model Editor, from the View menu, choose Model Text.

   The Model Editor displays the PSpice syntax for the copied model in the text editing area.

Editing the Q2N2222 model instance

Text edits appropriate to this example are as follows:

- Add the `DEV 5%` clause to the `Rb` statement (required).
- Change the model name to `Q2N2222-MC` (optional, for descriptive purposes only).

To find out more about PSpice command and netlist syntax, refer to the *PSpice A/D Reference Guide*.

Saving the edits and updating the schematic

When you choose Save from the File menu, two things happen:

@ 2023 All Rights Reserved.

- The Model Editor saves the model definition to the model library.
- The schematic page editor updates the Implementation property value to Q2N2222-MC for the Q6 part instance.

In this example, the default model library is `MY.LIB`. If `MY.LIB` does not already exist, the Model Editor creates and saves it in the current working directory. The schematic page editor then automatically configures it as a design model library for use with the current design only.

Now you are ready to set up and run the Monte Carlo analysis.

Note: If you verify the model library configuration (in the Simulation Settings dialog box, click the Configuration Files tab and view the Library files list), you see entries for NOM.LIB (for global use, as denoted by the icon) and MY.LIB (for design use, as denoted by the icon) in the Library files list.
You can change the model reference for this part back to the original Q2N2222 by the procedure explained in To change model references for part instances on your design section.

Using the Create Subcircuit Format Netlist Command (Capture only)

The Create Subcircuit Format Netlist command is used from Capture. This command creates a subcircuit netlist definition for the displayed level of hierarchy and all lower levels in your design.

Note: The Create Subcircuit Format Netlist command does not help you create a hierarchical design. You need to do this yourself before using the Create Subcircuit Format Netlist command. For information on hierarchical designs and how to create them, refer to the *OrCAD X Capture User Guide*.

The schematic page editor does the following things for you:

- Maps any named interface ports at the active level of hierarchy to terminal nodes in the PSpice .SUBCKT statement.
- Saves the subcircuit definition to a file named *DESIGN_NAME-SCHEMATIC_NAME*.LIB.

Before you can use the subcircuit definition in your design, you need to:

- Create a part for the subcircuit.
- Configure the *DESIGN_NAME-SCHEMATIC_NAME*.LIB file so PSpice A/D knows where to find it.

@ 2023 All Rights Reserved.

To create a subcircuit definition for a portion of your design

To create a subcircuit netlist definition

1. In the Project Manager, select the schematic folder that contains the circuitry for which a subcircuit netlist definition is to be created.
2. If the schematic folder is not the root folder, choose Make Root from the Design menu. You may be prompted to save the design first.
3. In the Project Manager, from the Tools menu, choose Create Netlist.
4. Select the PSpice tab.
5. In the Options frame, select Create SubCircuit Format Netlist.
6. Click OK to generate the subcircuit definition and save it to *DESIGN_NAME-SCHEMATIC_NAME*.LIB.

To create a part for the subcircuit netlist definition

1. Open the Model Editor alone.
2. From the File menu, select the Open command and open the DESIGN_NAME-SCHEMATIC_NAME.LIB file.
3. Select the model from the Models List and, if necessary, refine the subcircuit definition.

   Refinements can include extending the subcircuit definition using the optional nodes construct, OPTIONAL:, the variable parameters construct, PARAMS:, and the .FUNC and local .PARAM commands.
4. From the File menu, select Create Capture Parts.
5. In the Enter Input Model Library text box, browse and open the DESIGN_NAME-SCHEMATIC_NAME.LIB file.

   Note: The Enter Output Part Library is automatically filled in with the DESIGN_NAME-SCHEMATIC_NAME.OLB file.
6. Click OK and OK again to clear the .ERR log window.

   The subcircuit part is now ready for use in a design.

To configure the subcircuit file

1. In the schematic page editor, from the PSpice menu, choose Edit Profile to display the Simulation Settings dialog box.
2. Click the Configuration Files tab.

@ 2023 All Rights Reserved.

3. Click either Library or Include in the Category field of the Configuration Files tab, and then configure *DESIGN_NAME-SCHEMATIC_NAME*.LIB as either a model library or an include file (see Configuring Model Libraries).

Changing the Model Reference to an Existing Model Definition

Parts are linked to models by the model name assigned to the parts' Implementation property. You can change this assignment by replacing the Implementation property value with the name of a different model that already exists in the library.

You can do this for:

- A part instance in your design.
- A part in the part library.

To change model references for part instances on your design

1. Find the name of the model that you want to use.
2. In the schematic page editor, select one or more parts on your schematic page.
3. From the Edit menu, choose Properties.

   The Parts spreadsheet appears.
4. Click the cell under the column Implementation Type.
5. From the Implementation list, select PSpice Model.
6. In the Implementation column, type the name of the existing model that you want to use if it is not already listed.
7. Click Apply to update the changes, then close the spreadsheet.

To change the model reference for a part in the part library

1. Find the name of the model that you want to use.
2. In the schematic page editor, select the part you want to change.
3. From the Edit menu, choose Part to load the part in the part editor for editing.

@ 2023 All Rights Reserved.

4. From the Options menu, choose Part Properties to display the User Properties dialog box.
5. Select Implementation Type.
6. From the Implementation list, select PSpice Model.
7. In the Implementation text box, type the name of the existing model that you want to use if it is not already listed.
8. Click OK to close the Edit Part dialog box.

Reusing instance models

If you created instance models in your design and want to reuse them, there are two things you can do:

- Attach the instance model implementation to other part instances in the same design.
- Change the instance model to a global model and create a part that corresponds to it.

For information on how to create instance models, see:

- Running the Model Editor from the schematic editor.
- Starting the Model Editor.

Reusing instance models in the same schematic

There are two ways to use the instance model elsewhere in the same design.

To use the instance model elsewhere in your design

Do one of the following:

- Change the model reference for other part instances to the name of the new model instance.

  See Changing the Model Reference to an Existing Model Definition.
- From the Edit menu, use the Copy and Paste commands to place more part instances.

@ 2023 All Rights Reserved.

Making instance models available to all designs

If you are refining model behavior specific to your design, and are ready to make it available to any design, then you need to link the model definition to a part and configure it for global use.

To make your instance model available to any design

1. Create a part and assign the instance model name to the Implementation property.
2. If needed, move the instance model definition to an appropriate model library, and make sure the library is configured for global use. See Configuring Model Libraries for more information.

Note: If you use the part wizard to create the part automatically from the model definition, then this step is completed for you.

@ 2023 All Rights Reserved.

Configuring Model Libraries

Although model libraries are usually configured for you, there are things that you sometimes must do yourself. These are:

- adding new model libraries that were created outside of the design entry tools or the Model Editor
- changing the global, design or profile scope of a model library
- changing the library search order
- changing or adding directory search paths

Note: It is recommended that any library that contains simulation models used in a design should be configured. This is to ensure that during netlisting, PSpice A/D is able to locate corresponding simulation models for all the parts used in a design.

The Configuration Files Tab

The *Configuration Files* tab of the *Simulation Settings* dialog box lets you add, change, and remove model libraries and include files from the configuration or modify the search order.

@ 2023
All Rights Reserved.

Note: Removing a library in this dialog box means that you are removing the model library from the configured list. The library still exists on your computer and you can add it back to the configuration later.

To display the Library files list

1. In PSpice A/D, open or create a PSpice project.
2. From the PSpice menu, choose either New Simulation Profile or Edit Profile if a profile already exists.
3. Click the Configuration Files tab in the Simulation Settings dialog box.
4. Click Library in the Category field to display the Library files list.

   The Library files list shows the model libraries that PSpice A/D searches for definitions matching the parts in your design. Files showing an icon after their name have global scope; files having the icon have a design scope and files with the icon have a profile scope.

   The buttons for adding model libraries to the configuration follow the same profile/local/global syntax convention. Click one of the following:

   - Add to Profile for profile models
   - Add to Design for design models
   - Add as Global for global models

Note: Editing models that are configured at the

- Profile or Global level creates a copy of the model under the simulation folder and the project folder respectively. Changes made to the model are saved in the copied model.
- Design level points to the original model. Changes made to the model are saved in the original model itself.

The Include files list in the Configuration Files tab contains include files. You can manually add profile, design and global include files to your configuration using the Add to Profile, Add to Design and Add as Global buttons, respectively.

The Stimulus files list in the Configuration Files tab contains stimulus files.

@ 2023 All Rights Reserved.

How PSpice A/D uses model libraries

PSpice A/D searches libraries for any information it needs to complete the definition of a part or to run a simulation. If an up-to-date index does not already exist, PSpice A/D automatically generates an index file and uses the index to access only the model definitions relevant to the simulation. This means:

- Disk space is not used up with definitions that your design does not use.
- There is no memory penalty for having large model libraries.
- Loading time is kept to a minimum.

Tip

If you refer to a library file from a location where you do not have write permissions, PSpice A/D might not be able to open or make an index file. To resolve this create a `.lib` file at a location you have permissions and include the original file in this `.lib` file, then add this file to the Configuration Files tab of the Simulation Settings dialog box. For example, if you use a file `original.lib` located at `\\server\Cadence\lib` and do not have permissions on `\\server\Cadence\lib`:

a. Create a file, say `my.lib`, at a location where you have permission

b. Edit `my.lib` to include the line: `.lib \\server\Cadence\lib\original.lib`

c. Add the location of `my.lib` in the Configuration Files tab of Simulation Settings dialog box.

Caution—When you use include files instead

PSpice A/D treats model library and include files differently as follows:

- For model library files, PSpice A/D reads in only the definitions it needs to run the current simulation.
- For include files, PSpice A/D reads in the file in its entirety.

This implies that if you configure a model library (*.LIB extension) as an include file using the Add to Design or Add as Global button, PSpice A/D loads every model definition contained in that file.

If the model library is large, you might overload the memory capacity of your system. However, when developing models, you can do the following:

@ 2023 All Rights Reserved.

1. Initially configure the model library as an include file; this avoids rebuilding the index files every time the model library changes.
2. When your models are stable, reconfigure the include file containing the model definitions as a library file.

To reconfigure an include file as a model library file:

1. From the Simulation menu, choose Edit Profile, then click the Configuration Files tab.
2. Click Include in the Category field to display the Include files list.
3. Select the include file that you want to change.
4. Click the Delete button located above the Include files list.

See Adding model libraries to the configuration.

Search order

When searching for model definitions, PSpice A/D scans the model libraries using these criteria:

- profile model libraries before design model libraries
- design model libraries before global model libraries
- model library sequence as listed in the Library files list in the Configuration Files tab of the Simulation Settings dialog box
- list of directories (including the design library) specified in the library search path in the order given (see Changing the library search path). From the directories listed in the Library Path, only the directories that contain a `NOM.LIB` file are looked up by PSpice A/D during the model search.

Tip

The search order for the device property files is same as that of the model library files. The .PRP files are searched along with the .LIB files.

Handling duplicate model names

If two or more model libraries contain models with the same name, PSpice A/D always uses the first model it finds. This means you might need to modify the search order to make sure PSpice A/D uses the model that you want. See Changing model library search order.

@ 2023 All Rights Reserved.

Note: PSpice A/D searches profile libraries before design libraries and design libraries before global libraries. Therefore, if the new model you want to use is specific to your profile and the duplicate definition is design or global, you do not need to make any changes. Similarly, if the new model you want to use is specific to your design and the duplicate definition is global, you do not need to make any changes.

Adding model libraries to the configuration

New libraries are added above the selected library name in the Library files list box.

To add model libraries to the configuration

1. From the Simulation menu, choose Edit Profile, then click the Configuration Files tab.
2. Click Library in the Category field to display the Library files list.
3. Click the library name positioned one entry below where you want to add the new library.
4. In the Filename text box, either:
   - ❑ type the name of the model library, or
   - ❑ click Browse to locate and select the library.
5. Do one of the following:
   - ❑ If the model definitions are for use in the current profile only, click the Add to Profile button.
   - ❑ If the model definitions are for use in the current design only, click the Add to Design button.
   - ❑ If the model definitions are for global use in any schematic, click the Add as Global button instead.
6. Click OK.

Note: If the model libraries reside in a directory that is not on the library search path, and you use the Browse button in step 4 to select the libraries you want to add, then the schematic editor automatically updates the library search path to include the selected library in the search path. Otherwise, you need to add the directory path yourself. See Changing the library search path.

@ 2023 All Rights Reserved.

Changing the Model Library Scope

Changing from profile to design, profile to global, design to global and vice versa

There are times when you might need to change the scope of a model library from profile to design, profile to global, design to global, or vice versa.

Example: If you have an instance model that you now want to make available to any design, then you need to set the scope of the local model library that contains it to global.

For more information, see Global vs. Design vs. Profile Models and Libraries.

To change the scope of a design model to global

1. From the Simulation menu, choose Edit Profile, then click the Configuration Files tab.
2. Click Library in the Category field to display the Library files list.
3. Select the model library that you want to change.
4. Click the Delete toolbar button to remove the local entry.
5. Add the model library as a global entry.

   For more information, see Adding model libraries to the configuration.

@ 2023 All Rights Reserved.

Changing model library search order

Two reasons why you might want to change the search order are to:

- reduce the search time
- avoid using the wrong model when there are model names duplicated across libraries. PSpice A/D always uses the first instance. See Handling duplicate model names for more information.

To change the order of libraries

1. Click the Configuration Files tab in the Simulation Settings dialog box.
2. Click Library in the Category field to display the Library files list. On the Library files list of the Configuration Files tab:
   a. Select the library name you wish to move.
   b. Use either the Up Arrow or Down Arrow toolbar button to move the library name to a different place in the list.

   Note: You can only change the order of libraries that have the same scope. This implies that though you can change the order of profile libraries, local libraries and global libraries, you cannot place a global library before a local library or a local library before a profile library.
3. If you have listed multiple `*.LIB` commands within a single library (like `NOM.LIB`), then edit the library using a text editor to change the order.

   Example: The model libraries `DIODES.LIB` and `EDIODES.LIB` (European manufactured diodes) shipped with your PSpice A/D programs have identically named device definitions. If your design uses a device out of one of these libraries, you need to position the model library containing the definition of choice earlier in the list. If your system is configured as originally shipped, this means you need to add the specific library to the list *before* `NOM.LIB`.

Caution

Do not edit NOM.LIB. If you do, PSpice A/D will recreate the indexes for every model library referenced in NOM.LIB. This can take some time.

4. After you have modified the library settings using the Simulation Settings dialog box, you must first select the design name in the Project Manager and then save the design by clicking the save button. This must be done every time you make a change in the Simulation Settings dialog box. This is to ensure that the changes in the simulation

@ 2023 All Rights Reserved.

settings are reflected in the `.OPJ` file and are picked up the netlister in the subsequent flows.

Order of Access and Precedence for Models used in Simulation

You can configure model libraries at simulation profile level, design-level, and custom-global-level (using the Simulation Setting dialog box).

Order of Access

You can access the models that are used for simulation in the following order:

- Models from libraries added to profile
- Models from libraries added to design
- Models from globally configured libraries

Order of Precedence

In the Simulation Settings dialog box, by selecting the check box *Prioritize models for same .lib for all circuits*, you can follow a particular order of precedence in the design.

If you select this check box, the following precedence for models takes place:

1. Models referred directly on schematic or design: If there is a model with multiple definitions in the configured libraries, then models are accessed as per the order explained above.

@ 2023
All Rights Reserved.

2. Model referred by a subcircuit model definition: For the models that are referred to from a specific subcircuit model definition, highest precedence is assigned to the model definition found in same library file. If models or subcircuits are not found in the current library, then they are searched from other libraries as per the order of access explained above.

Example

If a subcircuit, TOP_B in a library file LIBRARY_B refers to another model/subcircuit, then this model/subcircuit will first be searched in the current library where TOP_B is defined, that is LIBRARY_B.

On a schematic, if you have placed the following parts:

- Subcircuit, **`TOP_B`**
  - Which refers to subcircuit, `SUB_TOP_B`
    - Where `SUB_TOP_B` refers to model, `MOSFET_M`
- Model, **`DIODE_M`**

The order in which libraries are indexed is as follows:

1. `LIBRARY_A`, has the following definition:
   - `SUB_TOP_B` (subcircuit)
   - **`DIODE_M`** (model)
   - `MOSFET_M` (model)
2. `LIBRARY_B`, which has the following definition:
   - **`TOP_B`** (subcircuit)
   - `SUB_TOP_B` (subcircuit)
   - **`DIODE_M`** (model)

Now on simulation, the models are picked in the following manner:

- **`TOP_B`** gets picked from `LIBRARY_B`
  - `SUB_TOP_B` gets picked from `LIBRARY_B`
    - `MOSFET_M` gets picked from `LIBRARY_A`
- **`DIODE_M`** gets picked from `LIBRARY_A`

@ 2023 All Rights Reserved.

The same behavior is applicable for parameter statements defined in the library file as shown in the following example.

Example

You have configured two model library files as follows:

```
LibraryA.lib
    .PARAM FCTR=1
    .SUBCKT MULT_1 IN1 IN2 OUT1 OUT2
     E_E1 OUT1 OUT2 VALUE { V(IN1, IN2)*{FCTR} }
    .ENDS
LibraryB.lib
    .PARAM FCTR=2
    .SUBCKT MULT_2 IN1 IN2 OUT1 OUT2
     E_E1 OUT1 OUT2 VALUE { V(IN1, IN2)*{FCTR} }
.ENDS
```

When the check box, *Prioritize models for same .lib for all circuits* is not selected, then:

On simulation, the value of parameter `FCTR` is taken as `1` by both the subcircuits, `MULT_1` and `MULT_2`. This is because `LibraryA.lib` is read first, which has the value of `FCTR=1`.

When the check box, *Prioritize models for same .lib for all circuits* is selected, then:

On simulation, the parameter value is picked from the same library where the model definition exists, that is:

- If subcircuit `MULT_1` is used in the design, then, on simulation, it will pick `FCTR=1` irrespective of the configured library order.
- If subcircuit `MULT_2` is used in the design, then, on simulation, it will pick `FCTR=2` irrespective of the configured library order.

@ 2023 All Rights Reserved.

Changing the library search path

For model libraries that are configured without explicit path names, PSpice A/D first searches the directory where the current design resides, then steps down the list of directories specified in the Library Path text box on the Library files list in the Configuration Files tab of the Simulation Settings dialog box.

To change the library search path

1. From the Simulation menu, choose Edit Profile to display the Simulation Settings dialog box.
2. Click the Configuration Files tab.
3. Click Library in the Category field to display the Library files list.
4. In the Library Path text box, position the pointer after the directory path that PSpice A/D should search before the new path.
5. Type in the new path name following these rules:
   - Use a semicolon character (;) to separate two path names.
   - Do not follow the last path name with a semicolon.

   Example: To search first `C:\ORCAD\PSPICE\LIBRARY`, then `C:\MYLIBS`, for model libraries, type

   ```
   "C:\ORCAD\PSPICE\LIBRARY";
   "C:\MYLIBS"
   ```

   in the Library Path text box.

@ 2023 All Rights Reserved.

Handling Smoke Information Using the Model Editor

You can use the Model Editor to add smoke information to a device type supported by the Model Editor. This feature is available only if you have PSpice Advanced Analysis installed.

Using the Model Editor, you can:

- Add smoke information PSpice models.
- Create template-base PSpice models with smoke information.
- Edit smoke information for the device types supported by the Model Editor.

Adding smoke information

If you have Advanced Analysis installed, an extra tab, the Smoke tab, appears next to the Simulation tab. Adding smoke information involves specifying the maximum operating conditions for different smoke parameters. Smoke parameters are the tests that are predefined in the device template file. These tests are performed between two different nodes of the device. The node to port mapping information for the device is available in the Test Node Mapping frame.

The information in the Test Node Mapping frame cannot be edited for template-based models. For models based on device characteristic curves, this information is editable. The Test Node Mapping information is not editable for template-based models. This is to avoid risk of smoke information getting out of sync with the smoke test.

See Smoke Parameters

Adding smoke information to PSpice models

Adding smoke information to PSpice models enable you to use the models for Advanced Analysis smoke run. Using the Model Editor, you can add smoke information only to the device types supported by the Model Editor.

For template-based models, smoke information is present by default and is visible in the Smoke Parameters frame. You can edit this information using the Model Editor. To add smoke information to a non template-based model, you need to complete the following steps.

1. From the Models List window, select the model to which the smoke information is to be added.

   Note: If the model that you are trying to edit, has multiple implementations, you first need to select the implementation for which changes have to be made, from the Select

@ 2023 All Rights Reserved.

Implementation dialog box. This dialog box lists all the implementations associated with a model. The changes are saved to the location where original implementation is stored.

2. From the Model menu choose Add Smoke.

- If the model uses the.SUBCKT definition, the Add Smoke dialog box appears.

  a. In the Add Smoke dialog box, specify the device type.

  b. Click OK.

  The Test Node Mapping and the Smoke Parameters frames appear along with the Model Text window that displays the model definition in the text format.

- If the Model uses the .MODEL definition, the Test Node Mapping and the Smoke Parameters frames appear along with the Model Text window.

In the Test Node Mapping frame, enter the name of the port that maps to each Node. The correct node to port mapping is essential to ensure that smoke analysis gives correct results. To know more about Test Node Mapping, see *Smoke Parameters*.

For the PSpice models with the .MODEL definition, node names should be the ones that get assigned by part created using the Model Editor. For the PSpice models with .SUBCKT definition, node names are defined by the ports in the subcircuit definition.

3. Enter the maximum operating values for the parameters in the Smoke Parameters frame.

4. Save the model.

When you save the model, the `.LIB` and `.PRP` files are updated.

Note: When you use the Model Editor to create device characteristic curves-based PSpice models, the smoke information is not available by default. After you add the smoke information, a new file, `LIB_NAME.PRP` is created.

Creating template-based PSpice models with smoke information

The steps for creating template-based models with smoke information are exactly the same as the steps for creating template-based simulation models. If you have Advanced Analysis installed, the Smoke tab appears by default. You are not required to select Add Smoke from the Model menu.

See Example: Creating template-based PSpice model

@ 2023 All Rights Reserved.

Using the Model Editor to edit smoke information

Open a model with smoke information in the Model Editor. You can modify the maximum operating values for different smoke parameters in the Smoke Parameters frame. For models based on device characteristic curves, you can also change the port to node mapping in the Test Node Mapping window. Saving the model will update the .PRP file with the latest information.

@ 2023 All Rights Reserved.

Examples: Smoke

Adding smoke information to the D1 diode model

In this example, you will use the Model Editor to add smoke information to the D1 diode model used in the half-wave rectifier design explained previously in Example: Creating a PSpice model based on device characteristic curves.

To add smoke information, complete the following steps.

1. From the Model menu, choose Add Smoke.

   The Test Node Mapping and Smoke Parameters frames appear.

   Note: The Add Smoke command is enabled only if you have Advanced Analysis installed.

2. In the Test Node Mapping frame, add the following information.

Node	Port
TERM_AN	1
NODE_AN	1
NODE_CAT	2

   You can get the port names by opening the symbol in the design entry tool.

   a. Select the part in the design entry tool.

   b. From the Edit menu, choose Part.

   c. The symbol view of the part displays. Double-click a pin.

@ 2023 All Rights Reserved.

d. The Name field in the Pin Properties dialog box displays the port name.

3. In the Smoke Parameters frame, add the following smoke information.

Property Name	Value
IF	**1**
VR	**30**
PDM	**1.5**
TJ	**175**
RJC	**50**
RCA	**50**

Note: Smoke information is available in the data sheets provided by the device vendor.

4. To save the changes to the diode model, choose Save from the File menu.

@ 2023 All Rights Reserved.

Once you have modified the D1 diode by adding smoke information, you can run Smoke analysis on your circuit.

Adding smoke information to the OPA_LOCAL operational amplifier model

In this example, you will add smoke information to an OPAMP model created using the Model Editor.

See Example: Creating template-based PSpice model

You can use the Model Editor to add/edit the smoke information only if you have Advanced Analysis installed.

The steps in this design example assume that you have Advanced Analysis installed.

1. Open the Model Editor.
2. Open the LOCAL_LIB library.
3. Select the OPA_LOCAL_3.

   Besides the Simulation Parameters and the Model Text window, a Smoke tab is also displayed.
4. Select the Smoke tab.

   The Test Node Mapping and Smoke Parameters frames are displayed.
5. In the Smoke Parameters frame, specify the maximum operating conditions for the operational amplifier to match the values shown in the following table.

Table 3-6 Modified value of smoke parameters

Device Max Ops	Value
IPLUS	.05
IMINUS	.05
IOUT	.04
VDIFF	32
VSMAX	32
VSMIN	-.3

@ 2023 All Rights Reserved.

Table 3-6 Modified value of smoke parameters

Device Max Ops	Value
VPMAX	.3
VPMIN	.3
VMMAX	-1.5
VMMIN	.3

6. Save the model.

   The LOCAL_LIB.PRP file will be updated with the smoke information. The smoke parameters added in this example are valid for all levels. Even if you delete OPA_LOCAL_3 from the LOCAL_LIB library, the LOCAL_LIB.PRP will not be deleted because it contains smoke information for OPA_LOCAL_1 and OPA_LOCAL_2.

Smoke Parameters

Using the Model Editor, you can add smoke information to the device types supported by the Model Editor.

When you add smoke information to a device, the following two frames are displayed in the Model Editor.

- Test Node Mapping

  In this section you specify the port name that must map to the predefined node names. The predefined terminal names starting with TERM indicate current nodes, and those starting with NODE indicate voltage nodes.

- Smoke Parameters

  In this section, you specify the maximum operating conditions in terms of the values of the smoke parameters.

The thermal parameters that are common to all device types supported by the Model Editor are listed below along with their description.

PDM	Maximum power dissipation
TJ	Maximum junction temperature

@ 2023 All Rights Reserved.

RJC	Maximum thermal resistance (Junction to case thermal resistance)
RCA	Maximum thermal resistance (Case to ambient thermal resistance)

Besides the parameters listed above, there are smoke parameters specific to a particular class of devices. An explanation of device-specific smoke parameters and node names for the Model Editor-supported device types is given below.

Diode

Test Node Mapping

Node	Port name
TERM_AN	Forward current terminal
NODE_AN	Anode voltage node
NODE_CAT	Cathode voltage node

Smoke Parameters

Smoke parameter	Maximum operating condition
IF	Maximum Forward Current (Current through Anode)

@ 2023 All Rights Reserved.

VR	Peak Reverse Voltage (Maximum voltage between Cathode and Anode)

Bipolar Junction Transistors

Test Node Mapping

Node	Port name
TERM_IC	Collector current terminal
TERM_IB	Base current terminal
NODE_VC	Collector voltage node
NODE_VB	Base voltage node
NODE_VE	Emitter voltage node

@ 2023 All Rights Reserved.

Smoke parameters

Smoke parameter	Maximum operating condition
IB	Maximum base current
IC	Maximum collector current
VCB	Maximum collector-base voltage (Maximum voltage difference between Collector and Base)
VCE	Maximum collector-emitter voltage (Maximum voltage difference between Collector and Emitter)
VEB	Maximum emitter-base voltage (Maximum voltage difference between Emitter and Base)
SBSLP	Secondary breakdown slope **Note:** Secondary breakdown is the voltage breakdown between VC and VE at maximum collector current.
SBINT	Secondary breakdown intercept
SBTSLP	Secondary breakdown temperature derating slope
SBMIN	Secondary breakdown derate percentage at maximum junction temperature

Magnetic Core

These are a special type of template-based PSpice models that do not have smoke information. Therefore, the Node Mapping and Smoke Parameters frames are not available for these models.

@ 2023 All Rights Reserved.

Ins Gate Bipolar Transistor (IGBT)

Test Node Mapping

Node	Port name
TERM_IC	Collector current terminal
TERM_IG	Gate current terminal
NODE_VC	Collector voltage node
NODE_VG	Gate voltage node
NODE_VS	Source voltage node

Smoke parameters

Smoke parameter	Maximum operating condition
IG	Maximum gate current

@ 2023 All Rights Reserved.

IC	Maximum collector current
VCG	Maximum collector-gate voltage (Voltage between Collector and Gate)
VCE	Maximum collector-emitter voltage (Voltage between Collector and Emitter)
VGEF	Maximum forward gate-emitter voltage (Voltage between Gate and Emitter)
VGER	Maximum reverse gate-emitter voltage

@ 2023 All Rights Reserved.

Junction FET

Test Node Mapping

Node	Port name
TERM_ID	Depletion current terminal
TERM_IG	Gate current terminal
NODE_VD	Depletion voltage node
NODE_VG	Gate voltage node
NODE_VS	Source voltage node

@ 2023 All Rights Reserved.

Smoke parameters

Smoke parameter	Maximum operating condition
ID	Maximum drain current
IG	Maximum forward gate current
VDG	Maximum drain-gate voltage (Maximum voltage between drain and gate)
VDS	Maximum drain-source voltage (Maximum voltage between drain and source)
VGS	Maximum gate-source voltage (Maximum voltage between gate and source)

@ 2023 All Rights Reserved.

Operational Amplifier

Test Node Mapping

Node	Port name
NODE_POS	Positive voltage source node (POS)
NODE_NEG	Negative voltage source node (NEG)
NODE_VCC	Positive voltage source node
NODE_VEE	Negative voltage source node
NODE_GND	
TERM_POS	Positive current terminal (POS)
TERM_NEG	Negative current terminal (NEG)
TERM_OUT	Output current terminal (OUT)

@ 2023 All Rights Reserved.

Smoke parameters

Smoke parameter	Maximum operating condition	Stands for...
IPLUS	Maximum input current (+)	Maximum value of current at the POS terminal
IMINUS	Maximum input current (-)	Maximum value of current at the NEG terminal
IOUT	Maximum output current	Maximum value of current at the OUT terminal
VDIFF	Maximum differential VIN	Absolute voltage difference between POS and NEG
VSMAX	Maximum supply voltage difference	Voltage difference between NODE_VCC and NODE_VEE
VSMIN	Minimum supply voltage difference	Voltage difference between NODE_VEE and NODE_VCC
VPMAX	Maximum input voltage (+)	Voltage difference between NODE_POS and NODE_VCC
VPMIN	Minimum input voltage (+)	Voltage difference between NODE_VEE and NODE_POS
VMMAX	Maximum input voltage (-)	Voltage difference between NODE_NEG and NODE_VCC
VMMIN	Minimum input voltage (-)	Voltage difference between NODE_VEE and NODE_NEG

@ 2023 All Rights Reserved.

MOSFET

Test Node Mapping

Node	Port name
TERM_ID	Drain current terminal
TERM_IG	Gate current terminal
NODE_VD	Drain voltage node
NODE_VG	Gate voltage node
NODE_VS	Source voltage node

Smoke parameters

Smoke Parameter	Maximum Operating Condition
IG	Maximum forward gate current
ID	Maximum drain current

@ 2023 All Rights Reserved.

VDG	Maximum drain-gate voltage (Maximum difference between NODE_VD and NODE_VG)
VDS	Maximum drain-source voltage (Maximum difference between NODE_VD and NODE_VS)
VGSF	Maximum forward gate-source voltage
VGSR	Maximum reverse gate-source voltage

@ 2023 All Rights Reserved.

Voltage Regulator

Test Node Mapping

Node	Port name
NODE_IN	Input voltage node (IN)
NODE_OUT	Output voltage node (OUT)
NODE_GND	Ground voltage node (GND)

@ 2023
All Rights Reserved.

Darlington Transistor

Test Node Mapping

Node	Port name
TERM_IC	Collector current terminal
TERM_IB	Base current terminal
NODE_VC	Collector voltage node
NODE_VB	Base voltage node
NODE_VE	Emitter voltage node

Smoke parameters

Smoke parameter	Maximum operating condition

@ 2023 All Rights Reserved.

IB	Maximum base current
	Max(TERM_IB)
IC	Maximum Collector current
	Max(TERM_IC)
VCB	Maximum collector-base voltage
	Max(NODE_VC -NODE_VB)
VCE	Maximum collector-emitter voltage
	Max(NODE_VC -NODE_VE)
VEB	Maximum emitter-base voltage
	Max(NODE_VE -NODE_VB)
SBSLP	Secondary breakdown slope
	Note: Secondary breakdown is the voltage breakdown between VC and VE at maximum collector current.
SBINT	Secondary breakdown intercept
SBTSLP	Secondary breakdown temperature derating slope
SBMIN	Secondary breakdown derate percentage at maximum junction temperature

@ 2023 All Rights Reserved.

@ 2023 All Rights Reserved.

4

Creating Parts for Models

A part used for simulation has the following special characteristics:

- a link to a simulation model
- a netlist translation
- modeled pins
- other simulation properties specific to the part, which can include hidden pin connections or propagation delay level (for digital parts)

Ways to Create Parts for Models

If you want to...	Then do this...	To find out more, see this...
■ **Create parts for a set** of vendor or user-defined models saved in a model library. ■ **Change the graphic standard** for an existing model library.	Use the Model Editor to create parts from a model library.	Basing New Parts on a Custom Set of Parts on page 142

@ 2023 All Rights Reserved.

If you want to...	Then do this...	To find out more, see this...
■ **Automatically create one part each time** you extract a new model.	Use the Model Editor and enable automatic creation of parts.	Using the Model Editor to Create Parts on page 125. For a list of device types that the Model Editor supports, see Running the Model Editor Alone on page 51. If the Model Editor does not support the device type for the model definition that you want to create, then you can use a standard text editor to create a model definition using the PSpice .MODEL and .SUBCKT command syntax. Remember to configure the new model library (see Configuring Model Libraries on page 88). Basing New Parts on a Custom Set of Parts on page 142
■ **Create parts for all** the models saved in a model library.		Creating Design Entry Tool parts for all Models in a Library on page 126
❑ in batch mode	Use the Export to Capture Part Library or Export to Design Entry HDL Part Library commands available in Model Editor to create parts from a model library.	
❑ in interactive mode	Use the Model Import Wizard [design entry tool to create parts from a model library.Use the Model Import Wizard [design entry tool] to create parts from a model library.	

@ 2023 All Rights Reserved.

Preparing Models for Part Creation

If you already have model definitions and want to create parts for them, you should organize the definitions into libraries containing similar device types.

To set up a model library for part creation

1. If all of your models are in one file and you wish to keep them that way, rename the file to:
   - ❑ Reflect the kinds of models contained in the file.
   - ❑ Have the `.LIB` extension.

     Note: Model libraries typically have a .LIB extension. However, you can use a different file extension as long as the file format conforms to the standard model library file format.
2. If each model is in its own file, and you want to concatenate them into one file, use the DOS copy command.

   Example:

   You can append a set of files with `.MOD` extensions into a single `.LIB` file using the DOS command:

   ```
   copy *.MOD MYLIB.LIB
   ```
3. Make sure the model names in your new library do not conflict with model names in any other model library.

   For information on managing model libraries, including the search order PSpice A/D uses, see Configuring Model Libraries on page 88.

Starting the Model Editor

To start the Model Editor alone

1. From the Windows Start menu, point to the installed Cadence release, then choose PSpice Accessories, Model Editor.
2. From the File menu, choose Open or New, and enter an existing or new model library name.
3. In the Models List frame, select the name of a model to display it for editing in the Spec Entry frame.

@ 2023 All Rights Reserved.

To start the Model Editor from within design entry tools

1. In the schematic page editor, select the part whose model you want to edit.
2. In Capture, from the Edit menu, choose *PSpice Model*. In Design Entry HDL, choose *PSpice Simulator - Edit Model*.

   The Model Editor starts with the model loaded for editing.

If you have already started the Model Editor from design entry tools, and want to continue working on new models, then:

1. Close the opened model library.
2. Open a new model library.
3. Load a device model or create a new one.

Note: Part creation is disabled, when you launch the Model Editor from design entry tools.

@ 2023 All Rights Reserved.

Using the Model Editor to Create Parts

If you want to create new parts that are not tied to a local design, open the Model Editor alone. Using Model Editor, you can create parts in two modes:

- Batch mode
- Interactive mode

Batch mode of part creation

In this method, you use the *Export to Capture Part Library* or *Export to Design Entry HDL Part Library* command from the File menu, to create part symbols for all models in a simulation library. In this approach, you can view the symbols only after all the changes have been done and saved in the `.olb` file. You need to open the `.olb` file in Capture and view the symbols.

To know more about how to use the *Export to Capture Part Library* or *Export to Design Entry HDL Part Library* command for part creation see Using Batch Mode on page 126.

Interactive Mode of Part Creation

In this mode, you can view the part symbol being attached to a simulation model before the changes are saved in the symbol library. You can also update and attach symbols of your own choice to the models in the library. To create Capture or Design Entry HDL symbols in an interactive mode, choose *Model Import Wizard [design entry tool]* command, from the File menu.

When you generate part symbols using Model Import wizard, you can review the symbol shapes before you save the changes to the part library. This is unlike using *Export To Capture Part Library* or *Export To Design Entry HDL Part Library* command, where all the changes are made to the part library before you can review the symbols by opening them in the design entry tool.

In the Model Import Wizard, specify the input model library and the output symbol library and click Next. The model names and the names of the symbol to be associated with each model is listed. The symbol shape is visible on the right.

If the users do not make any change and selects the Finish button, a message box may appear. This message box appears only if there are models for which symbols could not be found. To attach a rectangular symbols to such models click Yes. If you do not wish to attach any symbols select No. If you select yes, the symbol library generated using the Model Import

@ 2023 All Rights Reserved.

Wizard will be exactly same as the `.olb` generated using the *Export to Capture Part Library* or *Export to Design Entry HDL Part Library* command.

To know more about how to create parts using Model Import Wizard, see Using Interactive Mode on page 127.

To find out which device types the Model Editor supports, see Running the Model Editor Alone on page 51. If the Model Editor does not support the device type for the model definition that you want to create, then you can use a standard text editor to create a model definition using the PSpice .MODEL and .SUBCKT command syntax. Remember to configure the new model library.

Creating Design Entry Tool parts for all Models in a Library

Note: Creating parts for all the models in a library is not supported for libraries that have models with multiple implementation.

Using Batch Mode

1. Open the Model Editor alone.
   From the Windows Start menu, point to the installed Cadence release, and then choose PSpice Accessories, Model Editor.

2. From the File menu, choose *Export to Capture Part Library* or *Export to Design Entry HDL Part Library*, for Capture and Design Entry HDL respectively.

   The Create Parts for Library dialog box appears.

Tip

Export to <design entry tool> Part Library option is available depending on the design entry tool selected as the schematic editor in the Options dialog box. To display the Options dialog box, choose Options from the Tools menu.

3. In the Enter Input Model Library text box, specify the location of the model library for which the design entry tool parts are to be created.

   Note: You can use the Browse button to specify a different location of the `.OLB or .LIB` file.

   The *Enter Output Part Library* text box automatically displays the name and the location of the new `.OLB` file (in Capture) or the folder (in Design Entry HDL where you

@ 2023 All Rights Reserved.

want library files to be created. The displayed library name or folder name is same as specified by the user in the *Save Part To* section of the *OPTIONS* dialog box.

4. Click OK to create the part and click OK again to clear the `.ERR` log dialog box.

Caution

Recreating an already existing part library, using the Export to <design entry tool> Part Library command does not overwrite the contents of the part library. Only the new parts are appended to the library. Consider a part library, MYLIB.OLB that has two parts a1 and a2, and a model library, MYLIB.LIB that has three models a2, b1, and b3. Recreating MYLIB.OLB from MYLIB.LIB will not delete a1 from MYLIB.OLB. The modified MYLIB.OLB has four parts, a1, a2, b1, and b2.

In Design Entry HDL:

The part library is created with the same name as the .lib file. Each model in the .lib file is represented as a separate folder, with the folder name same as the model name.

Each of the Design Entry HDL part that is created has following views.

- sym_1 - Contains graphical description of the part
- chips - Contains pin name to pin number mapping needed for packaging onto physical board.
- entity - Contains the port information for the part.
- pspice_lnk - Contains the pspice.pll file. This file points to the Advanced Analysis enabled simulation model for the part.

  Note: The pspice_lnk view is created only when you create Design Entry HDL parts for Advanced Analysis enabled PSpice models using the Model Editor.

Using Interactive Mode

Important

Model Import Wizard is not recommended if you want to create new symbol shapes. Using the wizard, you can only associate existing PSpice model to existing symbols and vice-versa.You can create new symbols in OrCAD X Capture. To know more about creating new part symbols or editing existing symbol graphics, see Editing Part Graphics (Capture only) on page 144.

@ 2023 All Rights Reserved.

Invoke Model Import wizard

From Model Editor

➡ From the File drop-down menu, choose *Model Import Wizard [<design entry tool>]*.

Important

To be able to generate a specific design entry tool parts using Model Editor, select the design entry tool as the schematic editor in the Options dialog box.

From Capture

1. Select the Project Manager window in Capture.
2. From the Tools drop-down menu, select Generate Part.
3. In the Generate Part dialog box, select the Pick symbol manually check box.
4. From the Netlist/source file type drop-down list box, select PSpice Model Library.
5. To start the symbol generation process, click OK.

Using Model Import wizard

1. In the Specify Library page of the Model Import Wizard, provide inputs.
   a. Specify the name and location of the input model library (`.lib`)
   b. Specify the name and location of the output part library (`.olb`)

Tip

While specifying the library name, you must ensure that special characters, such as tilde (~), backtick (`), exclamation mark (!), dollar ($), percentage (%), caret (^), ampersand (&), plus (+), minus (-), comma (,), semi colon(;), single quotes ('), and double quotes (") are not used in the library name.

   c. Click the Next button to move to the next step of the wizard.

   When you click the Next button, the Model Import Wizard automatically starts the process of looking up symbols for each of the models in the `.lib` file and associating the symbols that match the model definition.

@ 2023 All Rights Reserved.

Important

Model Import Wizard uses the model definition to find a matching symbol for a simulation model. Therefore, Model Import Wizard can automatically match symbols only for the device types supported by the Model Editor. To know more about these device types, see Model Editor-supported device types based on device characteristic curves *on page 57.*

The matching symbols are stored in the destination part library specified by you.

2. View the symbols provided by the Model Import Wizard.

The Associate/Replace Symbol page of the wizard lists the models in the `.lib` file and the corresponding symbol names. By default, all models are listed. You can customize the view to display only those models that have symbols attached or the models that do not have any symbols attached.

To display only the models with symbols attached, ensure that only the *Models with symbols* check box is selected.

To display only the models with no symbols attached, only the *Models without symbols* check box should be selected. In this case, the Symbol Name column is blank
.

Check box / Displays:	Models with symbols	Models without symbols
Only models with symbols attached	✔	✘
Only models with no symbols attached	✘	✔
All Models	✔	✔

You can view the attached symbol by selecting a model from the list of the models with attached symbols. The symbol shape appears on the right of the wizard.

@ 2023 All Rights Reserved.

At this stage, you can complete the process of associating symbols, and close the Model import wizard by clicking the Finish button. When you click the Finish button, you receive a message stating whether you want to attach rectangular symbols to the models that do not have symbols attached to them. This message appears only if the `.lib` file has some models for which symbols were not available

In this case, you want rectangular shaped symbols to be associated with the models, select Yes. To close the wizard without attaching rectangular symbols to the models, select No.

Yes	Destination symbol library (`.olb`) has symbol looked up by the Model Import wizard
No	The `.olb` file has symbols for all the models in the `.lib` file.

3. Associate/Replace desired part symbol to a PSpice model

As the name suggests, the Associate/Replace Symbol page of the wizard can also be use to review symbol shapes and if required, attach user-defined symbols to the models.

The Model import wizard allows you to attach user-defined symbols to the models without symbols or to replace the attached symbols with a symbol of your own choice.

- To change the model-symbol association suggested by Model Import wizard, click the Replace Symbol button.
- To attach a specific symbol to a model for which matching symbols could not be found, click the Associate Symbol button.

To attach user-defined symbols to a model:

a. Click the Associate/Replace symbol button.

b. In the Select Matching page of the wizard, specify the path to the base library

@ 2023 All Rights Reserved.

A base library can be defined as the symbol library (`.olb`) that contains the desired part symbol. From the specified library, the Model Import wizard filters and lists the symbol names that can be attached to the selected model. You can then select a symbol from the list, to be associated with the model.

c. Complete the model terminal and symbol pin mapping.

Use the View Model Test button, to display the model definition for the selected mode. The pin names of the selected symbols appear in the Symbol Pin drop-down list box.

d. Select the Save button to update the destination symbol file specified in step 1, with the changes made by you.

Note: The changes made to the destination library are irreversible. Once saved, you cannot undo the changes at the click of a button.

Similarly, you can make associate an existing symbol to any of the models in the `.lib` file.

When you use the Model Import wizard to attach a symbol to a model and vice-versa, following tasks are performed:

- The value of IMPLEMENTATION TYPE property attached to the symbol is set to PSpice Model.
- The value of the IMPLEMENTATION property attached to the symbol is set to the name of the selected model. (Except in case of leveled models where the level number is removed.)
- For models based on device characteristic curves, the PSPICETEMPLATE property is updated with the pinname information.
- For template-based PSpice models, `pspice_lnk` view is generated.

Setting up Automatic Part Creation

Automatic part creation from the Model Editor is optional. By default, automatic part creation is disabled. You can enable it using the procedure below.

Note: Instead of using the PSpice default part set, you can use your own set of standard parts. To find out more, see Basing New Parts on a Custom Set of Parts on page 142.

@ 2023 All Rights Reserved.

Tip

In case you want to create a new symbol for only one model in a library that contains several models, set up the automatic part creation feature and use the File, Save command. Every time the file is saved with automatic part creation enabled, a new symbol is created for the selected model. All other symbols are left alone.

To enable automatic part creation for new models

1. In the Model Editor with a library open, choose Options from the Tools menu.
2. In the Part Creation Setup frame, select Always Create Part when Saving Model if it is not already checked.

   Note: The Always Create Part when Saving Model option will be disabled if you start the Model Editor from a design entry tool.
3. Select the Pick symbols Manually check box, to ensure that Model Import Wizard is invoked every time you generate a part symbol for a model.

   Note: In this case, the Specify Library page of the wizard is not required and is therefore, not visible. The Associate/Replace symbol page will be invoked.
4. In the Save Part To frame, define the name of the part library for the new part. Choose one of the following:

   - Part library path same as model library to create or open the `*.OLB` file that has the same filename as the open model library (`*.LIB`).

     For example, if the model library is named MYPARTS.LIB, then the Model Editor creates the part library named MYPARTS.OLB.
   - User-defined part library, and then enter a library name in the part Library Name text box.

Part Creation Examples

The examples covered in this section demonstrate the steps involved in creating a part using the Model Editor. The examples covered in this section are:

- Creating Parts in Batch Mode
- Creating parts using interactive mode

@ 2023 All Rights Reserved.

Creating Parts in Batch Mode

This example is the continuation of the Example: Creating template-based PSpice model.

The tasks covered in this example are:

- Creating parts for the models in a model library using the Export to <design entry tool> Part Library command.
- Using the Model Editor created part in a schematic design.
  - Passing parameters to a simulation model from the schematic.
  - Using model with multiple levels of simulation parameters. Difference in the use model for Model Editor-created model and OrCAD X supplied model

In this example, you will create parts for the simulation models in the LOCAL_LIB model library. The LOCAL_LIB library created as part of Example: Creating template-based PSpice model has a template-based model of an operational amplifier, OPA_LOCAL.

Creating design entry tool parts

To create design entry tool parts for all the models in the `LOCAL_LIB.LIB` library:

1. From the File menu choose *Export to Capture Part library* for Capture parts or *Export to D3esign Entry HDL Part library* for Design Entry HDL parts.

   Note: The *Export to Capture Part Library* or *Export to Design Entry Part library* option is available depending on the design entry tool you select as the schematic editor in the Options dialog box.

2. In the Create Parts for Library dialog box, ensure that the input model library is `LOCAL_LIB.LIB` and save the new design entry tool part library in the same location as the model library.
3. Click OK, to create parts.

In Capture:

The `LOCAL_LIB.OLB` that gets created has one part, `OPA_LOCAL`, with the LEVEL property is attached to the part symbol. By default, the value of the LEVEL property is set to `1`.

Template-based operational amplifier model created using the Model Editor has multiple level support. If the value of the LEVEL property is set to `1`, first level of simulation parameters are

@ 2023 All Rights Reserved.

used while simulating the model. Similarly, if the LEVEL value set to `2`, second level of simulation parameters are used, and so on.

in Design Entry HDL:

The Model Editor generated part library has 5.x architecture. The part library directory has the same name as that of the model library, `local_lib`. A new directory, `local_lib`, is generated. In the 5.x architecture, the part library further has sub directories, one for each part in the library. Therefore, the `local_lib` directory has a sub directory `OPA_LOCAL`.

With in the part library, each of the part has multiple views, such as chips, entity, pspice_lnk, and symbol views. The directory structure for the local_lib library, is shown in the following figure.

Figure 4-1 The 5.x architecture of the local_lib library

Using the Model Editor created parts in a design

In this section you will create an amplifier using two internally compensated operational amplifiers. One of the Operational Amplifier, CA1458, is supplied along with PSpice A/D in the `OPA.LIB` model library, and the second Operational Amplifier used is `OPA_LOCAL`, created using the Model Editor. Both the models are template-based and both support multiple levels of simulation parameters.

In Capture:

1. In Capture, create a new project `FUNC_GEN.OPJ`, and include `LOCAL_LIB.LIB` as one of the model library.

@ 2023 All Rights Reserved.

2. Create a circuit as shown in the following figure.

The model CA1458, is a multi-level model and by default, the model level is 3. The LEVEL property for the OPA_LOCAL model is set to 2.

Note: You can also pick the design from, ...\tools\pspice\tutorial\capture\modeleditor.

3. To simulate the circuit, choose Run from the PSpice menu in Capture.

In Design Entry HDL:

1. In Design Entry HDL, create a new project amplifier, and include `local_lib.lib` as one of the model library.

@ 2023
All Rights Reserved.

2. Create a circuit design as shown in the following figure.

The model CA1458, is also a multi-level model. By default, the model level is 3. For OPA_LOCAL, the value of the LEVEL property is set to2.

Note: You can also pick the design from, `..\tools\pspice\tutorial\concept\modeleditor`. Besides the `local_lib`, and `opa.lib` other libraries included in the project are `pspice_elem` and `analog`. If you use the design from the hierarchy, edit the `cds.lib` to specify the correct location of `local_lib`.

3. To simulate the circuit, choose *Run* from the *PSpice* menu in Design Entry HDL.

Changing the level of simulation parameters used in the design

We will now modify the circuit design, such that level 2 simulation parameters are used for CA1458 and level 3 simulation parameters are used for the OPA_LOCAL model, created using the Model Editor.

Changing the level for CA1458

To change the level of simulation parameters for CA1458 from 3 to 2,

1. Select the part *CA1458*.
2. In Capture, from the Edit menu, choose *Properties*.

   In Design Entry HDL, choose *Text - Attributes*.

@ 2023
All Rights Reserved.

3. In Capture, in the Property editor dialog box, change the value of LEVEL property from 3 to2.

   In Design Entry HDL, in the Attributes dialog box, change the value of LEVEL property from 3 to 2.

4. In Capture, click *Apply* and close the Property Editor.

   In Design Entry HDL, click *OK* to save the changes.

Similarly, change the value of the LEVEL property on OPA_LOCAL from 2 to 3.

You can now simulate the circuit using PSpice A/D or PSpice Advanced Analysis.

Changing Simulation Properties from design entry tools

For all the editable simulation properties, you can specify the value of simulation parameters from the design entry tools.

In Capture:

1. Select OPA_LOCAL.
2. From the Edit menu, choose Properties.
3. In the Property Editor dialog box, select New Row.
4. In the Add New Row dialog box, enter the name of the simulation parameter in the Name text box. Enter VOS.
5. In the Value text box, add the value of the simulation parameter as $2e^{-3}$ and select OK.
6. To make the property and its value visible, first select property row and then select the Display button in the Property Editor dialog box.
7. Specify the Display Format. Select Name and Value.
8. Select Apply and close the dialog box.

The simulation parameter and its value appear in the Capture. If you now simulate the design, the simulation parameter value specified in Capture will be used.

In Design Entry HDL:

1. Select OPA_LOCAL_3.

@ 2023 All Rights Reserved.

2. From the Edit menu, choose Properties.
3. In the Attributes dialog box, select Add.
4. In the new row enter the name of the simulation parameter in the Name text box. Enter VOS.
5. In the Value text box, add the value of the simulation parameter as $2e^{-3}$ and select OK.
6. To make the property and its value visible, first select property row and then select the Display button in the Property Editor dialog box.
7. Specify the Display Format. Select Name and Value.
8. Select *Apply* and close the dialog box.

The simulation parameter and its value are now visible in Design Entry HDL. If you now simulate the design, the simulation parameter value specified in Design Entry HDL will be used.

Creating parts using interactive mode

In this section, we will use the Model Import wizard to create models stored in a user defined library, MYLIB. This library consists of four simulation models. These are `LM339`; which is a voltage comparator macro model subcircuit, `INA105E`; which is an operational amplifier plus precision resistor network, `LF442A/NS`; which is a JFET OPAMP, and `mybjt`; which is a Bipolar Junction Transistor.

Launch Model Import Wizard

@ 2023 All Rights Reserved.

1. From the File drop-down menu in Model Editor, choose Model Import Wizard [<design entry tool>].

Provide Inputs

1. In the Specify Library page of the wizard, specify the location of `MYLIB.LIB`.
2. Specify the name of the part library in which the part symbols generated by the Wizard are to be stored.

   You can either specify the name of an existing `.olb` file or create a new `.olb` file.

   When you specify the name of an existing `.olb` file, you also need to specify if symbols should be created for all the models in the model library while replacing the existing symbols, or whether part symbols are to be created only for the models without the symbols.

   The name and the location of the destination symbol library gets populated by default. For the current example, accept the default name, `mylib.olb` and click Next.

@ 2023 All Rights Reserved.

Depending on the model definition, the Model Import wizard could locate appropriate part symbol for the BJT model only. The named of the symbol attached to the `mybjt` model, is listed in the Symbol Name column and the symbol shape is also visible.

3. You can now complete one of the following steps:

   a. Close the wizard without attaching any symbols of the rest of the three models in `mylib.lib`.

   b. Close the wizard, after Model Import wizard attaches rectangular symbols to rest of the models in `mylib.lib`.

   c. Use the Model Import wizard to select and attach an existing symbol shape to one or all the models in `mylib.lib`.

   To execute the first two options, click the Finish button. In the message box that appears, select No to close the library and Yes to attach rectangular symbols and then close the library.

Associating desired part symbol to a PSpice model

We will now use the Model Import wizard to attach an existing user-defined, symbol to `LF442A/NS`; which is a JFET OPAMP. `LF442A` is a Dual Low Power JFET Input Operational

@ 2023 All Rights Reserved.

Amplifier, downloaded from National Semiconductor's web site. We will pick up the symbol from the symbol library `opamp.olb`, which contains symbols for different types of OPAMPs.

1. In the Associate/Replace page of the Model Import Wizard, select `LF442A/NS`, and click Associate Symbol.
2. In the Select Matching page of the wizard, specify the name and the location of `opamp.olb`.

   All symbols that match the model definition are listed in the Matching Symbols list.
3. From the Matching symbols list, select `LM158` and click Next.
4. Before you start mapping the model terminals to the symbol pin names, click the View Model Text button.

   The model definition of LF442A/NS appears in a different window.

5. Map the model terminals and the symbol pin names.

Model terminal	Mapped to Symbol Pin
1	+
2	-
99	V+
50	V-
28	OUT

@ 2023 All Rights Reserved.

6. To attach the symbol to LF442A/NS, click the Save Symbol button.

   The name of the symbol attached to the model appears in the Symbol Name list.

You have successfully attached a symbol to the selected model. Similarly, you can attach models for the other two symbols as well.

Note: You can use the Model Import wizard to replace a symbol attached to a model with another symbol. To do this, select the model that has a symbol attached to it, and click the Replace Symbol button.

Important

Model Import Wizard can be invoked from Capture to associate a PSpice model to an existing Capture Symbol. To know more about associating PSpice model to a Capture symbol, see *OrCAD X Capture User Guide*.

Basing New Parts on a Custom Set of Parts

If you are using the Model Editor to automatically generate parts for model definitions, and you want to base the new parts on a custom graphic standard (rather than the PSpice A/D default parts), then you can change which underlying parts either application uses by setting up your own set of parts.

Note: If you use a custom part set, the Model Editor always checks the custom part library first for a part that matches the model definition. If none can be found, they use the PSpice A/D default part instead.

To create a custom set of parts for automatic part generation

1. Create a part library with the custom parts.

   Be sure to name these parts by their device type as shown in the table. This is how the Model Editor determines which part to use for a model definition.

@ 2023 All Rights Reserved.

For more information on creating parts, refer to the *OrCAD X Capture User Guide.*

Table 4-1 Symbol Names for custom symbol generation for regular PSpice models

For this device type...	Use this symbol name...	For this device type...	Use this symbol name...
Bipolar transistor: LPNP	LPNP	Magnetic core	CORE
Bipolar transistor: NPN	NPN	MOSFET: N-channel	NMOS
Bipolar transistor: PNP	PNP	MOSFET: P-channel	PMOS
Capacitor	CAP	OPAMP: 5-pin	OPAMP5
Darlington: N-channel	DARNPN	OPAMP: 7-pin	OPAMP7
Darlington: P-channel	DARPNP	Resistor	RES
Diode	DIODE	Switch: voltage-controlled	VSWITCH
GaAsFET	GASFET	Transmission line	TRN
IGBT: N-channel	NIGBT	Voltage comparator	VCOMP
Inductor	IND	Voltage comparator: 6 pin	VCOMP6
JFET: N-channel	NJF	Voltage reference	VREF
JFET: P-channel	PJF	Voltage regulator	VREG

Table 4-2 Symbol Names for Custom Symbol Generation for template-based Models

For this device type...	Use this symbol name...	For this device type...	Use this symbol name...
Bipolar transistor: LPNP		Magnetic core	AACORE
Bipolar transistor: NPN	AANPN3	MOSFET: N-channel	AANMOSFET3
Bipolar transistor: PNP	AAPNP3	MOSFET: P-channel	AAPMOSFET3
Capacitor	CAP	OPAMP: 5-pin	AA5_PIN_OPAMP
Darlington: N-channel	AADARNPN3	OPAMP: 7-pin	AA7_PIN_OPAMP
Darlington: P-channel	AADARPNP3	Resistor	
Diode	AADIODE	Switch: voltage-controlled	
GaAsFET		Transmission line	

@ 2023 All Rights Reserved.

Table 4-2 Symbol Names for Custom Symbol Generation for template-based Models

For this device type...	Use this symbol name...	For this device type...	Use this symbol name...
IGBT: N-channel	AANIGBT3	Voltage comparator	
Inductor		Voltage comparator: 6 pin	
JFET: N-channel	AANCHANNEL3	Voltage reference	
JFET: P-channel	AANPHANNEL3	Voltage regulator	AAVREG

2. For each custom part, set its IMPLEMENTATION property to `` `M `` where `` ` `` is a back-single quote or grave character.

   This tells the Model Editor to substitute the correct model name.

To base new parts on custom parts using the Model Editor

1. In the Model Editor with the library open, choose Part Creation Setup from the Options menu, and enable automatic part creation as described in Setting up Automatic Part Creation.
2. In the Base Parts On frame, enter the name of the existing part library (`*.OLB`) that contains your custom parts.
3. Click OK.

Editing Part Graphics (Capture only)

If you created parts using the Model Editor, and you want to make further changes, the following sections explain a few important things to remember when you edit the parts.

@ 2023 All Rights Reserved.

How Capture Places Parts

When placing parts on the schematic page, the schematic page editor uses the grid as a point of reference for different editing activities. The part's pin ends are positioned on the grid points.

To edit a part in a library

1. From Capture's File menu, point to Open, then choose Library.
2. Select the library that has the part you want to edit.

   The library opens and displays all its parts.
3. Double-click the part you want to edit.

   The part appears in the part editor.
4. Edit the part.

   You can resize it, add or delete graphics, and add or delete pins. For more information about specific part editing tasks, refer to the *OrCAD X Capture User Guide*.
5. After you have finished editing the part, from the File menu, choose Save to save the part to its library.

Here are the things to check when editing part properties:

- Does the PSPICETEMPLATE specify the correct number of pins/nodes?
- Are the pins/nodes in the PSPICETEMPLATE specified in the proper order?
- Do the pin/node names in the PSPICETEMPLATE match the pin names on the part?

@ 2023 All Rights Reserved.

Defining Grid Spacing

Grid spacing for graphics

The grid, denoted by evenly spaced grid points, regulates the sizing and positioning of graphic objects and the positioning of pins. The default grid spacing with snap-to-grid enabled is 0.10", and the grid spacing is 0.01".

You can turn off the grid spacing when you need to draw graphics in a tighter space.

To edit the part graphics

1. In Capture's part editor, display the part you want to edit.
2. Select the line, arc, circle, or other graphic object you want to change, and do any of the following:
   - ❑ To stretch or shrink the graphic object, click and drag one of the size handles.
   - ❑ To move the entire part graphic, click and drag the edge of the part.

     The part body border automatically changes to fit the size of the part graphic.
3. After you have finished editing the part, from the File menu, choose Save to save the part to its library.

Note: When changing part graphics, check to see that all pins are on the grid.

Grid spacing for pins

The part editor always places pins on the grid, even when the snap-to-grid option is turned off. The size of the part is relative to the pin-to-pin spacing for that part. That means that pins placed one grid space apart in the part editor are displayed as one grid space apart in the schematic page editor.

Pins *must* be placed on the grid at integer multiples of the grid spacing. Because the default grid spacing for the Schematic Page Grid is set at 0.10", you will achieve the best results by setting pin spacing in the Part and Symbol Grid at 0.10" intervals from the origin of the part and at least 0.10" from any adjacent pins. For more information about grid spacing and pin placement, refer to the *OrCAD X Capture User Guide*.

The part editor considers pins that are not placed at integer multiples of the grid spacing from the origin as *off-grid*, and a warning appears when you try to save the part.

@ 2023 All Rights Reserved.

Here are two guidelines:

- Make sure Pointer Snap to Grid is enabled when editing part pins and editing schematic pages so you can easily make connections.
- Make sure the Part and Symbol Grid spacing *matches* the Schematic Page Grid spacing.

Note: Pin changes that alter the part template can occur if you either change pin names or delete pins.

In these cases you must adjust the value of the part's PSPICETEMPLATE property to reflect these changes. To find out how, see Pin callout in subcircuit templates on page 156.

Attaching Models to Parts

If you create parts and want to simulate them, you need to attach model implementations to them. If you created your parts using any of the methods discussed in this chapter, then your part will have a model implementation already attached to it.

MODEL

The IMPLEMENTATION property defines the name of the model that PSpice A/D must use for simulation. When attaching this implementation, this rule applies:

- The Implementation name should match the name of the .MODEL or .SUBCKT definition of the simulation model as it appears in the model library (`*.LIB`).

Example:

If your design includes a 2N2222 bipolar transistor with a .MODEL name of Q2N2222, then the Implementation name for that part should be Q2N2222.

Note: Make sure that the model library containing the definition for the attached model is configured in the list of libraries for your project. See Configuring Model Libraries on page 88 for more information.

For more information on model editing in general, see Chapter 3, "." For specific information on changing model references, see Changing the Model Reference to an Existing Model Definition on page 85.

@ 2023 All Rights Reserved.

To attach a model implementation

1. In the schematic page editor, double-click a part to display the Parts spreadsheet of the Property Editor.

   Assume the spreadsheet is displayed in columns and follow the instructions below.

2. Click on the empty cell under the Implementation Type column.

   A drop-down list appears in the cell.

3. From the Implementation Type drop-down list, select PSpice Model.

4. Click on the empty cell under the Implementation column, and type the name of the model to attach to the part.

   Note: You do not need to enter an Implementation Path because PSpice A/D searches for the model in the list of model libraries you configure for this project.

5. Click Apply to update the design, then close the Parts spreadsheet.

Caution

In case you want to reuse the part symbol that was originally attached to a characteristic curve-based model, with a template-based simulation model, you must delete the PSPICETEMPLATE property from the part symbol.

Example:

Consider a scenario where you have two simulation models for a bipolar transistor, with the same name 2n2222 in two different libraries. First simulation model is based on characteristic curves and the second based on PSpice templates. Both the simulation models have same name, therefore the same value for the IMPLEMENTATION property.

The simulation model based on characteristic curves, which is of .MODEL type, is used in a schematic design. The part symbol for the bipolar transistor will have the IMPLEMENTAION property set to Q2n2222 and the PSPICETEMPLATE property attached to it. Now modify the BJT symbol by attaching a template-based model to the symbol. The value of the IMPLEMENTATION property will not change because the name of the template-based model is same as that of the characteristic curves-based model. Therefore, to ensure that the correct model is used during simulation, you must delete the PSPICETEMPLATE property from the part symbol and configure the library containing the template-based BJT model.

Note: You can check whether the right model is being used or not, by viewing the simulation netlist generated by PSpice A/D. The simulation netlist for a .MODEL type BJT model starts with Q, where as the simulation netlist for a .SUBCKT model starts with X.

@ 2023 All Rights Reserved.

Defining Part Properties Needed for Simulation

If you created your parts using any of the methods discussed in this chapter, then your part will have these properties already defined for it:

- PSPICETEMPLATE for simulation in both Capture and Design Entry HDL
- PART and REFDES for identification in Capture
- LOCATION for identification in Design Entry HDL

You can also add other simulation-specific properties for digital parts: IO_LEVEL, MNTYMXDLY, and PSPICEDEFAULTNET (for pins).

Example:

If you create a part that has electrical behavior described by the subcircuit definition that starts with:

```
.SUBCKT 7400 A B Y
+  optional: DPWR=$G_DPWR DGND=$G_DGND
+  params: MNTYMXDLY=0 IO_LEVEL=0
```

then the appropriate part properties are:

```
IMPLEMENTATION = 7400
MNTYMXDLY = 0
IO_LEVEL = 0
PSPICETEMPLATE = X^@REFDES %A %B %Y %PWR %GND
  @MODEL PARAMS:IO_LEVEL=@IO_LEVEL   MNTYMXDLY=@MNTYMXDLY
```

Note: For clarity, the PSPICETEMPLATE property value is shown here in multiple lines; in a part definition, it is specified in one line (no line breaks).

To edit a property needed for simulation in Capture:

1. In the schematic page editor, select the part to edit.
2. From the Edit menu, choose Properties to display the Parts spreadsheet of the Property Editor.
3. Click in the cell of the column you want to change (for example, PSPICETEMPLATE), or click the New button to add a property (and type the property name in the Name text box).
4. If needed, type a value in the Value text box.
5. Click Apply to update the design, then close the spreadsheet.

@ 2023 All Rights Reserved.

To edit a property needed for simulation in Design Entry HDL:

1. From the Text menu in Design Entry HDL, choose Attributes.
2. Click on the part to display the Attributes dialog box.
3. Click in the Name text box of the property you want to change (for example, PSPICETEMPLATE), or click the Add button to add a property (and type the property name in the Name text box).
4. Specify the value of the property in the Value text box.
5. Click OK to save the changes and close the Attributes dialog box.

Here are the things to check when editing part properties:

1. Does the PSPICETEMPLATE specify the correct number of pins/nodes?
2. Are the pins/nodes in the PSPICETEMPLATE specified in the proper order?
3. Do the pin/node names in the PSPICETEMPLATE match the pin names on the part?

Table 4-3

Property	See this
PSPICETEMPLATE	PSPICETEMPLATE on page 150
IO_LEVEL	IO_LEVEL on page 158
MNTYMXDLY	MNTYMXDLY on page 158
PSPICEDEFAULTNET	PSPICEDEFAULTNET on page 159

PSPICETEMPLATE

The PSPICETEMPLATE property defines the PSpice syntax for the part's netlist entry. When creating a netlist, the design entry tool substitutes actual values from the circuit into the appropriate places in the PSPICETEMPLATE syntax, then saves the translated statement to the netlist file.

@ 2023 All Rights Reserved.

Caution

The PSPICETEMPLATE property is required only for models based on device characteristic curves. For template-based models, netlisting is done using the PORT_ORDER information available in the device property file.

Note: The PSPICETEMPLATE property is not required with template-based models. Therefore, while modifying a design, if you attach a template-based model to a part originally attached to a characteristic curve based model, then besides changing the Implementation property, you must also delete the PSPICETEMPLATE property.

Any part that you want to simulate must have a defined PSPICETEMPLATE property. These rules apply:

- The pin names specified in the PSPICETEMPLATE property must match the pin names on the part.
- The number and order of the pins listed in the PSPICETEMPLATE property must match those for the associated .MODEL or .SUBCKT definition referenced for simulation.
- The first character in a PSPICETEMPLATE must be a PSpice device letter appropriate for the part (such as Q for a bipolar transistor).

  Device types and PSpice device letters are also listed in the *PSpice A/D Reference Guide*.

Caution

Creating parts not intended for simulation
Some part libraries contain parts designed only for board layout; PSpice A/D cannot simulate these parts. This means they do not have PSPICETEMPLATE properties or that the PSPICETEMPLATE property value is blank.

PSPICETEMPLATE syntax

The PSPICETEMPLATE contains:

- *regular characters* that the schematic page editor interprets verbatim
- *property names* and *control characters* that the schematic page editor translates

@ 2023 All Rights Reserved.

Regular characters in templates

Regular characters include the following:

- alphanumerics
- any keyboard part *except* the special syntactical parts used with properties (@ & ? ~ #).
- white space

An *identifier* is a collection of regular characters of the form:

alphabetic character [*any other regular character*]*.

Property names in templates

Property names are preceded by a special character as follows:

[@ | ? | ~ | # | &]<*identifier*>

The schematic page editor processes the property according to the special character as shown in the following table.

Table 4-4

This syntax...[1]	Is replaced with this...
@<*id*>	Value of <*id*>. Error if no <*id*> attribute or if no value assigned.
&<*id*>	Value of <*id*> if <*id*> is defined.
?<*id*>*s…s*	Text between *s…s* separators if <*id*> is defined.
?<*id*>*s…ss…s*	Text between the first *s…s* separators if <*id*> is defined, else the second *s…s* clause.
~<*id*>*s…s*	Text between *s…s* separators if <*id*> is undefined.
~<*id*> *s…ss…s*	Text between the first *s…s* separators if <*id*> is undefined, else the second *s…s* clause.

@ 2023 All Rights Reserved.

Table 4-4

This syntax...[1]	Is replaced with this...
#<id>s…s	Text between *s…s* separators if *<id>* is defined, but delete rest of template if *<id>* is undefined.

1. *s* is a separator character

Separator characters include commas (,), periods (.), semicolons (;), forward slashes (/), and vertical
bars (|). You must always use the same character to specify an opening-closing pair of separators.

Example: The template fragment `?G|G=@G||G=1000|` uses the vertical bar as the separator between the if-then-else parts of this conditional clause. If G has a value, then this fragment translates to G=<*G property value*>. Otherwise, this fragment translates to `G=1000`.

Note: You can use different separator characters to nest conditional property clauses.

The ^ character in templates

The schematic page editor replaces the ^ character with the complete hierarchical path to the device being netlisted.

The \n character sequence in templates

The part editor replaces the character sequence \n with a new line. Using \n, you can specify a multi-line netlist entry from a one-line template.

The % character and pin names in templates

Pin names are denoted as follows:

%<*pin name*>

where *pin name* is one or more regular characters.

The schematic page editor replaces the %<*pin name*> clause in the template with the name of the net connected to that pin.

@ 2023 All Rights Reserved.

The end of the pin name is marked with a separator (see Property names in templates). To avoid name conflicts in PSpice A/D, the schematic page editor translates the following characters contained in pin names.

Table 4-5

Pin name character...	Is replaced with this...
<	l (L)
>	g
=	e
\XXX\	XXXbar

Note: To include a literal % character in the netlist, type `%%` in the template.

Caution

Recommended scheme for netlist templates
Templates for devices in the part library start with a PSpice device letter, followed by the hierarchical path, and then the reference designator (REFDES) property. We recommend that you adopt this scheme when defining your own netlist templates.

Example: R^@REFDES ... for a resistor

PSPICETEMPLATE examples

Simple resistor (R) template

The R part has:

- two pins: 1 and 2
- two required properties: REFDES and VALUE

Template

```
R^@REFDES %1 %2 @VALUE
```

Sample translation

```
R_R23 abc def 1k
```

where REFDES equals R23, VALUE equals 1k, and R is connected to nets abc and def.

@ 2023 All Rights Reserved.

Voltage source with optional AC and DC specifications (VAC) template

The VAC part has:

- two properties: AC and DC
- two pins: + and -

Template

```
V^@REFDES %+ %- ?DC|DC=@DC| ?AC|AC=@AC|
```

Sample translation

```
V_V6 vp vm DC=5v
```

where REFDES equals V6, VSRC is connected to nodes vp and vm, DC is set to 5v, and AC is undefined.

Sample translation

```
V_V6 vp vm DC=5v AC=1v
```

where, in addition to the settings for the previous translation, AC is set to 1v.

Parameterized subcircuit call (X) template

Suppose you have a subcircuit Z that has:

- two pins: a and b
- a subcircuit parameter: G, where G defaults to 1000 when no value is supplied

To allow the parameter to be changed on the schematic page, treat G as an property in the template.

Template

```
X^@REFDES %a %b Z PARAMS: ?G|G=@G|
  ~G|G=1000|
```

Note: For clarity, the PSPICETEMPLATE property value is shown here in multiple lines; in a part definition, it is specified in one line (no line breaks).

Equivalent template (using the if...else form)

```
X^@REFDES %a %b Z PARAMS: ?G|G=@G||G=1000|
```

Sample translation

```
X_U33 101 102 Z PARAMS: G=1024
```


@ 2023 All Rights Reserved.

where REFDES equals U33, G is set to 1024, and the subcircuit connects to nets 101 and 102.

Sample translation

```
X_U33 101 102 Z PARAMS: G=1000
```

where the settings of the previous translation apply except that G is undefined.

Digital stimulus parts with variable width pins template

For a digital stimulus device template (such as that for a DIGSTIM part), a pin name can be preceded by a * character. This signifies that the pin can be connected to a bus and the width of the pin is set to be equal to the width of the bus.

Template

```
U^@REFDES STIM(%#PIN, 0) %*PIN
  \n+ STIMULUS=@STIMULUS
```

where #PIN refers to a variable width pin.

Note: For clarity, the PSPICETEMPLATE property value is shown here in multiple lines; in a part definition, it is specified in one line (no line breaks).

Sample translation

```
U_U1 STIM(4,0) 5PIN1 %PIN2 %PIN3 %PIN4
+ STIMULUS=mystim
```

where the stimulus is connected to a four-input bus, a[0-3].

Pin callout in subcircuit templates

The number and sequence of pins named in a template for a subcircuit must agree with the definition of the subcircuit itself—that is, the node names listed in the .SUBCKT statement, which heads the definition of a subcircuit. These are the pinouts of the subcircuit. To find out how to define subcircuits, refer to the .SUBCKT command in the *PSpice A/D Reference Guide*.

Example:

Consider the following first line of a (hypothetical) subcircuit definition:

```
.SUBCKT SAMPLE 10 3 27 2
```

The four numbers following the name SAMPLE—10, 3, 27, and 2—are the node names for this subcircuit's pinouts.

@ 2023 All Rights Reserved.

Now suppose that the part definition shows four pins:

```
IN+      OUT+      IN-      OUT-
```

The number of pins on the part equals the number of nodes in the subcircuit definition.

If the correspondence between pin names and nodes is as follows:

Table 4-6

This node name...	Corresponds to this pin name...
10	IN+
3	IN-
27	OUT+
2	OUT-

then the template looks like this:

```
X^@REFDES %IN+ %IN- %OUT+ %OUT- @MODEL
```

The *rules of agreement* (rules for pin callout in subcircuit templates) are outlined in the following figure.

* Unmodeled pins may appear on a part (like the two voltage offset pins on a 741 opamp part). These pins are not netlisted and do not appear on the template.

@ 2023 All Rights Reserved.

IO_LEVEL

All digital parts provided in the PSpice libraries have an IO_LEVEL property.

The IO_LEVEL property defines what level of interface subcircuit model PSpice A/D must use for a digital part that is connected to an analog part.

To use the IO_LEVEL property with a digital part

1. Add the IO_LEVEL property to the part and assign a value shown in the table below.

Table 4-7

Assign this value...	To use this interface subcircuit (level)...
0	circuit-wide default
1	AtoD1 and DtoA1
2	AtoD2 and DtoA2
3	AtoD3 and DtoA3
4	AtoD4 and DtoA4

2. Use this property in the PSPICETEMPLATE property definition (IO_LEVEL is also a subcircuit parameter used in calls for digital subcircuits).

   Example:

```
PSPICETEMPLATE=X^@REFDES %A %B %C %D %PWR %GND
  @MODEL PARAMS:\n+
  IO_LEVEL=@IO_LEVEL
  MNTYMXDLY=@MNTYMXDLY
```

Note: For clarity, the PSPICETEMPLATE property value is shown here in multiple lines; in a part definition, it is specified in one line (no line breaks).

MNTYMXDLY

All digital parts provided in the PSpice libraries have a MNTYMXDLY property.

The MNTYMXDLY property defines the digital propagation delay level that PSpice A/D must use for a digital part. To find out more about propagation delays, see Timing Characteristics on page 218 and Selecting Propagation Delays on page 442.

@ 2023 All Rights Reserved.

To use the MNTYMXDLY property with a digital part

1. Add the MNTYMXDLY property to the part and assign a value shown in the table below.

Table 4-8

Assign this value...	To use this propagation delay...
0	circuit-wide default
1	minimum
2	typical
3	maximum
4	worst-case (min/max)

2. Use this property in the PSPICETEMPLATE property definition (MNTYMXDLY is also a subcircuit parameter used in calls for digital subcircuits).

   Example:

```
PSPICETEMPLATE=X^@REFDES %A %B %C %D %PWR %GND
  @MODEL PARAMS:\n+
  IO_LEVEL=@IO_LEVEL
  MNTYMXDLY=@MNTYMXDLY
```

Note: For clarity, the PSPICETEMPLATE property value is shown here in multiple lines; in a part definition, it is specified in one line (no line breaks).

PSPICEDEFAULTNET

The PSPICEDEFAULTNET pin property defines the net name to which a hidden (invisible) pin is connected. Hidden pins are typically used for power and ground on digital parts.

To use the PSPICEDEFAULTNET property with a digital part

1. For each PSPICEDEFAULTNET property, assign the name of the digital net to which the pin is connected.

   Example:

   If power (PWR) and ground (GND) pins of a digital part connect to the digital nets $G_DPWR and $G_DGND, respectively, then the PSPICEDEFAULTNET properties for these pins are:

@ 2023 All Rights Reserved.

```
PSPICEDEFAULTNET=$G_DPWR
PSPICEDEFAULTNET=$G_DGND
```

2. Use the appropriate hidden pin name in the PSPICETEMPLATE property definition.

   Example:

   If the name of the hidden power pin is PWR and the name of the hidden ground pin is GND, then the template might look like this:

```
PSPICETEMPLATE=X^@REFDES %A %B %C %D %PWR %GND
  @MODEL PARAMS:\n+
  IO_LEVEL=@IO_LEVEL
  MNTYMXDLY=@MNTYMXDLY
```

Note: For clarity, the PSPICETEMPLATE property value is shown here in multiple lines; in a part definition, it is specified in one line (no line breaks).

@ 2023 All Rights Reserved.

5

Analog Behavioral Modeling

You can use the Analog Behavioral Modeling (ABM) feature of PSpice A/D to make flexible descriptions of electronic components in terms of a transfer function or lookup table. In other words, a mathematical relationship is used to model a circuit segment, so you do not need to design the segment component by component.

The part library contains several ABM parts that are classified as either control system parts or as PSpice-equivalent parts.

Control system parts are defined with the reference voltage preset to ground so that each controlling input and output are represented by a single pin in the part.

PSpice-equivalent parts reflect the structure of the PSpice E and G device types, which respond to a differential input and have double-ended output.

You can also use the Device Equations Developer's Kit for modeling of this type, but we recommend using the ABM feature wherever possible. With the Device Equations Developer's Kit, the PSpice A/D source code is actually modified. While this is more flexible and produces faster results, it is also much more difficult to use and to troubleshoot. Also, any changes you make using the Device Equations Developer's Kit must be made to all new PSpice A/D updates you install.

Note: The Device Equations Developer's Kit is available to qualified customers only. Please contact PSpice A/D Customer Support for qualification criteria.

Device models made with ABM can be used for most cases, are much easier to create, and are compatible with PSpice A/D updates.

Related Topics

- Basic Controlled Sources
- Control System Parts
- PSpice-equivalent parts

@ 2023 All Rights Reserved.

The ABM Part Library File

The ABM part library contains the ABM components. This library contains two sections.

The first section has parts that you can quickly connect to form control system types of circuits. These components have names like SUM, GAIN, LAPLACE, and HIPASS.

The second section contains parts that are useful for more traditional controlled source forms of schematic parts. These PSpice-equivalent parts have names like EVALUE and GFREQ and are based on extensions to traditional PSpice E and G device types.

Implement ABM components by using PSpice primitives; there is no corresponding `ABM.LIB` model library. A few components generate multi-line netlist entries, but most are implemented as single PSpice E or G device declarations. See ABM Part Templates on page 165 for a description of PSPICETEMPLATE properties and their role in generating netlist declarations. See Implementation of PSpice-equivalent parts on page 192 for more information about PSpice E and G syntax.

@ 2023 All Rights Reserved.

Placing and Specifying ABM Parts

Place and connect ABM parts the same way you place other parts. After you place an ABM part, you can edit the instance properties to customize the operational behavior of the part. This is equivalent to defining an ABM expression describing how inputs are transformed into outputs. The following sections describe the rules for specifying ABM expressions.

Net names and device names in ABM expressions

In ABM expressions, refer to signals by name. This is also considerably more convenient than having to connect a wire from a pin on an ABM component to a point carrying the voltage of interest.

If you used an expression such as V(2), then the referenced net (2 in this case) is interpreted as the name of a local or global net. A local net is a labeled wire or bus segment in a hierarchical schematic, or a labeled offpage connector. A global net is a labeled wire or bus segment at the top level, or a global connector.

Note: The name of an interface port does not extend to any connected nets. To refer to a signal originating at an interface port, connect the port to an offpage connector of the desired name.

OrCAD X Capture recognizes these constructs in ABM expressions:

```
V(<net name>)
V(<net name>,<net name>)
I(<vdevice>)
```

When one of these is recognized, Capture searches for *<net name>* or *<vdevice>* in the net name space or the device name space, respectively. Names are searched for first at the hierarchical level of the part being netlisted. If not found there, then the set of global names is searched. If the fragment is not found, then a warning is issued but Capture still outputs the resulting netlist. When a match is found, the original fragment is replaced by the fully qualified name of the net or device.

For example, suppose we have a hierarchical part U1. Inside the schematic representing U1 we have an ABM expression including the term V(Reference). If “Reference” is the name of a local net, then the fragment written to the netlist will be translated to V(U1_Reference). If “Reference” is the name of a global net, the corresponding netlist fragment will be V(Reference).

Names of voltage sources are treated similarly. For example, an expression including the term I(Vsense) will be output as I(V_U1_Vsense) if the voltage source exists locally, and as I(V_Vsense) if the voltage source exists at the top level.

@ 2023 All Rights Reserved.

Forcing the use of a global definition

If a net name exists both at the local hierarchical level and at the top level, the search mechanism used by Capture will find the local definition. You can override this, and force Capture to use the global definition, by prefixing the name with a single quote (') character.

For example, suppose there is a net called Reference both inside hierarchical part U1 and at the top level. Then, the ABM fragment V(Reference) will result in V(U1_Reference) in the netlist, while the fragment V('Reference) will produce V(Reference).

@ 2023 All Rights Reserved.

ABM Part Templates

For most ABM parts, a single PSpice “E” or “G” device declaration is output to the netlist per part instance. The PSPICETEMPLATE property in these cases is straightforward. For example the LOG part defines an expression variant of the E device with its output being the natural logarithm of the voltage between the input pin and ground:

```
E^@REFDES %out 0 VALUE { LOG(V(%in)) }
```

The fragment E^@REFDES is standard. The “E” specifies a PSpice A/D controlled voltage source (E device); %in and %out are the input and output pins, respectively; VALUE is the keyword specifying the type of ABM device; and the expression inside the curly braces defines the logarithm of the input voltage.

Several ABM parts produce more than one primitive PSpice A/D device per part instance. In this case, the PSPICETEMPLATE property may be quite complicated. An example is the DIFFER (differentiator) part. This is implemented as a capacitor in series with a current sensor together with an E device which outputs a voltage proportional to the current through the capacitor.

The template has several unusual features: it gives rise to three primitives in the PSpice A/D netlist, and it creates a local node for the connection of the capacitor and its current-sensing V device.

```
C^@REFDES %in $$U^@REFDES 1\n
V^@REFDES $$U^@REFDES 0 0v\n
E^@REFDES %out 0 VALUE {@GAIN * I(V^@REFDES)}
```

Note: For clarity, the template is shown on three lines although the actual template is a single line.

The fragments C^@REFDES, V^@REFDES, and E^@REFDES create a uniquely named capacitor, current sensing V device, and E device, respectively. The fragment $$U^@REFDES creates a name suitable for use as a local node. The E device generates an output proportional to the current through the local V device.

@ 2023 All Rights Reserved.

Control System Parts

Control system parts have single-pin inputs and outputs. The reference for input and output voltages is analog ground (0) from the SOURCE.OLB library. These components can be connected together with no need for dummy load or input resistors.

Table 5-1 lists the control system parts, grouped by function. Also listed are characteristic properties that may be set. In the sections that follow, each part and its properties are described in more detail.

Table 5-1 Control system parts

Category	Part	Description	Properties
Basic components	CONST	constant	VALUE
	SUM	adder	
	MULT	multiplier	
	GAIN	gain block	GAIN
	DIFF	subtractor	
Limiters	LIMIT	hard limiter	LO, HI
	GLIMIT	limiter with gain	LO, HI, GAIN
	SOFTLIM	soft (tanh) limiter	LO, HI, GAIN
Chebyshev filters	LOPASS	lowpass filter	FP, FS, RIPPLE, STOP
	HIPASS	highpass filter	FP, FS, RIPPLE, STOP
	BANDPASS	bandpass filter	F0, F1, F2, F3, RIPPLE, STOP
	BANDREJ	band reject (notch) filter	F0, F1, F2, F3, RIPPLE, STOP

@ 2023 All Rights Reserved.

Table 5-1 Control system parts, *continued*

Category	Part	Description	Properties
Integrator and differentiator	INTEG	integrator	GAIN, IC
	DIFFER	differentiator	GAIN
Table look-ups	TABLE	lookup table	ROW1...ROW5
	FTABLE	frequency lookup table	ROW1...ROW5
Laplace transform	LAPLACE	Laplace expression	NUM, DENOM
Math functions (where ‘x’ is the input)	ABS	$\|x\|$	
	SQRT	$x^{1/2}$	
	PWR	$\|x\|^{EXP}$	EXP
	PWRS	x^{EXP}	EXP
	LOG	$ln(x)$	
	LOG10	$log(x)$	
	EXP	e^x	
	SIN	$sin(x)$	
	COS	$cos(x)$	
	TAN	$tan(x)$	
	ATAN	$tan^{-1}(x)$	
	ARCTAN	$tan^{-1}(x)$	

@ 2023 All Rights Reserved.

Table 5-1 Control system parts, *continued*

Category	Part	Description	Properties
Expression functions	ABM	no inputs, V out	EXP1...EXP4
	ABM1	1 input, V out	EXP1...EXP4
	ABM2	2 inputs, V out	EXP1...EXP4
	ABM3	3 inputs, V out	EXP1...EXP4
	ABM/I	no input, I out	EXP1...EXP4
	ABM1/I	1 input, I out	EXP1...EXP4
	ABM2/I	2 inputs, I out	EXP1...EXP4
	ABM3/I	3 inputs, I out	EXP1...EXP4

@ 2023 All Rights Reserved.

Basic Components

The basic components provide fundamental functions and in many cases, do not require specifying property values. These parts are described below.

CONST

VALUE constant value

The CONST part outputs the voltage specified by the VALUE property. This part provides no inputs and one output.

SUM

The SUM part evaluates the voltages of the two input sources, adds the two inputs together, then outputs the sum. This part provides two inputs and one output.

MULT

The MULT part evaluates the voltages of the two input sources, multiplies the two together, then outputs the product. This part provides two inputs and one output.

GAIN

GAIN constant gain value

The GAIN part multiplies the input by the constant specified by the GAIN property, then outputs the result. This part provides one input and one output.

DIFF

The DIFF part evaluates the voltage difference between two inputs, then outputs the result. This part provides two inputs and one output.

@ 2023 All Rights Reserved.

Limiters

The Limiters can be used to restrict an output to values between a set of specified ranges. These parts are described below.

LIMIT

HI	upper limit value
LO	lower limit value

The LIMIT part constrains the output voltage to a value between an upper limit (set with the HI property) and a lower limit (set with the LO property). This part takes one input and provides one output.

GLIMIT

HI	upper limit value
LO	lower limit value
GAIN	constant gain value

The GLIMIT part functions as a one-line opamp. The gain is applied to the input voltage, then the output is constrained to the limits set by the LO and HI properties. This part takes one input and provides one output.

SOFTLIMIT

HI	upper limit value
LO	lower limit value
GAIN	constant gain value
A, B, V, TANH	internal variables used to define the limiting function

@ 2023 All Rights Reserved.

The SOFTLIMIT part provides a limiting function much like the LIMIT device, except that it uses a continuous curve limiting function, rather than a discontinuous limiting function. This part takes one input and provides one output.

Caution

Besides the limiters listed above, the ABM.OLB consists of a legacy part, HILO. This part is in the library for backward compatibility and should not be used in your designs.

Chebyshev Filters

The Chebyshev filters allow filtering of the signal based on a set of frequency characteristics. The output of a Chebyshev filter depends upon the analysis being performed.

Note: PSpice A/D computes the impulse response of each Chebyshev filter used in a transient analysis during circuit read-in. This may require considerable computing time. A message is displayed on your screen indicating that the computation is in progress.

For DC and bias point, the output is simply the DC response of the filter. For AC analysis, the output for each frequency is the filter response at that frequency. For transient analysis, the output is then the convolution of the past values of the input with the impulse response of the filter. These rules follow the standard method of using Fourier transforms.

Note: To obtain a listing of the filter Laplace coefficients for each stage, choose Setup from the Analysis menu, click on Options, and enable LIST in the Options dialog box.

We recommend looking at one or more of the references cited in Frequency-domain device models on page 198, as well as some of the following references on analog filter design:

- Ghavsi, M.S. & Laker, K.R., *Modern Filter Design*, Prentice-Hall, 1981.
- Gregorian, R. & Temes, G., *Analog MOS Integrated Circuits*, Wiley-Interscience, 1986.
- Johnson, David E., *Introduction to Filter Theory*, Prentice-Hall, 1976.
- Lindquist, Claude S., *Active Network Design with Signal Filtering Applications*, Steward & Sons, 1977.
- Stephenson, F.W. (ed), *RC Active Filter Design Handbook*, Wiley, 1985.
- Van Valkenburg, M.E., *Analog Filter Design*, Holt, Rinehart & Winston, 1982.
- Williams, A.B., *Electronic Filter Design Handbook*, McGraw-Hill, 1981.

@ 2023 All Rights Reserved.

Each of the Chebyshev filter parts is described in the following pages.

LOPASS

FS	stop band frequency
FP	pass band frequency
RIPPLE	pass band ripple in dB
STOP	stop band attenuation in dB

The LOPASS part is characterized by two cutoff frequencies that delineate the boundaries of the filter pass band and stop band. The attenuation values, RIPPLE and STOP, define the maximum allowable attenuation in the pass band, and the minimum required attenuation in the stop band, respectively. The LOPASS part provides one input and one output.

The following figure shows an example of a LOPASS filter device. The filter provides a pass band cutoff of 800 Hz and a stop band cutoff of 1.2 kHz. The pass band ripple is 0.1 dB and the minimum stop band attenuation is 50 dB.

Figure 5-1 LOPASS filter part example

Assuming that the input to the filter is the voltage at net 10 and output is a voltage between nets 5 and 0, this will produce a PSpice netlist declaration like this:

```
ELOWPASS 5 0 CHEBYSHEV {V(10)} LP (800Hz 1.2kHz) .1dB 50dB
```

HIPASS

FS	stop band frequency
FP	pass band frequency
RIPPLE	pass band ripple in dB

@ 2023 All Rights Reserved.

STOP	stop band attenuation in dB

The HIPASS part is characterized by two cutoff frequencies that delineate the boundaries of the filter pass band and stop band. The attenuation values, RIPPLE and STOP, define the maximum allowable attenuation in the pass band, and the minimum required attenuation in the stop band, respectively. The HIPASS part provides one input and one output.

The following figure shows an example of a HIPASS filter device. This is a high pass filter with the pass band above 1.2 kHz and the stop band below 800 Hz.

Figure 5-2 HIPASS filter part example

Again, the pass band ripple is 0.1 dB and the minimum stop band attenuation is 50 dB. This will produce a PSpice netlist declaration like this:

```
EHIGHPASS 5 0 CHEBYSHEV {V(10)} HP (1.2kHz 800Hz) .1dB 50dB
```

BANDPASS

RIPPLE	pass band ripple in dB
STOP	stop band attenuation in dB
F0, F1, F2, F3	cutoff frequencies

The BANDPASS part is characterized by four cutoff frequencies. The attenuation values, RIPPLE and STOP, define the maximum allowable attenuation in the pass band, and the minimum required attenuation in the stop band, respectively. The BANDPASS part provides one input and one output.

@ 2023 All Rights Reserved.

The following figure shows an example of a BANDPASS filter device. This is a band pass filter with the pass band between 1.2 kHz and 2 kHz, and stop bands below 800 Hz and above 3 kHz.

Figure 5-3 BANDPASS filter part example

The pass band ripple is 0.1 dB and the minimum stop band attenuation is 50 dB. This will produce a PSpice A/D netlist declaration like this:

```
EBANDPASS 5 0 CHEBYSHEV
+ {V(10)} BP (800 1.2kHz 2kHz 3kHz) .1dB 50dB
```

BANDREJ

RIPPLE	is the pass band ripple in dB
STOP	is the stop band attenuation in dB
F0, F1, F2, F3	are the cutoff frequencies

The BANDREJ part is characterized by four cutoff frequencies. The attenuation values, RIPPLE and STOP, define the maximum allowable attenuation in the pass band, and the minimum required attenuation in the stop band, respectively. The BANDREJ part provides one input and one output.

The following figure shows an example of a BANDREJ filter device. This is a band reject (or “notch”) filter with the stop band between 1.2 kHz and 2 kHz, and pass bands below 800 Hz and above 3 kHz.

@ 2023 All Rights Reserved.

Figure 5-4 BANDREJ filter part example

The pass band ripple is 0.1 dB and the minimum stop band attenuation is 50 dB. This will produce a PSpice netlist declaration like this:

```
EBREJ 5 0 CHEBYSHEV {V(10)} BR (800Hz 1.2kHz 3kHz 2kHz) .1dB 50dB
```

Integrator and differentiator

The integrator and differentiator parts are described below.

INTEG

IC	initial condition of the integrator output
GAIN	gain value

The INTEG part implements a simple integrator. A current source/capacitor implementation is used to provide support for setting the initial condition.

DIFFER

GAIN	gain value

The DIFFER part implements a simple differentiator. A voltage source/capacitor implementation is used. The DIFFER part provides one input and one output.

Table look-up parts

TABLE and FTABLE parts provide a lookup table that is used to correlate an input and an output based on a set of data points. These parts are described below and on the following pages.

@ 2023 All Rights Reserved.

TABLE

ROW*n* is an (input, output) pair; by default, up to five triplets are allowed where *n*=1, 2, 3, 4, or 5

If more than five values are required, the part can be customized through the part editor. Insert additional row variables into the template using the same form as the first five, and add ROW*n* properties as needed to the list of properties.

The TABLE part allows the response to be defined by a table of one to five values. Each row contains an input and a corresponding output value. Linear interpolation is performed between entries.

For values outside the table's range, the device's output is a constant with a value equal to the entry with the smallest (or largest) input. This characteristic can be used to impose an upper and lower limit on the output. The TABLE part provides one input and one output.

FTABLE

ROW*n* either an (input frequency, magnitude, phase) triplet, or an (input frequency, real part, imaginary part) triplet describing a complex value; by default, up to five triplets are allowed where n=1, 2, 3, 4, or 5

If more than five values are required, the part can be customized through the part editor. Insert additional row variables into the template using the same form as the first five, and add ROW*n* properties as needed to the list of properties.

@ 2023 All Rights Reserved.

DELAY	group delay increment; defaults to 0 if left blank
R_I	table type; if left blank, the frequency table is interpreted in the (input frequency, magnitude, phase) format; if defined with any value (such as YES), the table is interpreted in the (input frequency, real part, imaginary part) format
MAGUNITS	units for magnitude where the value can be DB (decibels) or MAG (raw magnitude); defaults to DB if left blank
PHASEUNITS	units for phase where the value can be DEG (degrees) or RAD (radians); defaults to DEG if left blank

The FTABLE part is described by a table of frequency responses in either the magnitude/phase domain (R_I=) or complex number domain (R_I=YES). The entire table is read in and converted to magnitude in dB and phase in degrees.

Interpolation is performed between entries. Magnitude is interpolated logarithmically; phase is interpolated linearly. For frequencies outside the table's range, 0 (zero) magnitude is used. This characteristic can be used to impose an upper and lower limit on the output.

The DELAY property increases the group delay of the frequency table by the specified amount. The delay term is particularly useful when a frequency table device generates a non-causality warning message during a transient analysis. The warning message issues a delay value that can be assigned to the part's DELAY property for subsequent runs, without otherwise altering the table.

The output of the part depends on the analysis being done. For DC and bias point, the output is the zero frequency magnitude times the input voltage. For AC analysis, the input voltage is linearized around the bias point (similar to EVALUE and GVALUE parts, Modeling mathematical or instantaneous relationships on page 193). The output for each frequency is then the input times the gain, times the value of the table at that frequency.

For transient analysis, the voltage is evaluated at each time point. The output is then the convolution of the past values with the impulse response of the frequency response. These rules follow the standard method of using Fourier transforms. We recommend looking at one

@ 2023 All Rights Reserved.

or more of the references cited in Frequency-domain device models on page 198 for more information.

Note: The table's frequencies must be in order from lowest to highest. The TABLE part provides one input and one output.

Example

A device, ELOFILT, is used as a frequency filter. The input to the frequency response is the voltage at net 10. The output is a voltage across nets 5 and 0. The table describes a low pass filter with a response of 1 (0 dB) for frequencies below 5 kilohertz and a response of 0.001 (-60 dB) for frequencies above 6 kilohertz. The phase lags linearly with frequency. This is the same as a constant time delay. The delay is necessary so that the impulse response is causal. That is, so that the impulse response does not have any significant components before time zero. The FTABLE part in the following figure could be used:

Figure 5-5 FTABLE part

This part is characterized by the following properties:

```
ROW1 = 0Hz  0   0
ROW2 = 5kHz 0   -5760
ROW3 = 6kHz -60 -6912
DELAY =
R_I =
MAGUNITS =
PHASEUNITS =
```

Since R_I, MAGUNITS, and PHASEUNITS are undefined, each table entry is interpreted as containing frequency, magnitude value in dB, and phase values in degrees. Delay defaults to 0.

This produces a PSpice netlist declaration like this:

```
ELOFILT 5 0 FREQ {V(10)}
+0Hz 0 0
+5kHz 0 -5760
+6kHz -60 -6912
```


@ 2023 All Rights Reserved.

Since constant group delay is calculated from the values for a given table entry as:

group delay = phase / 360 / frequency

An equivalent FTABLE instance could be defined using the DELAY property. For this example, the group delay is 3.2 msec (6912 / 360 / 6k = 5760 / 360 / 6k = 3.2m). Equivalent property assignments are:

```
ROW1 = 0Hz  0   0
ROW2 = 5kHz 0   0
ROW3 = 6kHz -60 0
DELAY = 3.2ms
R_I =
MAGUNITS =
PHASEUNITS =
```

This produces a PSpice netlist declaration like this:

```
ELOFILT 5 0 FREQ {V(10)}
+0Hz 0 0
+5kHz 0 0
+6kHz -60 0
+DELAY=3.2ms
```


@ 2023 All Rights Reserved.

Laplace transform part

The LAPLACE part specifies a Laplace transform which is used to determine an output for each input value.

LAPLACE

NUM	numerator of the Laplace expression
DENOM	denominator of the Laplace expression

The LAPLACE part uses a Laplace transform description. The input to the transform is a voltage. The numerator and denominator of the Laplace transform function are specified as properties for the part.

Note: Voltages, currents, and TIME may not appear in a Laplace transform specification.

The output of the part depends on the type of analysis being done. For DC and bias point, the output is the zero frequency gain times the value of the input. The zero frequency gain is the value of the Laplace transform with s=0. For AC analysis, the output is then the input times the gain times the value of the Laplace transform. The value of the Laplace transform at a frequency is calculated by substituting $j \cdot \omega$ for s, where ω is $2\pi \cdot$frequency. For transient analysis, the output is the convolution of the input waveform with the impulse response of the transform. These rules follow the standard method of using Laplace transforms.

Example one

The input to the Laplace transform is the voltage at net 10. The output is a voltage and is applied between nets 5 and 0. For DC, the output is simply equal to the input, since the gain at s=0 is 1. The transform, $1/(1+.001 \cdot s)$, describes a simple, lossy integrator with a time constant of 1 millisecond. This can be implemented with an RC pair that has a time constant of 1 millisecond.

For AC analysis, the gain is found by substituting $j \cdot \omega$ for s. This gives a flat response out to a corner frequency of $1000/(2\pi) = 159$ hertz and a roll-off of 6 dB per octave after 159 Hz. There is also a phase shift centered around 159 Hz. In other words, the gain has both a real and an imaginary component. For transient analysis, the output is the convolution of the input waveform with the impulse response of $1/(1+.001 \cdot s)$. The impulse response is a decaying exponential with a time constant of 1 millisecond. This means that the output is the “lossy

@ 2023 All Rights Reserved.

integral" of the input, where the loss has a time constant of 1 millisecond. The LAPLACE part shown in the following figure could be used for this purpose:

Figure 5-6 LAPLACE part

The transfer function is the Laplace transform (1/[1+.001*s]). This LAPLACE part is characterized by the following properties:

```
NUM = 1
DENOM = 1 + .001*s
```

The gain and phase characteristics are shown in the following figure:

@ 2023 All Rights Reserved.

Figure 5-7 Viewing gain and phase characteristics of a lossy integrator.

This produces a PSpice netlist declaration like this:

```
ERC    5 0 LAPLACE {V(10)} = {1/(1+.001*s)}
```

Example two

The input is V(10). The output is a current applied between nets 5 and 0. The Laplace transform describes a lossy transmission line. R, L, and C are the resistance, inductance, and capacitance of the line per unit length.

If R is small, the characteristic impedance of such a line is $Z = ((R + j\cdot\omega\cdot L)/(j\cdot\omega\cdot C))^{1/2}$, the delay per unit length is $(L\ C)^{1/2}$, and the loss in dB per unit length is $23\cdot R/Z$. This could be represented by the device in the following figure:

Figure 5-8 LAPLACE part

The parameters R, L, and C can be defined in a .PARAM statement contained in a model file. (Refer to the *PSpice A/D Reference Guide* for more information about using .PARAM statements.) More useful, however, is for R, L, and C to be arguments passed into a subcircuit. This part has the following characteristics:

```
NUM = EXP(-SQRT(C*s*(R+L*s)))
DENOM = 1
```

This produces a PSpice netlist declaration like this:

```
GLOSSY 5 0 LAPLACE {V(10)} = {exp(-sqrt(C*s*(R + L*s)))}
```

The Laplace transform parts are, however, an inefficient way, in both computer time and memory, to implement a delay. For ideal delays we recommend using the transmission line part instead.

@ 2023 All Rights Reserved.

Math functions

The ABM math function parts are shown in Table 5-2. Math function parts are based on the PSpice "E" device type. Each provides one or more inputs, and a mathematical function which is applied to the input. The result is output on the output net.

Table 5-2 ABM math function parts

For this device...	Output is the...
ABS	absolute value of the input
SQRT	square root of the input
PWR	result of raising the absolute value of the input to the power specified by EXP
PWRS	result of raising the (signed) input value to the power specified by EXP
LOG	LOG of the input
LOG10	LOG_{10} of the input
EXP	result of *e* raised to the power specified by the input value (e^x where x is the input)
SIN	*sin* of the input (where the input is in radians)
COS	*cos* of the input (where the input is in radians)
TAN	*tan* of the input (where the input is in radians)
ATAN, ARCTAN	tan^{-1} of the input (where the output is in radians)

Note: Since Berkeley SPICE did not handle math functions, different SPICE implementations use different conventions for signed and unsigned power functions.

@ 2023 All Rights Reserved.

ABM expression parts

The expression parts are shown in Table 5-3. These parts can be customized to perform a variety of functions depending on your requirements. Each of these parts has a set of four expression *building block* properties of the form:

`EXP`*n*

where *n* = 1, 2, 3, or 4.

During netlist generation, the complete expression is formed by concatenating the building block expressions in numeric order, thus defining the transfer function. Hence, the first expression fragment should be assigned to the EXP1 property, the second fragment to EXP2, and so on.

Expression properties can be defined using a combination of arithmetic operators and input designators. You may use any of the standard PSpice arithmetic operators within an expression statement. You may also use the EXP*n* properties as variables to represent nets or constants.

Table 5-3 ABM expression parts

Part	Inputs	Output
ABM	none	V
ABM1	1	V
ABM2	2	V
ABM3	3	V
ABM/I	none	I
ABM1/I	1	I
ABM2/I	2	I
ABM3/I	3	I

The following examples illustrate a variety of ABM expression part applications.

@ 2023 All Rights Reserved.

Example one

Suppose you want to set an output voltage on net 4 to 5 volts times the square root of the voltage between nets 3 and 2. You could use an ABM2 part (which takes two inputs and provides a voltage output) to define a part like the one shown in the following figure:

Figure 5-9 ABM expression part example one.

In this example of an ABM device, the output voltage is set to 5 volts times the square root of the voltage between net 3 and net 2. The property settings for this part are as follows:

```
EXP1 = 5V *
EXP2 = SQRT(V(%IN2,%IN1))
```

This will produce a PSpice netlist declaration like this:

```
ESQROOT 4 0 VALUE = {5V*SQRT(V(3,2))}
```

Example two

GPSK is an oscillator for a PSK (Phase Shift Keyed) modulator. Current is pumped from net 11 through the source to net 6. Its value is a sine wave with an amplitude of 15 mA and a frequency of 10 kHz. The voltage at net 3 can shift the phase of GPSK by 1 radian/volt. Note the use of the TIME parameter in the EXP2 expression. This is the PSpice A/D internal sweep variable used in transient analysis. For any analysis other than transient, TIME = 0. This could be represented with an ABM1/I part (single input, current output) like the one shown in the following figure:

Figure 5-10 ABM expression part example two.

This part is characterized by the following properties:

@ 2023 All Rights Reserved.

```
EXP1 = 15ma * SIN(
EXP2 = 6.28*10kHz*TIME
EXP3 = + V(%IN))
```

This produces a PSpice netlist declaration like this:

```
GPSK   11 6 VALUE = {15MA*SIN(6.28*10kHz*TIME+V(3))}
```

Example three

A device, EPWR, computes the instantaneous power by multiplying the voltage across nets 5 and 4 by the current through VSENSE. Sources are controlled by expressions which may contain voltages or currents or both. The ABM2 part (two inputs, voltage output) in the following figure could represent this:

Figure 5-11 ABM expression part example three.

This part is characterized by the following properties:

```
EXP1 = V(%IN2,%IN1) *
EXP2 = I(VSENSE)
```

This produces a PSpice netlist declaration like this:

```
EPWR    3 0 VALUE = {V(5,4)*I(VSENSE)}
```

Example four

The output of a component, GRATIO, is a current whose value (in amps) is equal to the ratio of the voltages at nets 13 and 2. If V(2) = 0, the output depends upon V(13) as follows:

if V(13) = 0, output = 0
if V(13) > 0, output = MAXREAL
if V(13) < 0, output = -MAXREAL

@ 2023 All Rights Reserved.

where MAXREAL is a PSpice internal constant representing a very large number (on the order of 1e30). In general, the result of evaluating an expression is limited to MAXREAL. This is modeled with an ABM2/I (two input, current output) part like this one in 5-12.

Figure 5-12 ABM expression part example four.

This part is characterized by the following properties:

```
EXP1 = V(%IN2)/V(%IN1)
```

Note that output of GRATIO can be used as part of the controlling function. This produces a PSpice netlist declaration like this:

```
GRATIO 2 3 VALUE = {V(13)/V(2)}
```

Note: Letting a current approach ±1e30 will almost certainly cause convergence problems. To avoid this, use the limit function on the ratio to keep the current within reasonable limits.

@ 2023 All Rights Reserved.

An instantaneous device example: modeling a triode

This section provides an example of using various ABM parts to model a triode vacuum tube. The schematic of the triode subcircuit is shown in the following figure:

Figure 5-13 Triode circuit.

Assumptions: In its main operating region, the triode's current is proportional to the 3/2 power of a linear combination of the grid and anode voltages:

$$ianode = k_0*(v_g + k_1*va)^{1.5}$$

For a typical triode, k_0 = 200e-6 and k_1 = 0.12.

Looking at the upper left-hand portion of the schematic, notice the a general-purpose ABM part used to take the input voltages from anode, grid, and cathode. Assume the following associations:

- V(anode) is associated with V(%IN1)
- V(grid) is associated with V(%IN2)
- V(cathode) is associated with V(%IN3)

The expression property EXP1 then represents V(grid, cathode) and the expression property EXP2 represents 0.12[V(anode, cathode)]. When the template substitution is performed, the resulting VALUE is equivalent to the following:

V = V(grid, cathode) + 0.12*V(anode, cathode)

@ 2023 All Rights Reserved.

The part would be defined with the following characteristics:

```
EXP1 = V(%IN2,%IN3)+
EXP2 = 0.12*V(%IN1,%IN3)
```

This works for the main operating region but does not model the case in which the current stays 0 when combined grid and anode voltages go negative. We can accommodate that situation as follows by adding the LIMIT part with the following characteristics:

```
HI = 1E3
LO = 0
```

This part instance, LIMIT1, converts all negative values of v_g+.12*v_a to 0 and leaves all positive values (up to 1 kV) alone. For a more realistic model, we could have used TABLE to correctly model how the tube turns off at 0 or at small negative grid voltages.

We also need to make sure that the current becomes zero when the anode alone goes negative. To do this, we can use a DIFF device, (immediately below the ABM3 device) to monitor the difference between V(anode) and V(cathode), and output the difference to the TABLE part. The table translates all values at or below zero to zero, and all values greater than or equal to 30 to one. All values between 0 and 30 are linearly interpolated. The properties for the TABLE part are as follows:

```
ROW1 = 00
ROW2 = 301
```

The TABLE part is a simple one, and ensures that only a zero value is output to the multiplier for negative anode voltages.

The output from the TABLE part and the LIMIT part are combined at the MULT multiplier part. The output of the MULT part is the product of the two input voltages. This value is then raised to the 3/2 or 1.5 power using the PWR part. The exponential property of the PWR part is defined as follows:

```
EXP = 1.5
```

The last major component is an ABM expression component to take an input voltage and convert it into a current. The relevant ABM1/I part property looks like this:

```
EXP1 = 200E-6 * V(%IN)
```

A final step in the model is to add device parasitics. For example, a resistor can be used to give a finite output impedance. Capacitances between the grid, cathode, and anode are also needed. The lower part of the schematic shows a possible method for incorporating these

@ 2023 All Rights Reserved.

effects. To complete the example, one could add a circuit which produces the family of I-V curves shown in the following figure.

Figure 5-14 Triode subcircuit producing a family of I-V curves.

@ 2023 All Rights Reserved.

PSpice-equivalent parts

PSpice-equivalent parts respond to a differential input and have double-ended output. These parts reflect the structure of PSpice E and G devices, thus having two pins for each controlling input and the output in the part. The following table summarizes the PSpice-equivalent parts available in the part library.

Table 5-4 PSpice-equivalent parts

Category	Part	Description	Properties
Mathematical expression	EVALUE	general purpose	EXPR
	GVALUE		
	ESUM	special purpose	(none)
	GSUM		
	EMULT		
	GMULT		
Table look-up	ETABLE	general purpose	EXPR
	GTABLE		TABLE
Frequency table look-up	EFREQ	general purpose	EXPR
	GFREQ		TABLE
Laplace transform	ELAPLACE	general purpose	EXPR
	GLAPLACE		XFORM

PSpice-equivalent ABM parts can be classified as either E or G device types. The E part type provides a voltage output, and the G device type provides a current output.

The device's transfer function can contain any mixture of voltages and currents as inputs. Hence, there is no longer a division between voltage-controlled and current-controlled parts. Rather the part type is dictated only by the output requirements. If a voltage output is required, use an E part type. If a current output is necessary, use a G part type.

Note: There are no equivalent "F" or "H" part types in the part library because PSpice "F" and "H" devices do not support the ABM extensions.

Each E or G part type in the `ABM.OLB` part file is defined by a template that provides the specifics of the transfer function. Other properties in the model definition can be edited to customize the transfer function. By default, the template cannot be modified directly choosing

@ 2023 All Rights Reserved.

Properties from the Edit menu in Capture. Rather, the values for other properties (such as the expressions used in the template) are usually edited, then these values are substituted into the template. However, the part editor can be used to modify the template or designate the template as modifiable from within Capture. This way, custom parts can be created for special-purpose application.

Implementation of PSpice-equivalent parts

Although you generally use Capture to place and specify PSpice-equivalent ABM parts, it is useful to know the PSpice command syntax for “E” and “G” devices. This is especially true when creating custom ABM parts since part templates must adhere to PSpice syntax.

The general forms for PSpice “E” and “G” extensions are:

```
E <name> <connecting nodes> <ABM keyword> <ABM function>
G <name> <connecting nodes> <ABM keyword> <ABM function>
```

where

<name>	is the device name appended to the E or G device type character
<connecting nodes>	specifies the <(+ *node name*, - *node name*)> pair between which the device is connected
<ABM keyword>	specifies the form of the transfer function to be used, as one of: VALUE arithmetic expression TABLE lookup table LAPLACE Laplace transform FREQ frequency response table CHEBYSHEV Chebyshev filter characteristics
<ABM function>	specifies the transfer function as a formula or lookup table as required by the specified *<ABM keyword>*

Refer to the *PSpice A/D Reference Guide* for detailed information.

@ 2023 All Rights Reserved.

Modeling mathematical or instantaneous relationships

The instantaneous models (using VALUE and TABLE extensions to PSpice “E” and “G” devices in the part templates) enforce a direct response to the input at each moment in time. For example, the output might be equal to the square root of the input at every point in time. Such a device has no memory, or, a flat frequency response. These techniques can be used to model both linear and nonlinear responses.

Note: For AC analysis, a nonlinear device is first linearized around the bias point, and then the linear equivalent is used.

EVALUE and GVALUE parts

The EVALUE and GVALUE parts allow an instantaneous transfer function to be written as a mathematical expression in standard notation. These parts take the input signal, perform the function specified by the EXPR property on the signal, and output the result on the output pins.

In controlled sources, EXPR may contain constants and parameters as well as voltages, currents, or time. Voltages may be either the voltage at a net, such as V(5), or the voltage across two nets, such as V(4,5). Currents must be the current through a voltage source (V device), for example, I(VSENSE). Voltage sources with a value of 0 are handy for sensing current for use in these expressions.

Functions may be used in expressions, along with arithmetic operators (+, -, *, and /) and parentheses.

The EVALUE and GVALUE parts are defined, in part, by the following properties (default values are shown):

```
EVALUE
   EXPR          V(%IN+, %IN-)
GVALUE
   EXPR          V(%IN+, %IN-)
```

Sources are controlled by expressions which may contain voltages, currents, or both. The following examples illustrate customized EVALUE and GVALUE parts.

@ 2023 All Rights Reserved.

Example 1

In the example of an EVALUE device shown in the following figure, the output voltage is set to 5 volts times the square root of the voltage between pins %IN+ and %IN-.

Figure 5-15 EVALUE part example.

The property settings for this device are as follows:

```
EXPR = 5v * SQRT(V(%IN+,%IN-))
```

Example 2

Consider the device in the following figure. This device could be used as an oscillator for a PSK (Phase Shift Keyed) modulator.

Figure 5-16 GVALUE part example.

A current through a source is a sine wave with an amplitude of 15 mA and a frequency of 10 kHz. The voltage at the input pin can shift the phase by 1 radian/volt. Note the use of the TIME parameter in this expression. This is the PSpice internal sweep variable used in transient analyses. For any analysis other than transient, TIME = 0. The relevant property settings for this device are shown below:

```
EXPR = 15ma*SIN(6.28*10kHz*TIME+V(%IN+,%IN-))
```


@ 2023 All Rights Reserved.

EMULT, GMULT, ESUM, and GSUM

The EMULT and GMULT parts provide output which is based on the product of two input sources. The ESUM and GSUM parts provide output which is based on the sum of two input sources. The complete transfer function may also include other mathematical expressions.

Example 1

Consider the device as shown in the following figure:

This device computes the instantaneous power through resistor VSENSE by multiplying the current through VSENSE (converted to voltage by H1) by the voltage across VSENSE.

Figure 5-17 EMULT part example.

This device's behavior is built-in to the PSPICETEMPLATE property as follows (appears on one line):

```
TEMPLATE=E^@REFDES %OUT+ %OUT- VALUE {V(%IN1+,%IN1-)
    *V(%IN2+,%IN2-)}
```

You can use the part editor to change the characteristics of the template to accommodate additional mathematical functions, or to change the nature of the transfer function itself. For example, you may want to create a voltage divider, rather than a multiplier. This is illustrated in the following example.

@ 2023 All Rights Reserved.

Example 2

Consider the device as shown in the following figure:

G^@REFDES %OUT+ %OUT- VALUE {V(%IN1+,%IN1-)/V(%IN2+,%IN2-)}

Figure 5-18 GMULT part example.

With this device, the output is a current is equal to the ratio of the voltages at input pins 1 and input pins 2. If V(%IN2+, %IN2-) = 0, the output depends upon V(%IN1+, %IN1-) as follows:

if V(%IN1+, %IN1-) = 0, output = 0
if V(%IN1+, %IN1-) > 0, output = MAXREAL
if V(%IN1+, %IN1-) < 0, output = -MAXREAL

where MAXREAL is a PSpice internal constant representing a very large number (on the order of 1e30). In general, the result of evaluating an expression is limited to MAXREAL. Note that the output of the part can also be used as part of the controlling function.

To create this device, you would first make a new part, GDIV, based on the GMULT part. Edit the GDIV template to divide the two input values rather than multiply them.

Lookup tables (ETABLE and GTABLE)

The ETABLE and GTABLE parts use a transfer function described by a table. These device models are well suited for use with measured data.

The ETABLE and GTABLE parts are defined in part by the following properties (default values are shown):

ETABLE

@ 2023 All Rights Reserved.

TABLE (-15, -15), (15,15)
EXPR V(%IN+, %IN-)

GTABLE

TABLE (-15, -15), (15,15)
EXPR V(%IN+, %IN-)

First, EXPR is evaluated, and that value is used to look up an entry in the table. EXPR is a function of the input (current or voltage) and follows the same rules as for VALUE expressions.

The table consists of pairs of values, the first of which is an input, and the second of which is the corresponding output. Linear interpolation is performed between entries. For values of EXPR outside the table's range, the device's output is a constant with a value equal to the entry with the smallest (or largest) input. This characteristic can be used to impose an upper and lower limit on the output.

An example of a table declaration (using the TABLE property) would be the following:

```
TABLE =
+ (0, 0) (.02, 2.690E-03) (.04, 4.102E-03) (.06, 4.621E-03)
+ (.08, 4.460E-03) (.10, 3.860E-03) (.12, 3.079E-03) (.14,
+ 2.327E-03)
+ (.16, 1.726E-03) (.18, 1.308E-03) (.20, 1.042E-03) (.22,
+ 8.734E-04)
+ (.24, 7.544E-04) (.26, 6.566E-04) (.28, 5.718E-04) (.30,
+ 5.013E-04)
+ (.32, 4.464E-04) (.34, 4.053E-04) (.36, 3.781E-04) (.38,
+ 3.744E-04)
+ (.40, 4.127E-04) (.42, 5.053E-04) (.44, 6.380E-04) (.46,
+ 7.935E-04)
+ (.48, 1.139E-03) (.50, 2.605E-03) (.52, 8.259E-03) (.54,
+ 2.609E-02)
+ (.56, 7.418E-02) (.58, 1.895E-01) (.60, 4.426E-01)
```


@ 2023 All Rights Reserved.

Frequency-domain device models

Frequency-domain models (ELAPLACE, GLAPLACE, EFREQ, and GFREQ) are characterized by output that depends on the current input as well as the input history. The relationship is therefore non-instantaneous. For example, the output may be equal to the integral of the input over time. In other words, the response depends upon frequency.

During AC analysis, the frequency response determines the complex gain at each frequency. During DC analysis and bias point calculation, the gain is the zero-frequency response. During transient analysis, the output of the device is the convolution of the input and the impulse response of the device.

Moving back and forth between the time and frequency-domains can cause surprising results. Often the results are quite different than what one would intuitively expect. For this reason, we strongly recommend familiarity with a reference on Fourier and Laplace transforms. A good one is:

- R. Bracewell, *The Fourier Transform and Its Applications*, McGraw-Hill, Revised Second Edition (1986)

We also recommend familiarity with the use of transforms in analyzing linear systems. Some references on this subject:

- W. H. Chen, *The Analysis of Linear Systems*, McGraw-Hill (1962)
- J. A. Aseltine, *Transform Method in Linear System Analysis*, McGraw-Hill (1958)

Laplace transforms (LAPLACE)

The ELAPLACE and GLAPLACE parts allow a transfer function to be described by a Laplace transform function. The ELAPLACE and GLAPLACE parts are defined, in part, by the following properties (default values are shown):

```
ELAPLACE

    EXPR        V(%IN+, %IN-)
    XFORM         1/s

GLAPLACE

    EXPR        V(%IN+, %IN-)
    XFORM         1/s
```

The LAPLACE parts use a Laplace transform description. The input to the transform is the value of EXPR, where EXPR follows the same rules as for VALUE expressions (see EVALUE and GVALUE parts on page 193). XFORM is an expression in the Laplace variable, s. It

@ 2023 All Rights Reserved.

follows the rules for standard expressions as described for VALUE expressions with the addition of the s variable.

Note: Voltages, currents, and TIME cannot appear in a Laplace transform.

The output of the device depends on the type of analysis being done. For DC and bias point, the output is simply the zero frequency gain times the value of EXPR. The zero frequency gain is the value of XFORM with s = 0. For AC analysis, EXPR is linearized around the bias point (similar to the VALUE parts). The output is then the input times the gain of EXPR times the value of XFORM. The value of XFORM at a frequency is calculated by substituting j·w for s, where w is 2p·frequency. For transient analysis, the value of EXPR is evaluated at each time point. The output is then the convolution of the past values of EXPR with the impulse response of XFORM. These rules follow the standard method of using Laplace transforms. We recommend looking at one or more of the references cited in Frequency-domain device models on page 198 for more information.

Example

The input to the Laplace transform is the voltage across the input pins, or V(%IN+, %IN-). The EXPR property may be edited to include constants or functions, as with other parts. The transform, 1/(1+.001·s), describes a simple, lossy integrator with a time constant of 1 millisecond. This can be implemented with an RC pair that has a time constant of 1 millisecond.

Using the part editor, you would define the XFORM and EXPR properties as follows:

```
XFORM = 1/(1+.001*s)
EXPR = V(%IN+, %IN-)
```

The default template remains (appears on one line):

```
TEMPLATE= E^@REFDES %OUT+ %OUT- LAPLACE {@EXPR}= (@XFORM)
```

After netlist substitution of the template, the resulting transfer function would become:

```
V(%OUT+, %OUT-) = LAPLACE {V(%IN+, %IN-)}= (1/1+.001*s))
```

The output is a voltage and is applied between pins %OUT+ and %OUT-. For DC, the output is simply equal to the input, since the gain at s = 0 is 1.

For AC analysis, the gain is found by substituting $j \cdot \omega$ for s. This gives a flat response out to a corner frequency of $1000/(2\pi) = 159$ Hz and a roll-off of 6 dB per octave after 159 Hz. There is also a phase shift centered around 159 Hz. In other words, the gain has both a real and an imaginary component. The gain and phase characteristic is the same as that shown for the equivalent control system part example using the LAPLACE part (see Figure 5-7 on page 182).

@ 2023 All Rights Reserved.

For transient analysis, the output is the convolution of the input waveform with the impulse response of 1/(1+.001·s). The impulse response is a decaying exponential with a time constant of 1 millisecond. This means that the output is the “lossy integral” of the input, where the loss has a time constant of 1 millisecond.

This will produce a PSpice A/D netlist declaration similar to:

```
ERC 5 0 LAPLACE {V(10)} = {1/(1+.001*s)}
```

Frequency response tables

The EFREQ and GFREQ parts are described by a table of frequency responses in either the magnitude/phase domain or complex number domain. The entire table is read in and converted to magnitude in dB and phase in degrees. Interpolation is performed between entries. Phase is interpolated linearly; magnitude is interpolated logarithmically. For frequencies outside the table’s range, 0 (zero) magnitude is used.

Note: You can also use the equivalent EFREQ_FILE and GFREQ_FILE parts to supply frequency tables in a CSV file.

EFREQ and GFREQ properties are defined as follows:

EXPR	Value used for table lookup; defaults to V(%IN+, %IN-) if left blank.
TABLE	Series of either (input frequency, magnitude, phase) triplets, or (input frequency, real part, imaginary part) triplets describing a complex value; defaults to (0,0,0) (1Meg,-10,90) if left blank.
DELAY	Group delay increment; defaults to 0 if left blank.
R_I	Table type; if left blank, the frequency table is interpreted in the (input frequency, magnitude, phase) format; if defined with any value (such as YES), the table is interpreted in the (input frequency, real part, imaginary part) format.

@ 2023 All Rights Reserved.

MAGUNITS	Units for magnitude where the value can be DB (decibels) or MAG (raw magnitude); defaults to DB if left blank.
PHASEUNITS	Units for phase where the value can be DEG (degrees) or RAD (radians); defaults to DEG if left blank.

The DELAY property increases the group delay of the frequency table by the specified amount. The delay term is particularly useful when an EFREQ or GFREQ device generates a non-causality warning message during a transient analysis. The warning message issues a delay value that can be assigned to the part's DELAY property for subsequent runs, without otherwise altering the table.

The output of the device depends on the analysis being done. For DC and bias point, the output is simply the zero frequency magnitude times the value of EXPR. For AC analysis, EXPR is linearized around the bias point (similar to EVALUE and GVALUE parts). The output for each frequency is then the input times the gain of EXPR times the value of the table at that frequency. For transient analysis, the value of EXPR is evaluated at each time point. The output is then the convolution of the past values of EXPR with the impulse response of the frequency response. These rules follow the standard method of using Fourier transforms. We recommend looking at one or more of the references cited in Frequency-domain device models on page 198 for more information.

Note: The table's frequencies must be in order from lowest to highest.

The following figure shows an EFREQ device used as a low pass filter.

Figure 5-19 EFREQ part example.

The input to the frequency response is the voltage across the input pins. The table describes a low pass filter with a response of 1 (0 dB) for frequencies below 5 kilohertz and a response

@ 2023 All Rights Reserved.

of .001 (-60 dB) for frequencies above 6 kilohertz. The output is a voltage across the output pins.

This part is defined by the following properties:

```
TABLE = (0, 0, 0) (5kHz, 0, -5760) (6kHz, -60, -6912)
DELAY =
R_I =
MAGUNITS =
PHASEUNITS =
```

Since R_I, MAGUNITS, and PHASEUNITS are undefined, each table entry is interpreted as containing frequency, magnitude value in dB, and phase values in degrees. Delay defaults to 0.

The phase lags linearly with frequency meaning that this table exhibits a constant time (group) delay. The delay is necessary so that the impulse response is causal. That is, so that the impulse response does not have any significant components before time zero.

The constant group delay is calculated from the values for a given table entry as follows:

group delay = phase / 360 / frequency

For this example, the group delay is 3.2 msec
(6912 / 360 / 6k = 5760 / 360 / 6k = 3.2m). An alternative specification for this table could be:

```
TABLE = (0, 0, 0) (5kHz, 0, 0) (6kHz, -60, 0)
DELAY = 3.2ms
R_I =
MAGUNITS =
PHASEUNITS =
```

This produces a PSpice netlist declaration like this:

```
ELOWPASS 5 0 FREQ {V(10)} = (0,0,0) (5kHz,0,0) (6kHz-60,0)
+ DELAY = 3.2ms
```

Describing Frequency Table Using a CSV File

The `FREQ_FILE` property of EFREQ_FILE and GFREQ_FILE can be used to specify the frequency table in a CSV file.

The frequency, real, and imaginary values can be provided in a CSV file and the file can be specified using the `FREQ_FILE` property using the following syntaxes:

```
E1 node1 node2 FREQ {V(node3,node4)}= R_I   FREQ_FILE="<path to csv>"

G1 node1 node2 FREQ {V(node3,node4) }= R_I   FREQ_FILE="<path
to csv>"
```


@ 2023 All Rights Reserved.

Frequency, magnitude, and phase values can also be provided in a CSV file using the following syntaxes:

```
E1 node1 node2 FREQ {V(node3,node4)}= db deg FREQ_FILE=”<path to
csv>"

E1 node1 node2 FREQ {V(node3,node4)}= mag deg FREQ_FILE=”<path to
csv>"

E1 node1 node2 FREQ {V(node3,node4)}= db rad   FREQ_FILE=”<path to
csv>"

E1 node1 node2 FREQ {V(node3,node4)}= mag rad FREQ_FILE=”<path to
csv>"
```

The supported units for magnitude are dB or a raw form. The phase can be in degrees or radians. If units are not specified, dB is used for magnitude and degree is used for phase by default.

Cautions and Recommendations for Simulation and Analysis

Instantaneous Device Modeling

During AC analysis, nonlinear transfer functions are handled the same way as other nonlinear parts: each function is linearized around the bias point and the resulting small-signal equivalent is used.

Consider the voltage multiplier (mixer) shown in the following figure.

Figure 5-20 Voltage multiplier circuit (mixer).

@ 2023 All Rights Reserved.

This circuit has the following characteristics:

```
Vin1:        DC=0v   AC=1v
Vin2:        DC=0v   AC=1v
```

where the output on net 3 is V(1)*V(2).

During AC analysis, V(3) = 0 due to the 0 volts bias point voltage on nets 1, 2, and 3. The small-signal equivalent therefore has 0 gain (the derivative of V(1)*V(2) with respect to both V(1) and V(2) is 0 when V(1)=V(2)=0). So, the output of the mixer during AC analysis will be 0 regardless of the AC values of V(1) and V(2).

Another way of looking at this is that a mixer is a nonlinear device and AC analysis is a linear analysis. The output of the mixer has 0 amplitude at the fundamental. (Output is nonzero at DC and twice the input frequency, but these are not included in a linear analysis.)

If you need to analyze nonlinear functions, such as a mixer, use transient analysis. Transient analysis solves the full, nonlinear circuit equations. It also allows you to use input waveforms with different frequencies (for example, VIN1 could be 90 MHz and VIN2 could be 89.8 MHz). AC analysis does not have this flexibility, but in return it uses much less computer time.

Frequency-domain Parts

Some caution is in order when moving between frequency and time domains. This section discusses several points that are involved in the implementation of frequency-domain parts. These discussions all involve the transient analysis, since both the DC and AC analyses are straightforward.

The first point is that there are limits on the maximum values and on the resolution of both time and frequency. These are related: the frequency resolution is the inverse of the maximum time and vice versa. The maximum time is the length of the transient analysis, TSTOP. Therefore, the frequency resolution is 1/TSTOP.

Laplace Transforms

For Laplace transforms, PSpice A/D starts off with initial bounds on the frequency resolution and the maximum frequency determined by the transient analysis parameters as follows. The frequency resolution is initially set below the theoretical limit to (.25/TSTOP) and is then made as large as possible without inducing sampling errors. The maximum frequency has an initial upper bound of (1/(RELTOL*TMAX)), where TMAX is the transient analysis Step Ceiling value, and RELTOL is the relative accuracy of all calculated voltages and currents. If a Step Ceiling value is not specified, PSpice A/D uses the Transient Analysis Print Step, TSTEP, instead.

@ 2023 All Rights Reserved.

Note: TSTOP, TMAX, and TSTEP values are configured using Transient on the Setup menu. The RELTOL property is set using Options on the Setup menu.

PSpice A/D then attempts to reduce the maximum frequency by searching for the frequency at which the response has fallen to RELTOL times the maximum response. For instance, for the transform:

$$1/(1+s)$$

the maximum response, 1.0, is at $s = j\cdot\omega = 0$ (DC). The cutoff frequency used when RELTOL=.001, is approximately $1000/(2\pi) = 159$ Hz. At 159 Hz, the response is down to .001 (down by 60 db). Since some transforms do not have such a limit, there is also a limit of 10/RELTOL times the frequency resolution, or 10/(RELTOL·TSTOP). For example, consider the transform:

$$e^{-0.001\cdot s}$$

This is an ideal delay of 1 millisecond and has no frequency cutoff. If TSTOP = 10 milliseconds and RELTOL=.001, then PSpice A/D imposes a frequency cutoff of 10 MHz. Since the time resolution is the inverse of the maximum frequency, this is equivalent to saying that the delay cannot resolve changes in the input at a rate faster than .1 microseconds. In general, the time resolution will be limited to RELTOL·TSTOP/10.

A final computational consideration for Laplace parts is that the impulse response is determined by means of an FFT on the Laplace expression. The FFT is limited to 8192 points to keep it tractable, and this places an additional limit on the maximum frequency, which may not be greater than 8192 times the frequency resolution.

If your circuit contains many Laplace parts which can be combined into a more complex single device, it is generally preferable to do this. This saves computation and memory since there are fewer impulse responses. It also reduces the number of opportunities for numerical artifacts that might reduce the accuracy of your transient analyses.

Laplace transforms can contain poles in the left half-plane. Such poles will cause an impulse response that increases with time instead of decaying. Since the transient analysis is always for a finite time span, PSpice A/D does not have a problem calculating the transient (or DC) response. However, such poles will make the actual device oscillate.

Non-causality and Laplace Transforms

PSpice A/D applies an inverse FFT to the Laplace expression to obtain an impulse response, and then convolves the impulse response with the dependent source input to obtain the output. Some common impulse responses are inherently non-causal. This means that the

@ 2023 All Rights Reserved.

convolution must be applied to both past and future samples of the input in order to properly represent the inverse of the Laplace expression.

For example, the expression {S} corresponds to differentiation in the time domain. The impulse response for {S} is an impulse pair separated by an infinitesimal distance in time. The impulses have opposite signs, and are situated one in the infinitesimal past, the other in the infinitesimal future. In other words, convolution with this corresponds to applying a finite-divided difference in the time domain.

The problem with this for PSpice A/D is that the simulator only has the present and past values of the simulated input, so it can only apply half of the impulse pair during convolution. This will obviously not result in time-domain differentiation. PSpice A/D can detect, but not fix this condition, and issues a non-causality warning message when it occurs. The message tells what percentage of the impulse response is non-causal, and how much delay would need to be added to slide the non-causal part into a causal region. {S} is theoretically 50% non-causal. Non-causality on the order of 1% or less is usually not critical to the simulation results.

You can delay {S} to keep it causal, but the separation between the impulses is infinitesimal. This means that a very small time step is needed. For this reason, it is usually better to use a macromodel to implement differentiation.

Here are some guidelines:

- In the case of a Laplace device (ELAPLACE), multiply the Laplace expression by *e* to the (-*s* * <the suggested delay>).
- In the case of a frequency table (EFREQ or GFREQ), do either of the following:
  - Specify the table with
    `DELAY=<the suggested delay>`.
  - Compute the delay by adding a phase shift.

Chebyshev Filters

All of the considerations given above for Laplace parts also apply to Chebyshev filter parts. However, PSpice A/D also attempts to deal directly with inaccuracies due to sampling by applying Nyquist criteria based on the highest filter cutoff frequency. This is done by checking the value of TMAX. If TMAX is not specified it is assigned a value, or if it is specified, it may be reduced.

For low pass and band pass filters, TMAX is set to (0.5/FS), where FS is the stop band cutoff in the case of a low pass filter, or the upper stop band cutoff in the case of a band pass filter.

@ 2023 All Rights Reserved.

For high pass and band reject filters, there is no clear way to apply the Nyquist criterion directly, so an additional factor of two is thrown in as a safety margin. Thus, TMAX is set to (0.25/FP), where FP is the pass band cutoff for the high pass case or the upper pass band cutoff for the band reject case. It may be necessary to set TMAX to something smaller if the filter input has significant frequency content above these limits.

Frequency Tables

For frequency response tables, the maximum frequency is twice the highest value. It will be reduced to 10/(RELTOL·TSTOP) or 8192 times the frequency resolution if either value is smaller.

The frequency resolution for frequency response tables is taken to be either the smallest frequency increment in the table or the fastest rate of phase change, whichever is least. PSpice A/D then checks to see if it can be loosened without inducing sampling errors.

Trading Off Computer Resources for Accuracy

There is a significant trade-off between accuracy and computation time for parts modeled in the frequency domain. The amount of computer time and memory scale approximately inversely to RELTOL. Therefore, if you can use RELTOL=.01 instead of the default .001, you will be ahead. However, this will not adversely affect the impulse response. You may also wish to vary TMAX and TSTOP, since these also come into play.

Since the trade-off issues are fairly complex, it is advisable to first simulate a small test circuit containing only the frequency-domain device, and then after proper validation, proceed to incorporate it in your larger design. The PSpice A/D defaults will be appropriate most of the time if accuracy is your main concern, but it is still worth checking.

Note: Do not set RELTOL to a value above 0.05. This can seriously compromise the accuracy of your simulation.

@ 2023 All Rights Reserved.

Basic Controlled Sources

As with basic SPICE, PSpice A/D has basic controlled sources derived from the standard SPICE E, F, G, and H devices. Table 5-5 summarizes the linear controlled source types provided in the standard part library.

Table 5-5 Basic controlled sources in ANALOG.OLB

Device type	Part name
Controlled Voltage Source (PSpice A/D E device)	E
Current-Controlled Current Source (PSpice A/D F device)	F
Controlled Current Source (PSpice A/D G device)	G
Current-Controlled Voltage Source (PSpice A/D H device)	H

Creating Custom ABM Parts

Create a custom part when you need a controlled source that is not provided in the special purpose set or that is more elaborate than you can build with the general purpose parts (with multiple controlling inputs, for example). Refer to your *OrCAD X Capture User Guide* for a description of how to create a custom part.

The transfer function can be built into the part two different ways:

- directly in the PSPICETEMPLATE definition.
- by defining the part's EXPR and related properties (if any).

The PSpice A/D syntax for declaring E and G devices can help you form a PSPICETEMPLATE definition. Refer to the *PSpice A/D Reference Guide* for more information about E and G devices.

@ 2023 All Rights Reserved.

6

Digital Device Modeling

The standard part libraries contain a comprehensive set of digital parts in many different technologies. Each digital part is described electrically by a digital device model in the form of a subcircuit definition stored in a model library. The corresponding subcircuit name is defined by the part's MODEL attribute value. Other attributes—MNTYMXDLY, IO_LEVEL, and the PSPICEDEFAULTNET set—are passed to the subcircuit, thus providing a high-level means for influencing the behavior of the digital device model.

Generally, the digital parts provided in the part libraries are satisfactory for most circuit designs. However, if your design requires digital parts that are not already provided in the PSpice A/D part and model libraries, you need to define digital device models corresponding to the new digital parts.

A complete digital device model has three main characteristics:

- Functional behavior: described by the gate-level and behavioral digital primitives comprising the subcircuit.
- I/O behavior: described by the I/O model, interface subcircuits, and power supplies related to a logic family.
- Timing behavior: described by one or more timing models, pin-to-pin delay primitives, or constraint checker primitives.

These characteristics are described in this chapter with a running example demonstrating the use of gate-level primitives.

@ 2023 All Rights Reserved.

Functional Behavior

A digital device model's functional behavior is defined by one or more interconnected digital primitives. Typically, a logic diagram in a data book can be implemented directly using the primitives provided by PSpice A/D. The table provides a summary of the digital primitives.

Table 6-1 Digital primitives summary

Type	Description
Standard gates	
BUF	buffer
INV	inverter
AND	AND gate
NAND	NAND gate
OR	OR gate
NOR	NOR gate
XOR	exclusive OR gate
NXOR	exclusive NOR gate
BUFA	buffer array
INVA	inverter array
ANDA	AND gate array
NANDA	NAND gate array
ORA	OR gate array
NORA	NOR gate array
XORA	exclusive OR gate array
NXORA	exclusive NOR gate array
AO	AND-OR compound gate
OA	OR-AND compound gate
AOI	AND-NOR compound gate
OAI	OR-NAND compound gate

@ 2023 All Rights Reserved.

Table 6-1 Digital primitives summary

Type	Description
Tristate gates	
BUF3	buffer
INV3	inverter
AND3	AND gate
NAND3	NAND gate
OR3	OR gate
NOR3	NOR gate
XOR3	exclusive OR gate
NXOR3	exclusive NOR gate
BUF3A	buffer array
INV3A	inverter array
AND3A	AND gate array
NAND3A	NAND gate array
OR3A	OR gate array
NOR3A	NOR gate array
XOR3A	exclusive OR gate array
NXOR3A	exclusive NOR gate array
Bidirectional transfer gates	
NBTG	N-channel transfer gate
PBTG	P-channel transfer gate
Flip-flops and latches	

@ 2023 All Rights Reserved.

Table 6-1 Digital primitives summary

Type	Description
JKFF	J-K, negative-edge triggered
DFF	D-type, positive-edge triggered
SRFF	S-R gated latch
DLTCH	D gated latch
Pullup/pulldown resistors	
PULLUP	pullup resistor array
PULLDN	pulldown resistor array
Delay lines	
DLYLINE	delay line
Programmable logic arrays	
PLAND	AND array
PLOR	OR array
PLXOR	exclusive OR array
PLNAND	NAND array
PLNOR	NOR array
PLNXOR	exclusive NOR array
PLANDC	AND array, true and complement
PLORC	OR array, true and complement
PLXORC	exclusive OR array, true and complement
PLNANDC	NAND array, true and complement
PLNORC	NOR array, true and complement
PLNXORC	exclusive NOR array, true and complement

@ 2023 All Rights Reserved.

Table 6-1 Digital primitives summary

Type	Description
Memory	
ROM	read-only memory
RAM	random access read-write memory
Multi-Bit A/D & D/A Converters	
ADC	multi-bit A/D converter
DAC	multi-bit D/A converter
Behavioral	
LOGICEXP	logic expression
PINDLY	pin-to-pin delay
CONSTRAINT	constraint checking

The format for digital primitives is similar to that for analog devices. One difference is that most digital primitives require two models instead of one:

- The *timing model*, which specifies propagation delays and timing constraints such as setup and hold times.
- The *I/O model*, which specifies information specific to the device's input/output characteristics.

The reason for having two models is that, while timing information is specific to a device, the input/output characteristics are specific to a whole logic family. Thus, many devices in the same family reference the same I/O model, but each device has its own timing model.

The Figure 6-1 on page 215 presents an overview of a digital device definition in terms of its primitives and underlying model attributes.

Digital Primitive Syntax

The general digital primitive format is as follows:

```
U<name> <primitive type> [( <parameter value>* )]
+ <digital power node> <digital ground node>
+ <node>*
```


@ 2023 All Rights Reserved.

```
+ <Timing Model name> <I/O Model name>
+ [MNTYMXDLY=<delay select value>]
+ [IO_LEVEL=<interface subckt select value>]
```

where

<primitive type> [(*<parameter value>**)]

is the type of digital device, such as NAND, JKFF, or INV. It is followed by zero or more parameters specific to the primitive type, such as number of inputs. The number and meaning of the parameters depends on the primitive type.

<digital power node> <digital ground node>

are the nodes used by the interface subcircuits which connect analog nodes to digital nodes or vice versa.

*<node>**

is one or more input and output nodes. The number of nodes depends on the primitive type and its parameters. Analog devices, digital devices, or both may be connected to a node. If a node has both analog and digital connections, then PSpice A/D automatically inserts an interface subcircuit to translate between digital output states and voltages.

<Timing model name>

is the name of a timing model that describes the device's timing characteristics, such as propagation delay and setup and hold times. Each timing parameter has a minimum, typical, and maximum value which may be selected during analysis setup.

This type of Timing model and its parameters are specific to each primitive type and are discussed in the *PSpice A/D Reference Guide*.

<I/O model name>

is the name of an I/O model that describes the device's loading and driving characteristics. I/O models also contain the names of up to four DtoA and AtoD interface subcircuits, which are automatically called by PSpice A/D to handle analog/digital interface nodes.

Note that some digital primitives, such as pullups, do not have timing models.

@ 2023 All Rights Reserved.

Figure 6-1 Elements of a digital device definition

MNTYMXDLY

@ 2023 All Rights Reserved.

is an optional device parameter that selects either the minimum, typical, or maximum delay values from the device's timing model. If not specified, MNTYMXDLY defaults to 0. Valid values are:

0 = the current value of the circuit-wide DIGMNTYMX option (default=2)

1 = minimum

2 = typical

3 = maximum

4 = worst-case timing (min-max)

IO_LEVEL

is an optional device parameter that selects one of the four AtoD or DtoA interface subcircuits from the device's I/O model. PSpice A/D calls the selected subcircuit automatically in the event a node connecting to the primitive also connects to an analog device. If not specified, IO_LEVEL defaults to 0. Valid values are:

0 = the current value of the circuit-wide DIGIOLVL option (default=1)

1 = AtoD1/DtoA1

2 = AtoD2/DtoA2

3 = AtoD3/DtoA3

4 = AtoD4/DtoA4

Following are some simple examples of "U" device declarations:

```
U1 NAND(2) $G_DPWR $G_DGND 1 2 10 D0_GATE IO_DFT
U2 JKFF(1) $G_DPWR $G_DGND 3 5 200 3 3 10 2 D_293ASTD
+ IO_STD
U3 INV $G_DPWR $G_DGND IN OUT D_INV IO_INV MNTYMXDLY=3
+ IO_LEVEL=2
```

For example, the 74393 part could be defined as a subcircuit composed of "U" devices as shown.

```
.subckt 74393  A CLR QA QB QC QD
+  optional: DPWR=$G_DPWR DGND=$G_DGND
+  params: MNTYMXDLY=0 IO_LEVEL=0
UINV inv DPWR DGND
```


@ 2023 All Rights Reserved.

```
+  CLR   CLRBAR D0_GATE IO_STD
+  IO_LEVEL={IO_LEVEL}
U1 jkff(1) DPWR DGND
+  $D_HI CLRBAR A   $D_HI $D_HI
+  QA_BUF $D_NC D_393_1 IO_STD
+  MNTYMXDLY={MNTYMXDLY}
+  IO_LEVEL={IO_LEVEL}
U2 jkff(1) DPWR DGND
+  $D_HI CLRBAR QA_BUF   $D_HI $D_HI
+  QB_BUF $D_NC D_393_2 IO_STD
+  MNTYMXDLY={MNTYMXDLY}
U3 jkff(1) DPWR DGND
+  $D_HI CLRBAR QB_BUF   $D_HI $D_HI
+  QC_BUF $D_NC D_393_2 IO_STD
+  MNTYMXDLY={MNTYMXDLY}
U4 jkff(1) DPWR DGND
+  $D_HI CLRBAR QC_BUF   $D_HI $D_HI
+  QD_BUF $D_NC D_393_3 IO_STD
+  MNTYMXDLY={MNTYMXDLY}
UBUFF bufa(4) DPWR DGND
+  QA_BUF QB_BUF QC_BUF QD_BUF
+  QA QB QC QD D_393_4 IO_STD
+  MNTYMXDLY={MNTYMXDLY}
+  IO_LEVEL={IO_LEVEL}
.ends
```

When adding digital parts to the part libraries, you must create corresponding digital device models by connecting U devices in a subcircuit definition similar to the one shown above. You should save these in your own custom model library, which you can then configure for use with a given design.

Related Topics

- Timing Model
- Input/Output Model

@ 2023 All Rights Reserved.

Timing Characteristics

A digital device model's timing behavior can be defined in one of two ways:

- Most primitives have an associated Timing model, in which propagation delays and timing constraints (such as setup/hold times) are specified. This method is used when it is easy to partition delays among individual primitives; typically when the number of primitives is small.
- Use the PINDLY and CONSTRAINT primitives, which can directly model pin-to-pin delays and timing constraints for the whole device model. With this method, all other functional primitives operate in zero delay. Refer to the *PSpice A/D Reference Guide* for a detailed discussion on these two primitives.

In addition to explicit propagation delays, other factors, such as output loads, can affect the total propagation delay through a device.

Timing Model

With the exception of the PULLUP, PULLDN, and PINDLY devices, all digital primitives have a Timing model which provides timing parameters to the simulator. The Timing model for each primitive type is unique. That is, the model name and the parameters that can be defined for that model vary with the primitive type.

Within a Timing model, there may be one or more types of parameters:

- Propagation delays (TP)
- Setup times (TSU)
- Hold times (TH)
- Pulse widths (TW)
- Switching times (TSW)

Each parameter is further divided into three values: minimum (MN), typical (TY), and maximum (MX). For example, the typical low-to-high propagation delay on a gate is specified as the parameter TPLHTY. The minimum data-to-clock setup time on a flip-flop is specified as the parameter TSUDCLKMN.

Several timing models are used by digital device 74393 from the model libraries. One of them, D_393_1, is shown here for an edge-triggered flip-flop.

```
.model D_393_1 ueff (
+  tppcqhlty=18ns   tppcqhlmx=33ns
+  tpclkqlhty=6ns   tpclkqlhmx=14ns
```


@ 2023 All Rights Reserved.

```
+  tpclkqhlty=7ns   tpclkqhlmx=14ns
+  twclkhmn=20ns    twclklmn=20ns
+  twpclmn=20ns     tsudclkmn=25ns
+  )
```

When creating your own digital device models, you can create Timing models like these for the primitives you are using. PSpice A/D recommends that you save these in your own custom model library, which you can then configure for use with a given design.

One or more parameters may be missing from the Timing model definition. Data books do not always provide all three (minimum, typical, and maximum) timing specifications. The way the simulator handles missing parameters depends on the type of parameter.

For a description of Timing model parameters, see the specific primitive type under U devices in the *PSpice A/D Reference Guide*.

Treatment of Unspecified Propagation Delays

Often, only the typical and maximum delays are specified in data books. If, in this case, the simulator were to assume that the unspecified minimum delay defaults to zero, the logic in certain circuits could break down.

For this reason, the simulator provides two configurable options, DIGMNTYSCALE and DIGTYMXSCALE, which are used to extrapolate unspecified propagation delays in the Timing models.

Note: This discussion applies only to propagation delay parameters (TP). All other timing parameters, such as setup/hold times and pulse widths are handled differently, and are discussed in the following section.

DIGMNTYSCALE

This option computes the minimum delay when a typical delay is known, using the formula:

$$TP\mathit{xx}MN = DIGMNTYSCALE \cdot TP\mathit{xx}TY$$

DIGMNTYSCALE defaults to the value 0.4, or 40% of the typical delay. Its value must be between 0.0 and 1.0.

DIGTYMXSCALE

This option computes the maximum delay from a typical delay, using the formula

$$TP\mathit{xx}MX = DIGTYMXSCALE \cdot TP\mathit{xx}TY$$

@ 2023 All Rights Reserved.

DIGTYMXSCALE defaults to the value 1.6. Its value must be greater than 1.0.

When a typical delay is unspecified, its value is derived from the minimum and/or maximum delays, in one of the following ways. If both the minimum and maximum delays are known, the typical delay is the average of these two values. If only the minimum delay is known, the typical delay is derived using the value of the DIGMNTYSCALE option. Likewise, if only the maximum delay is specified, the typical delay is derived using DIGTYMXSCALE. Obviously, if no values are specified, all three delays will default to zero.

Treatment of Unspecified Timing Constraints

The remaining timing constraint parameters are handled differently than the propagation delays. Often, data books state pulse widths, setup times, and hold times as a minimum value. These parameters do not lend themselves to the extrapolation method used for propagation delays.

Instead, when one or more timing constraints are omitted, the simulator uses the following steps to fill in the missing values:

- If the minimum value is omitted, it defaults to zero.
- If the maximum value is omitted, it takes on the typical value if one was specified, otherwise it takes on the minimum value.
- If the typical value is omitted, it is computed as the average of the minimum and maximum values.

Propagation Delay Calculation

The timing characteristics of digital primitives are determined by both the timing models and the I/O models. Timing models specify propagation delays and timing constraints such as setup and hold times. I/O models specify input and output loading, driving resistances, and switching times.

When a device's output connects to another digital device, the total propagation delay through a device is determined by adding the loading delay (on the output terminal) to the delay specified in the device's timing model. Loading delay is calculated from the total load on the output and the device's driving resistances. The total load on an output is found by summing the output and input loads (OUTLD and INLD in the I/O model) of all devices connected to that output. This total load, combined with the device's driving resistances (DRVL and DRVH in the I/O model), allows the loading delay to be calculated:

$$\text{Loading delay} = R_{DRIVE} \cdot C_{TOTAL} \cdot \ln(2)$$

@ 2023 All Rights Reserved.

The loading delay is calculated for each output terminal of every device before the simulation begins. The total propagation delay is easily calculated during the simulation by adding the pre-calculated loading delay to the device's timing delay. However, for any individual timing delay specification (e.g., TPLH) having a value of 0, the loading delay *is not used*.

When outputs connect to analog devices, the propagation delay is reduced by the switching times specified in the I/O model.

Inertial and Transport Delay

The simulator uses two different types of internal delay functions when simulating the digital portion of the circuit: *inertial delay* and *transport delay*. The application of these concepts is embodied within the implementation of the digital primitives within the simulator. Therefore, they are not user-selectable.

Inertial Delay

The simulation of a device may be described as the application of some *stimulus* (S) to a *function* (F) and predicting the *response* (R).

If this device is electrical in nature, application of the stimulus implies that energy will be imparted to the device to cause it to change state. The amount of such energy is a function of the signal's amplitude and duration. If the stimulus is applied to the device for a length of time that is too short, the device will not switch. The minimum duration required for an input change to have an effect on a device's output state is called the *inertial delay* of the device. For digital simulation, all delay parameters specified in timing models are considered inertial, with the exception of the delay line primitive, DLYLINE.

To model the noise immunity behavior of digital devices correctly, the TPWRT (pulse width rejection threshold) parameter can be set in the digital device's I/O model. When *pulse width* ≥ TPWRT and *pulse width* < *propagation delay*, then the device generates either a 0-R-0, 1-F-1, or an X pulse.

@ 2023 All Rights Reserved.

This example shows normal operation in which a pulse of 20 nsec width is applied to a BUF primitive having propagation delays of 10 nsec. TPWRT is not set.

The same device with a short pulse applied produces no output change.

However, if TPWRT is assigned a numerical value (1 or 2 for this example), then the device outputs a glitch.

Transport Delay

The delay line primitive is the only simulator model that can propagate any width pulse applied to its input. Its function is to skew the applied stimulus by some constant time value. For example:

See the DLYLINE digital primitive in the *PSpice A/D Reference Guide.*

@ 2023 All Rights Reserved.

Input/Output Characteristics

A digital device model's input/output characteristics are defined by the I/O model that it references. Some characteristics, such as output drive resistance and loading capacitances, apply to digital simulation. Others, such as the interface subcircuits and the power supplies, apply only to mixed analog/digital simulation.

This section describes in detail:

- the I/O model
- the relationship between drive resistances and output strengths
- charge storage on digital nets
- the format of the interface subcircuits

Input/Output Model

I/O models are common to entire logic families. For example, in the model libraries, there are only four I/O models for the entire 74LS family: IO_LS, for standard inputs and outputs; IO_LS_OC, for standard inputs and open-collector outputs; IO_LS_ST, for Schmitt trigger inputs and standard outputs; and IO_LS_OC_ST, for Schmitt trigger inputs and open-collector outputs. In contrast, timing models are unique to each device.

I/O models are specified as

.MODEL <*I/O model name*> UIO [*model parameters*]*

where valid model parameters are described in Table 6-2.

INLD and OUTLD

These are used in the calculation of loading capacitance, which factors into the propagation delay discussed under timing models on Timing Model on page 218. Note that INLD does not apply to stimulus generators because they have no input nodes.

DRVH and DRVL

These are used to determine the strength of the output. Strengths are discussed on Defining Output Strengths on page 229.

@ 2023 All Rights Reserved.

DRVZ, INR, and TSTOREMN

These are used to determine which nets should be simulated as charge storage nets. These are discussed in Charge storage nets on page 231.

TPWRT

This is used to specify the pulse width above which the noise immunity behavior of a device is to be considered. See Inertial Delay on page 221 on inertial delay for detail.

The following UIO model parameters are needed only when creating models for use in mixed-signal simulations, and therefore only apply to PSpice simulations.

AtoD1 through AtoD4, and DtoA1 through DtoA4

These are used to hold the names of interface subcircuits. Note that AtoD1 through AtoD4 do not apply to stimulus generators because digital stimuli have no input nodes.

DIGPOWER

This is used to specify the name of the digital power supply. PSpice A/D should call if one of the AtoD or DtoA interface subcircuits is called.

TSWLHn and TSWHLn

These switching times are subtracted from a device's propagation delay on the outputs which connect to interface nodes. This compensates for the time it takes the DtoA device to change its output voltage from its current level to that of the switching threshold. By subtracting the switching time from the propagation delay, the analog signal reaches the switching threshold at the correct time (that is, at the exact time of the digital transition). The values for these model parameters should be obtained by measuring the time it takes the analog output of the DtoA (with a nominal analog load attached) to change to the switching threshold after its digital input changes. If the switching time is larger than the propagation delay for an output, no warning is issued, and a delay of zero is used.

When creating your own digital device models, you can create I/O models like these for the primitives you are using. We recommend that you save these in your own custom model library, which you can then configure for use with a given design.

See the *PSpice A/D Reference Guide* for more information on units and defaults for these parameters.

@ 2023 All Rights Reserved.

Note: The switching time parameters are not used when the output drives a digital node.

Table 6-2 Digital I/O model parameters

UIO model parameter	Description
INLD	input load capacitance
OUTLD	output load capacitance
DRVH	output high level resistance
DRVL	output low level resistance
DRVZ	output Z-state leakage resistance
INR	input leakage resistance
TSTOREMN	minimum storage time for net to be simulated as a charge
TPWRT	pulse width rejection threshold
AtoD1 (Level 1)	name of AtoD interface subcircuit
DtoA1 (Level 1)	name of DtoA interface subcircuit
AtoD2 (Level 2)	name of AtoD interface subcircuit
DtoA2 (Level 2)	name of DtoA interface subcircuit
AtoD3 (Level 3)	name of AtoD interface subcircuit
DtoA3 (Level 3)	name of DtoA interface subcircuit
AtoD4 (Level 4)	name of AtoD interface subcircuit
DtoA4 (Level 4)	name of DtoA interface subcircuit
DIGPOWER	name of power supply subcircuit
TSWLH1	switching time low to high for DtoA1
TSWLH2	switching time low to high for DtoA2
TSWLH3	switching time low to high for DtoA3
TSWLH4	switching time low to high for DtoA4
TSWHL1	switching time high to low for DtoA1
TSWHL2	switching time high to low for DtoA2

@ 2023 All Rights Reserved.

Table 6-2 Digital I/O model parameters, *continued*

UIO model parameter	Description
TSWHL3	switching time high to low for DtoA3
TSWHL4	switching time high to low for DtoA4

The digital primitives comprising the 74393 part reference the IO_STD I/O model in the model libraries as shown:

```
.model IO_STD uio (
+  drvh=96.4   drvl=104
+  AtoD1="AtoD_STD"  AtoD2="AtoD_STD_NX"
+  AtoD3="AtoD_STD"  AtoD4="AtoD_STD_NX"
+  DtoA1="DtoA_STD"  DtoA2="DtoA_STD"
+  DtoA3="DtoA_STD"  DtoA4="DtoA_STD"
+  tswhl1=1.373ns          tswlh1=3.382ns
+  tswhl2=1.346ns          tswlh2=3.424ns
+  tswhl3=1.511ns          tswlh3=3.517ns
+  tswhl4=1.487ns          tswlh4=3.564ns
+  )
```

I/O Models for Digital Modeling Applications

Digital devices can also be modeled from the *Modeling Application* panel of the schematic capture application, OrCAD X Capture. To model a digital device, you can either use the default value of the I/O model parameters or modify the values, as required.

Following are the I/O model and AtoD, DtoA interface subcircuit definition used by the primitives describing a circuit:

DtoA Subcircuit:

```
.subckt CDN_DtoA_APP  D A  DPWR DGND
+       params: CAPACITANCE=0 ROUT=1m  drvl=0 drvh=10  voh=5 vol=0
*
N1  A DGND DPWR CDN_DINAPP DGTLNET=D CDN_IO_APP
C1  A DGND {CAPACITANCE+ 1p}

.model CDN_DINAPP dinput (
+       s0name="0"      s0tsw=1p     s0rlo={ROUT}      s0rhi=1e9
+       s1name="1"      s1tsw=1p     s1rlo=1e9       s1rhi={ROUT}
+       s2name="X"      s2tsw=1p     s2rlo={ROUT}      s2rhi={ROUT}
+       s3name="Z"      s3tsw=1p     s3rlo=1e9     s3rhi=1e9
```


@ 2023 All Rights Reserved.

```
+        s4name="R"      s4tsw=1p     s4rlo={ROUT}     s4rhi={ROUT}
+        s5name="F"      s5tsw=1p     s5rlo={ROUT}     s5rhi={ROUT}
+        )


.ends CDN_DtoA_APP
```

Following are the circuits for logic HIGH and logic LOW output voltages:

Circuit 1 is the circuit for logic HIGH output voltage in which output is connected to VOH through ROUT and output is isolated from VOL with high resistance (1e9).

Circuit 2 is the circuit for logic LOW output voltage in which output is connected to VOL through ROUT and output is isolated from VOH with high resistance (1e9).

Note: In the model definition of the DtoA circuit, a capacitor of `1pF` is being added to avoid convergence issues in complex designs that might occur due to abrupt changes in levels.

AtoD Subcircuit:

```
.subckt CDN_AtoD_APP  A D  DPWR DGND optional: DPWR1=$G_DPWR3 DGND1=$G_DGND3
+   params: CAPACITANCE=0  RIN=1e6 TH=5 HYSTWD=0
*
O0  A DGND1 CDN_DOUTAPP DGTLNET=D CDN_IO_APP RIN={RIN}
Rxxx DPWR1 0 1meg
```


@ 2023 All Rights Reserved.

```
.model CDN_DOUTAPP doutput (
+   s1name="0"s1vlo=-1e12s1vhi={TH+HYSTWD/2}
+   s2name="1"s2vlo={TH-HYSTWD/2}s2vhi=1e12
+   )

.ends CDN_AtoD_APP
```

For Digital Power Supply

```
.subckt CDN_DIGIFPWR_APP  AGND DPWR2 DGND2
+       optional: DPWR1=$G_DPWR3 DGND1=$G_DGND3
+       params:   VOH=5 VOL=-5
*
VDPWR2  DPWR2 AGND  {VOH}
VDPWR1  PWRx AGND  5
RPWRx      PWRx DPWR1 1m
Rxnew1  DPWR1 AGND 1meg
R1     DPWR2 AGND  1MEG
VDGND2  DGND2 AGND  {VOL}
VDGND1  PWRx1 AGND  0
RPWRx1  PWRx1 DGND1 1m
Rxnew  DGND1 AGND 1meg
R2     DGND2 AGND  1MEG
.ends CDN_DIGIFPWR_APP

.model CDN_IO_APP uio (
+   drvh=96.4drvl=104
+   AtoD1="CDN_AtoD_APP"AtoD2="CDN_AtoD_APP"
+   AtoD3="CDN_AtoD_APP"AtoD4="CDN_AtoD_APP"
+   DtoA1="CDN_DtoA_APP"DtoA2="CDN_DtoA_APP"
+   DtoA3="CDN_DtoA_APP"DtoA4="CDN_DtoA_APP"
+       tswhl1=0tswlh1=0
+   DIGPOWER="CDN_DIGIFPWR_APP"
+ ROUT=10  VOL=0  VOH=5  RIN=10000000  TH=2.5  HYSTWD=1
+   )
*$
```

In the power supply subcircuit, the following parameters are used:

- Separate digital power and digital ground are used for AtoD and DtoA interfaces to support the parameterization of power levels.

@ 2023 All Rights Reserved.

- DPWR1 and DGND1 are used for the AtoD interface.
- DPWR2 and DGND2 are used for the DtoA interface.
- 1m-ohm resistance (RPWRx1) is connected between DGND1 and AGND to avoid shorting voltage sources during simulation.

  This might result in very small additional current flow through input pins. The current can be decreased by decreasing the value of resistance RPWRx1 or increasing the input resistance of the device.

Related Topics

- PSpice A/D Digital Modeling Application

Defining Output Strengths

The goal of running simulations is to calculate values for each node in the circuit. For analog nodes, the values are voltages. For digital nodes, these values are *states*. The state of a digital node is calculated from the output *strengths* of the devices driving the node and the logic level of the node. Node strength calculations are described in Digital Simulation.

The purpose of strengths is to allow the simulator to find the value of a node when more than one output is driving it. A common example is a bus line which is driven by more than one tristate driver. Under normal circumstances, all drivers except one are driving at the Z (high impedance) strength. Thus, the bus line will take on the value of the one gate that is driving at a higher strength (lower impedance).

Another example is a bus line connected to several open collector output devices and a digital pullup resistor. The pullup resistor outputs a 1 level at a weak (but non-Z) strength. If all of the open-collector devices are outputting at Z strength, then the node will have a 1 level because of the pullup resistor. If any of the open collectors output a 0, at a higher strength than the pullup resistor, then the 0 will overpower the weak 1 from the pullup, and the node will be a 0 level.

Configuring the strength scale

The 64 strengths are determined by two configurable options: DIGDRVZ and DIGDRVF. You can set these options in the Simulation Settings dialog box in PSpice A/D.

DIGDRVZ defines the impedance of the Z strength, and DIGDRVF defines the impedance of the forcing strength. These two values define a logarithmic scale consisting of 64 *ranges* of impedance values. By default, DIGDRVZ is 20 kohms and DIGDRVF is 2 ohms. The larger

@ 2023 All Rights Reserved.

the range between DIGDRVZ and DIGDRVF, the larger the range of impedance values in each of the 64 strengths.

Determining the strength of a device output

The simulator uses the value of the DRVH (high-level driving resistance) or DRVL (low-level driving resistance) parameters from the device's I/O model. If the level of the output is a 1, the simulator obtains the strength by finding the *bin* which contains the value of the DRVH parameter. Likewise, if the level is a 0, the simulator uses the value of the DRVL parameter to obtain the strength.

See Input/Output Model on page 223 for more information.

Figure 6-2 Level 1 and 0 strength determination.

Note that if the values of DRVH and DRVL in the I/O model are different, it is possible for the 1 and 0 levels to have different strengths. This is useful for open-collector devices, where the 0 level is at a higher strength than the 1 level (which drives at the Z strength).

Drive impedances which are higher than the value of DIGDRVZ are assigned the Z strength (0). Likewise, drive impedances lower than the value of DIGDRVF are assigned the forcing strength (63).

Controlling overdrive

During a simulation, the simulator uses only the strength range number (0-63) to compare the driving strength of outputs. The simulator allows you to control how much *stronger* an output

@ 2023 All Rights Reserved.

must be before it overdrives the other outputs driving the same node. This is controlled with the configurable DIGOVRDRV option. By default, DIGOVRDRV is 3, meaning that the strength value assigned to an output must be at least 3 greater than all other drivers before it determines the level of the node.

The accuracy of the DIGOVRDRV strength comparison is limited by the size of the strength range, DIGDRVZ through DIGDRVF. The default drive range of 2 ohms to 20,000 ohms gives strength ranges of 7.5%. The accuracy of the strength comparison is 15%. In other words, depending on the particular values of DRVH and DRVL, it might take as much as a factor of 3.45 to overdrive a signal, or as little as a factor of 2.55. The accuracy of the comparison increases as the ratio between DIGDRVF and DIGDRVZ decreases.

You can set the DRVH, DRVL, DIGDRVF, DIGDRVZ, and DIGOVRDRV options in the Simulation Settings dialog box in PSpice A/D.

Charge storage nets

The ability to model charge storage on digital nets is useful for engineers who are designing dynamic MOS integrated circuits. In such circuits, it is common for the designer to temporarily store a one or zero on a net by driving the net to the appropriate voltage and then turning off the drive. The charge which is trapped on the net causes the net's voltage to remain unchanged for some time after the net is no longer driven. The technique is not normally used on PCB nets because sub-nanoampere input and output leakage currents would be required, as well as low coupling from adjacent signals.

The simulator models the stored charge nets using a simplified *switch-level* simulation technique. A normalized (with respect to power supply) charge or discharge current is calculated for each output or transfer gate attached to the net. This current, divided by the net's total capacitance, is integrated and recalculated at intervals which are appropriate for the particular net. The net's digital level is determined by the normalized voltage on the net. Only the digital level (1, 0, R, F, X) on the net is used by device inputs attached to the net.

This technique allows accurate simulation of networks of transfer gates and capacitive loads. The sharing of charge among several nets which are connected by transfer gates is handled properly because the simulation method calculates the charge transferred between the nets, and maintains a floating-point value for the charge on the net (not just a one or zero). Because of the increased computation, it takes the simulator longer to simulate charge storage nets than normal digital nets. However, charge storage nets are simulated much faster than analog nets.

The I/O model parameters INR, DRVZ, and TSTOREMN (see Table 6-2 on page 225) are used by the simulator to determine which nets should be simulated as charge storage nets. The simulator will simulate charge storage only for a net which has some devices attached to

@ 2023 All Rights Reserved.

it which can be high impedance (Z), and which has a storage time greater than or equal to the smallest TSTOREMN of all inputs attached to the net. The storage time is calculated as the total capacitance (sum of all INLD and OUTLD values for attached inputs and outputs) multiplied by the total leakage resistance for the net (the parallel combination of all INR and DRVZ values for attached inputs and outputs).

Note: The default values provided by the UIO model will **not** allow the use of charge-storage simulation techniques—even with circuits using non-PSpice libraries of digital devices. This is appropriate, since these libraries are usually for PCB-based designs.

Creating your own interface subcircuits for additional technologies

If you are creating custom digital parts for a technology which is not in the model libraries, you may also need to create AtoD and DtoA subcircuits. The new subcircuits need to be referenced by the I/O models for that technology. The AtoD and DtoA interfaces have specific formats, such as node order and parameters, which are expected by PSpice A/D for mixed-signal simulations.

If you are creating parts in one of the logic families *already* in the model libraries, you should reference the existing I/O models appropriate to that family. The I/O models, in turn, automatically reference the correct interface subcircuits for that family. These, too, are already contained in the model libraries.

The AtoD interface subcircuit format is shown here:

```
.SUBCKT ATOD <name suffix>
+ <analog input node>
+ <digital output node>
+ <digital power supply node>
+ <digital ground node>
+ PARAMS: CAPACITANCE=<input load value>
+ {O device, loading capacitor, and other
+ declarations}
.ENDS
```

It has four nodes as described. The AtoD subcircuit has one parameter, CAPACITANCE, which corresponds to the input load. PSpice A/D passes the value of the I/O model parameter INLD to this parameter when the interface subcircuit is called.

The DtoA interface subcircuit format is shown here:

```
.SUBCKT DTOA <name suffix>
+ <digital input node> <analog output node>
+ <digital power supply node> <digital
+ ground node>
+ PARAMS:  DRVL=<0 level driving resistance>
+ DRVH=<1 level driving resistance>
+ CAPACITANCE=<output load value>
```


@ 2023 All Rights Reserved.

```
+ {N device, loading capacitor, and other
+ declarations}
.ENDS
```

It also has four nodes. Unlike the AtoD subcircuit, the DtoA subcircuit has three parameters. PSpice A/D will pass the values of the I/O model parameters DRVL, DRVH, and OUTLD to the interface subcircuit's DRVL, DRVH, and CAPACITANCE parameters when it is called.

The library file `DIG_IO.LIB` contains the I/O models and interface subcircuits for all logic families supported in the model libraries. You should refer to this file for examples of the I/O models, interface subcircuits, and the proper use of N and O devices.

Following are the I/O model and AtoD interface subcircuit definition used by the primitives describing the 74393 part.

```
.model IO_STD uio (
+  drvh=96.4     drvl=104
+  AtoD1="AtoD_STD"   AtoD2="AtoD_STD_NX"
+  AtoD3="AtoD_STD"   AtoD4="AtoD_STD_NX"
+  DtoA1="DtoA_STD"   DtoA2="DtoA_STD"
+  DtoA3="DtoA_STD"   DtoA4="DtoA_STD"
+  tswhl1=1.373             tswlh1=3.382ns
+  tswhl2=1.346ns           tswlh2=3.424ns
+  tswhl3=1.511ns           tswlh3=3.517ns
+  tswhl4=1.487ns           tswlh4=3.564ns
+  )
.subckt AtoD_STD  A D  DPWR DGND
+  params: CAPACITANCE=0
*
O0  A DGND DO74 DGTLNET=D IO_STD
C1  A 0 {CAPACITANCE+0.1pF}
.ends
```

If an instance of the 74393 part is connected to an analog part via node AD_NODE, PSpice A/D generates an interface block using the I/O model specified by the digital primitive actually at the interface. Suppose that U1 is the primitive connected at AD_NODE (see the 74393 subcircuit definition on page 216), and that the IO_LEVEL is set to 1. PSpice A/D determines that IO_STD is the I/O model used by U1. Notice how IO_STD identifies the interface subcircuit names AtoD_STD and DtoA_STD to be used for level 1 subcircuit selection. If the connection with U1 is an input (such as a clock line), PSpice A/D creates an instance of the subcircuit AtoD_STD:

```
X$AD_NODE_AtoD1 AD_NODE AD_NODE$AtoD $G_DPWR
+ $G_DGND
+ AtoD_STD
+ PARAMS: CAPACITANCE=0
```

The AtoD_STD interface subcircuit references the DO74 model in its PSpice O device declaration. This model, stated elsewhere in the model libraries, describes how to translate an analog signal on the analog side of an interface node, to a digital state on the digital side of an interface node.

@ 2023 All Rights Reserved.

```
.model DO74 doutput
+  s0name="X"  s0vlo=0.8  s0vhi=2.0
+  s1name="0"  s1vlo=-1.5 s1vhi=0.8
+  s2name="R"  s2vlo=0.8  s2vhi=1.4
+  s3name="R"  s3vlo=1.3  s3vhi=2.0
+  s4name="X"  s4vlo=0.8  s4vhi=2.0
+  s5name="1"  s5vlo=2.0  s5vhi=7.0
+  s6name="F"  s6vlo=1.3  s6vhi=2.0
+  s7name="F"  s7vlo=0.8  s7vhi=1.4
+
```

The DOUTPUT model parameters are described under O devices in the *PSpice A/D Reference Guide*.

Supposing the output of the 74393 is connected to an analog part via the digital primitive UBUFF. At IO_LEVEL set to 1, PSpice A/D determines that the DtoA_STD interface subcircuit identified in the IO_STD model, should be used.

```
.subckt DtoA_STD  D A  DPWR DGND
+ params: DRVL=0 DRVH=0 CAPACITANCE=0
*
N1  A DGND DPWR DIN74 DGTLNET=D IO_STD
C1  A DGND {CAPACITANCE+0.1pF}
.ends
```

For this subcircuit, the DRVH and DRVL parameters values specified in the IO_STD model would be passed to it. (The interface subcircuits in the model libraries do not currently use these values.)

The DtoA_STD interface subcircuit references the DIN74 model in its PSpice N device declaration. This model, stated elsewhere in the libraries, describes how to translate a digital state into a voltage and impedance.

```
.model DIN74 dinput (
+ s0name="0" s0tsw=3.5ns s0rlo=7.13
+ s0rhi=389 ; 7ohm, 0.09v
+ s1name="1" s1tsw=5.5ns s1rlo=467
+ s1rhi=200 ; 140ohm, 3.5v
+ s2name="X" s2tsw=3.5ns s2rlo=42.9
+ s2rhi=116 ; 31.3ohm, 1.35v
+ s3name="R" s3tsw=3.5ns s3rlo=42.9
+ s3rhi=116 ; 31.3ohm, 1.35v
+ s4name="F" s4tsw=3.5ns s4rlo=42.9
+ s4rhi=116 ; 31.3ohm, 1.35v
+ s5name="Z" s5tsw=3.5ns s5rlo=200K
+ s5rhi=200K
+)
```

The DINPUT model parameters are described under PSpice N devices in the *PSpice A/D Reference Guide*.

Each state is turned into a pullup and pulldown resistor pair to provide the correct voltage and impedance. The Z state is accounted for as well as the 0, 1, and X logic levels.

@ 2023 All Rights Reserved.

You can create your own interface subcircuits, DINPUT models, DOUTPUT models, and I/O models like these for technologies not currently supported in the model libraries. We recommend that you save these in your own custom model library, which you can then configure for use with a given design.

Related Topics

- Digital Modeling Applications

@ 2023 All Rights Reserved.

Creating a Digital Model using the PINDLY and LOGICEXP Primitives

Unlike the majority of analog device types, the bulk of digital devices are not primitives that are compiled into the simulator. Instead, most digital models are macro models or subcircuits that are built from a few primitive devices.

These subcircuits reference interface and timing models to handle the D-to-A and A-to-D interfaces and the overall timing parameters of the physical device. For most families of digital components, the interface models are already defined and available in the `DIG_IO.LIB` library, which is supplied with all digital and mixed-signal packages. If you are unsure of the exact name of the interface model you need to use, use a text editor to look in `DIG_IO.LIB`.

For instance, if you are trying to model a 74LS component that is not already in a library, open `DIG_IO.LIB` with your text editor and search for 74LS to get the interface models for the 74LS family. You can also read the information at the beginning of the file which explains many of the terms and uses for the I/O models.

In the past, the timing model has presented the greatest challenge when trying to model a digital component. This was due to the delays of a component being distributed among the various gates. Recently, the ability to model digital components using logic expressions (LOGICEXP) and pin-to-pin delays (PINDLY) has been added to the simulator. Using the LOGICEXP and PINDLY digital primitives, you can describe the logic of the device with zero delay and then enter the timing parameters for the pin-to-pin delays directly from the manufacturer's data sheet. Digital primitives still must reference a standard timing model, but when the PINDLY device is used, the timing models are simply zero-delay models that are supplied in `DIG_IO.LIB`. The default timing models can be found in the same manner as the standard I/O models. The PINDLY primitive also incorporates constraint checking which allows you to enter device data such as pulse width and setup/hold timing from the data sheet. Then the simulator can verify that these conditions are met during the simulation.

Digital primitives

Primitives in the simulator are devices or functions which are compiled directly into the code. The primitives serve as fundamental building blocks for more complex macro models.

There are two types of primitives in the simulator: gate level and behavioral. A gate level primitive normally refers to an actual physical device (such as buffers, AND gates, inverters). A behavioral primitive is not an actual physical device, but rather helps to define parameters of a higher level model. Just like gate level primitives, behavioral primitives are intrinsic functions in the simulator and are treated in much the same manner. They are included in the gate count for circuit size and cannot be described by any lower level model.

@ 2023 All Rights Reserved.

In our 74160 example (see *The TTL Data Book* from Texas Instruments for schematic and description), the four J-K flip-flops are the four digital gate level primitives. While flip-flops are physically more complex than gates in terms of modeling, they are defined on the same level as a gate (for example, flip-flops are a basic device in the simulator). Since all four share a common Reset, Clear, and Clock signal, they can be combined into one statement as an array of flip-flops. They could just as easily have been written separately, but the array method is more compact. See the *Digital Devices* chapter in the *PSpice A/D Reference Guide* for more information.

Logic Expression (LOGICEXP primitive)

Looking at the listing in 74160 Example on page 242 and at the schematic representation of the 74160 subcircuit, you can see that there are three main parts to the subcircuit. Following the usual header information, .SUBCKT keyword, subcircuit name, interface pin list, and parameter list is the LOGICEXP primitive. It contains everything in the component that can be expressed in terms of simple combinational logic. The logic expression device also serves to buffer other input signals that will go to the PINDLY primitive. In this case, LOGICEXP buffers the ENP_I, ENT_I, CLK_I, CLRBAR_I, LOADBAR_I, and four data signals. See the *Digital Devices* chapter in the *PSpice A/D Reference Guide* for more information.

For our 74160 example, the logic expression (LOGICEXP) has fourteen inputs and twenty outputs. The inputs are the nine interface input pins in the subcircuit plus five feedback signals that come from the flip-flops (QA, QB, QC, QD, and QDBAR). The flip-flops are primitive devices themselves and are not part of the logic expression. The outputs are the eight J-K data inputs to the flip-flops, RCO, the four data lines used internal to the logic expression (A, B, C, D), and the seven control lines: CLK, CLKBAR, EN, ENT, ENP, CLRBAR, and LOADBAR.

The schematic representation of the device shows buffers on every input signal of the model, while the logic diagram of the device in the data book shows buffers or inverters on only the CLRBAR_I, CLK_I, and LOADBAR_I signals. We have added buffers to the inputs to minimize the insertion of A-to-D interfaces when the device is driven by analog circuitry. The best example is the CLK signal. With the buffer in place, if the CLK signal is analog, one A-to-D interface device will be inserted into the circuit by the simulator. If the buffer was not present, then an interface device would be inserted at the CLK pin of each of the flip-flops. The buffers have no delay associated with them, but by minimizing the number of A-to-D interfaces, we speed up the mixed-signal simulation by reducing the number of necessary calculations. For situations where the device is only connected to other digital nodes, the buffers have no effect on the simulation.

The D0_GATE, shown in the listing, is a zero-delay primitive gate timing model. For most TTL modeling applications, this only serves as a place holder and is not an active part of the model. Its function has been replaced by the PINDLY primitive. The D0_GATE model can be

@ 2023 All Rights Reserved.

found in the library file `DIG_IO.LIB`. For a more detailed description of digital primitives, see the *Digital Devices* chapter in the *PSpice A/D Reference Guide*.

IO_STD, shown in the listing, is the standard I/O model. This determines the A-to-D and D-to-A interface characteristics for the subcircuit. The device contains family-specific information, but the models have been created for nearly all of the stock families. The various I/O models can be found in the library file `DIG_IO.LIB`.

The logic expressions themselves are straightforward. The first nine are buffering the input signals from outside the subcircuit. The rest describe the logic of the actual device up to the flip-flops. By tracing the various paths in the design, you can derive each of the logic equations.

The D0_EFF timing model, shown in the listing, is a zero-delay default model already defined in `DIG_IO.LIB` for use with flip-flops. All of the delays for the device are defined in the PINDLY section. The I/O model is IO_STD as identified previously. We have not specified a MNTYMXDLY or IO_LEVEL parameter, so the default values are used. For a more detailed description of the general digital primitives MNTYMXDLY and IO_LEVEL, see the *Digital Devices* chapter in the *PSpice A/D Reference Guide*.

The primitive MNTYMXDLY specifies whether to use the minimum, typical, maximum, or digital worst-case timing values from the device's timing model (in this case the PINDLY device). For the 74160, MNTYMXDLY is set to 0. This means that it takes on the current value of the DIGMNTYMX parameter. DIGMNTYMX defaults to 2 (typical timing) unless specifically changed using the .OPTIONS command.

The primitive IO_LEVEL selects one of four possible A-to-D and D-to-A interface subcircuits from the device's I/O model. In the header of this subcircuit, IO_LEVEL is set to 0. This means that it takes on the value of the DIGIOLVL parameter. DIGIOLVL defaults to 1 unless specifically changed using the .OPTIONS command.

Pin-to-pin delay (PINDLY primitive)

The delay and constraint specifications for the model are specified using the PINDLY primitive. The PINDLY primitive is evaluated every time any of its inputs or outputs change. See the *Digital Devices* chapter in the *PSpice A/D Reference Guide* for more information.

For the 74160, we have five delay paths, the four flip-flop outputs to subcircuit outputs QA...QD to QA_O...QD_O, and RCO to RCO_O. The five paths are seen in the Delay & Constraint section of the design. For delay paths, the number of inputs must equal the number of outputs. Since the 74160 does not have TRI-STATE outputs, there are no enable signals for this example, but there are ten reference nodes. The first four (CLK, LOADBAR, ENT, and

@ 2023 All Rights Reserved.

CLRBAR) are used for both the pin-to-pin delay specification and the constraint checking. The last six (ENP, A, B, C, D, and EN) are used only for the constraint checking.

The PINDLY primitive also allows constraint checking of the model. It can verify the setup, hold times, pulse width, and frequency. It also has a general mechanism to allow for user-defined conditions to be reported. The constraint checking only reports timing violations; it does not affect the propagation delay or the logic state of the device. Since the timing parameters are generally specified at the pin level of the actual device, the checking is normally done at the interface pins of the subcircuit after the appropriate buffering has been done.

BOOLEAN

The keyword BOOLEAN begins the boolean assignments which define temporary variables that can be used later in the PINDLY primitive. The form is:

boolean variable = {boolean expression}

The curly braces are required.

In the 74160 model, the boolean expressions are actually reference functions. There are three reference functions available: CHANGED, CHANGED_LH, and CHANGED_HL. The format is:

function name (node, delta time)

For our example, we define the variable CLOCK as a logical TRUE if there has been a LO-to-HI transition of the CLK signal at simulation time. We define CNTENT as TRUE if there has been any transition of the ENT signal at the simulation time.

Boolean operators take the following boolean values as operands:

- reference functions
- transition functions
- previously assigned boolean variables
- boolean constants TRUE and FALSE

Transition functions have the general form of:

TRN_pn

For a complete list of reference functions and transition functions, see the *Digital Devices* chapter in the *PSpice A/D Reference Guide*.

@ 2023 All Rights Reserved.

PINDLY

PINDLY contains the actual delay and constraint expressions for each of the outputs.

The CASE function defines a more complex, rule-based <*delay expression*> and works as a rule section mechanism for establishing path delays. Each boolean expression in the CASE function is evaluated in order until one is encountered that produces a TRUE result. Once a TRUE expression is found, the delay expression portion of the rule is associated with the output node being evaluated, and the remainder of the CASE function is ignored. If none of the expressions evaluate to TRUE, then the DEFAULT delay is used. Since it is possible for none of the expressions to yield a TRUE result, you must include a default delay in every CASE function. Also note that the expressions must be separated by a comma.

In the PINDLY section of the PINDLY primitive in the model listing, the four output nodes (QA_O through QD_O) all use the same delay rules. The CASE function is evaluated independently for each of the outputs in turn. The first delay expression is:

```
CLOCK & LOADBAR=='1 & TRN_LH, DELAY(-1,13NS,20NS)
```

This means that if CLOCK is TRUE, and LOADBAR is equal to 1, and QA_O is transitioning from 0 to 1, then the values of -1, 13ns, and 20ns are used for the MINIMUM, TYPICAL, and MAXIMUM propagation delay for the CLK-to-QA data output of the chip. In this case, the manufacturer did not supply a minimum prop delay, so we used the value -1 to tell the simulator to derive a value from what was given. If this statement is TRUE, then the simulator assigns the values and move on to the CASE function for QB_O and eventually RCO_O.

For instances where one or more propagation delay parameters are not supplied by the data sheet, the simulator derives a value from what is known and the values specified for the .OPTION DIGMNTYSCALE and DIGTYMXSCALE.

When the typical value for a delay parameter is known but the minimum is not, the simulator uses the formula:

TPxxMN = DIGMNTYSCALE X TPxxTY

where the value of DIGMNTYSCALE is between 0.1 and 1.0 with the default value being 0.4. If the typical is known and the maximum is not, then the simulator uses the formula:

TPxxMX = DIGTYMXSCALE X TPxxTY

where the value of DIGTYMXSCALE is greater than 1.0 with the default being 1.6. If the typical value is not known, and both the minimum and maximum are, then the typical value used by the simulator will be the average of the minimum and maximum propagation delays. If only one of min or max is known, then the typical delay is calculated using the appropriate formula as listed above. If all three are unknown, then they all default to a value of 0.

@ 2023 All Rights Reserved.

Constraint Checker (CONSTRAINT Primitive)

The CONSTRAINT primitive provides a general constraint checking mechanism to the digital device modeler. It performs setup and hold time checks, pulse width checks, frequency checks, and includes a general mechanism to allow user-defined conditions to be reported. See the *Digital Devices* chapter in the *PSpice A/D Reference Guide* for more information.

Setup_Hold

The expressions in the SETUP_HOLD specification may be listed in any order.

CLOCK defines the node that is to be used as the reference for the setup/hold/release specification. The assertion edge must be LH or HL (for example, a transition from logic state 0 to 1 or from 1 to 0.)

DATA specifies which node(s) is to have its setup/hold time measured.

SETUPTIME defines the minimum time that all DATA nodes must be stable prior to the assertion edge of the clock. The time value must be a nonnegative constant or expression and is measured in seconds. If the device has different setup/hold times depending on whether the data is HI or LOW at the clock change, you can use either or both of the following forms:

SETUPTIME_LO = <time value>
SETUPTIME_HI = <time value>

If either of the time values is 0, then no check is done for that case.

HOLDTIME is used in the same way as SETUPTIME and also has the alternate _LH and _HL formats and 0 value condition.

RELEASETIME causes the simulator to perform a special-purpose setup check. Release time (also referred to as recovery time in some data sheets) refers to the minimum time that a signal can go inactive before the active clock edge. Again, the _LH and _HL forms are available. The difference between RELEASETIME and SETUPTIME checking is that simultaneous CLOCK/DATA transitions are never allowed (this assumes a nonzero hold time). RELEASETIME is usually not used in conjunction with SETUPTIME or HOLDTIME.

Width

WIDTH does the minimum pulse-width checking. MIN_HI/MIN_LO is the minimum time that the node can remain HI/LOW. The value must be a nonnegative constant, or expression. A

@ 2023 All Rights Reserved.

value of 0 means that any pulse width is allowed. At least one of MIN_HI or MIN_LO must be used within a WIDTH section.

Freq

FREQ checks the frequency. MINFREQ/MAXFREQ is the minimum/maximum frequency that is allowed on the node in question. The value must be a nonnegative floating point constant or expression measured in hertz. At least one of MINFREQ or MAXFREQ must be used within a FREQ section.

AFFECTS clauses (not used in this example) can be included in constraints to describe how the simulator should associate the failure of a constraint check with the outputs (paths through the device) of the PINDLY. This information does not affect the logic state of the outputs but provides causality detail used by the error tracking mechanism in PSpice A/D waveform analysis.

74160 Example

In the 74160 example, we are checking that the maximum clock frequency (CLK) is not more than 25 MHz and the pulse width is 25 ns. We are also checking that the CLRBAR signal has a minimum LO pulse width of 20 ns, and that the 4 data inputs (A, B, C, D) have a setup/hold time of 20 ns in reference to the CLK signal. We are also checking that ENP and ENT have a setup/hold time of 20 ns with respect to the 0 to 1 transition of the CLK signal, but only when the conditions in the WHEN statement are met. All of the delay and constraint checking values were taken directly from the actual data sheet. This makes the delay modeling both easy and accurate.

All of the above primitives and modeling methods, as well as a few special cases that are not covered here, can be found in the *Digital Devices* chapter of the *PSpice A/D Reference Guide*.

```
* 74160  Synchronous 4-bit Decade Counters with asynchronous clear

* Modeled using LOGICEXP, PINDLY, & CONSTRAINT devices

.SUBCKT 74160 CLK_I ENP_I ENT_I CLRBAR_I LOADBAR_I A_I B_I C_I D_I
+ QA_O QB_O QC_O QD_O RCO_O
+ OPTIONAL: DPWR=$G_DPWR DGND=$G_DGND
+ PARAMS: MNTYMXDLY=0 IO_LEVEL=0
*
U160LOG LOGICEXP(14,20) DPWR DGND
+ CLK_I ENP_I ENT_I CLRBAR_I LOADBAR_I A_I B_I C_I D_I
+     QDBAR QA QB QC QD
+ CLK ENP ENT CLRBAR LOADBAR A B C D
+ CLKBAR RCO JA JB JC JD KA KB KC KD EN
+ D0_GATE IO_STD IO_LEVEL={IO_LEVEL}
+ LOGIC:
```


@ 2023 All Rights Reserved.

```
+   CLK = { CLK_I }                     ;Buffering
+   ENP = { ENP_I }
+   ENT = { ENT_I }
+   CLRBAR = { CLRBAR_I }
+   LOADBAR = { LOADBAR_I }
+   A = { A_I }
+   B = { B_I }
+   C = { C_I }
+   D = { D_I }
+   CLKBAR = { ~CLK }               ;Logic expressions
+   LOAD = { ~LOADBAR }
+   EN = { ENP & ENT }
+   I1A = { LOAD | EN }
+   I2A = { ~(LOAD & A) }
+   JA = { I1A & ~(LOAD & I2A) }
+   KA = { I1A & I2A }
+   I1B = { (QA & EN & QDBAR) | LOAD }
+   I2B = { ~(LOAD & B) }
+   JB = { I1B & ~(LOAD & I2B) }
+   KB = { I1B & I2B }
+   I1C = { (QA & EN & QB) | LOAD }
+   I2C = { ~(LOAD & C) }
+   JC = { I1C & ~(LOAD & I2C) }
+   KC = { I1C & I2C }
+   I1D = { ((QC & QB & QA & EN) | (EN & QA & QD)) | LOAD }
+   I2D = { ~(LOAD & D) }
+   JD = { I1D & ~(LOAD & I2D) }
+   KD = { I1D & I2D }
+   RCO = { QD & QA & ENT }
*
UJKFF JKFF(4) DPWR DGND $D_HI CLRBAR CLKBAR JA JB JC JD KA KB KC KD
+ QA QB QC QD QABAR QBBAR QCBAR QDBAR D0_EFF IO_STD
U160DLY PINDLY (5,0,10) DPWR DGND
+ RCO QA QB QC QD
+ CLK LOADBAR ENT CLRBAR ENP A B C D EN
+ RCO_O QA_O QB_O QC_O QD_O
+ IO_STD MNTYMXDLY={MNTYMXDLY} IO_LEVEL={IO_LEVEL}
+ BOOLEAN:
+   CLOCK = { CHANGED_LH(CLK,0) }
+   CNTENT = { CHANGED(ENT,0) }
+ PINDLY:
+   QA_O QB_O QC_O QD_O = {
+     CASE(
+       CLOCK & LOADBAR=='1 & TRN_LH, DELAY(-1,13NS,20NS),
+       CLOCK & LOADBAR=='1 & TRN_HL, DELAY(-1,15NS,23NS),
+       CLOCK & LOADBAR=='0 & TRN_LH, DELAY(-1,17NS,25NS),
+       CLOCK & LOADBAR=='0 & TRN_HL, DELAY(-1,19NS,29NS),
+       CHANGED_HL(CLRBAR,0), DELAY(-1,26NS,38NS),
+       DELAY(-1,26NS,38NS)
+   )
+     }
+   RCO_O = {
+     CASE(
+       CNTENT, DELAY(-1,11NS,16NS),
+       CLOCK, DELAY(-1,23NS,35NS),
+       DELAY(-1,23NS,35NS)
+       )
+     }
+ FREQ:
+   NODE = CLK
+   MAXFREQ = 25MEG
```


@ 2023 All Rights Reserved.

```
+ WIDTH:
+   NODE = CLK
+   MIN_LO = 25NS
+   MIN_HI = 25NS
+ WIDTH:
+   NODE = CLRBAR
+   MIN_LO = 20NS
+ SETUP_HOLD:
+   DATA(4) = A B C D
+   CLOCK LH = CLK
+   SETUPTIME = 20NS
+   WHEN = { (LOADBAR!='1 ^ CHANGED(LOADBAR,0)) &
+     CLRBAR!='0 }
+ SETUP_HOLD:
+   DATA(2) = ENP ENT
+   CLOCK LH = CLK
+   SETUPTIME = 20NS
+   WHEN = { CLRBAR!='0 & (LOADBAR!='0 ^
+     CHANGED(LOADBAR,0))
+     & CHANGED(EN,20NS) }
+ SETUP_HOLD:
+   DATA(1) = LOADBAR
+   CLOCK LH = CLK
+   SETUPTIME = 25NS
+   WHEN = { CLRBAR!='0 }
+ SETUP_HOLD:
+   DATA(1) = CLRBAR
+   CLOCK LH = CLK
+   RELEASETIME_LH = 20NS
.ENDS
```


@ 2023 All Rights Reserved.

7

Setting Up Analyses and Starting Simulation

You are required to create a simulation profile specifying the details of simulation to be performed.

Before you simulate a PSpice-compatible design, you need to define analyses settings, which includes analysis types such as standard analysis, multi-run analysis and analytical analysis.

The following table provides a summary of the available PSpice A/D analyses and the corresponding analysis type options where the analysis parameters are specified:

Analysis	Analysis Option	Swept Variable
Standard Analyses		
DC sweep	DC Sweep	source parameter temperature
Bias point	Bias Point	-
Small-signal DC transfer	Bias Point	-
DC sensitivity	Bias Point	-
Frequency response	AC Sweep or Noise	frequency
Noise (requires a frequency response analysis)	AC Sweep or Noise	frequency
Transient response	Time Domain (Transient)	time
Fourier (requires transient response analysis)	Time Domain (Transient)	time

@ 2023 All Rights Reserved.

Analysis	Analysis Option	Swept Variable
Multi-run Analyses		
Parametric	Parametric Sweep	
Temperature	Temperature (Sweep)	
Statistical Analyses		
Monte Carlo	Monte Carlo or Worst Case	
Sensitivity/worst-case	Monte Carlo or Worst Case	

For normal simulations that are run from a simulation profile, or in batch mode, only the particular analysis type that is specified will be run.

During simulation of a circuit file, the analysis types are performed in the order shown in the following table:

Table 7-1 Order for Standard Analysis

1. DC sweep
2. Bias point
3. Frequency response
4. Noise
5. DC sensitivity
6. Small-signal DC transfer
7. Transient response
8. Fourier components

Each type of analysis is conducted only once per run.

Several analyses such as small-signal transfer, DC sensitivity, and frequency response depend upon the bias point calculation. Because so many analyses use the bias point, PSpice A/D calculates this automatically. PSpice A/D bias point calculation computes initial states of digital components as well as the analog components.

@ 2023 All Rights Reserved.

Output Variables

For certain analyses such as noise, Monte Carlo, sensitivity or worst-case, DC Sensitivity, Fourier, and small-signal DC transfer function, you require to specify output variables for voltages and currents at specific points on the schematic. Depending upon the analysis type, you may need to specify the following:

- Voltage on a net, a pin, or at a terminal of a semiconductor device
- Current through a part or into a terminal of a semiconductor device
- A device name

If output variables or other information are required, select *More Settings...* to open *Output File Options* in the Monte Carlo/Worst Case dialog box and enter the required parameters.

Voltage

Specify voltage in the following format:

v[*modifiers*](<*out id*>[,<*out id*>]) where <*out id* > is:	(1)
<*net id*> or <*pin id*>	(2)
<*net id*> is a fully qualified net name	(3)
<*pin id*> is <*fully qualified device name*>:<*pin name*>	(4)

@ 2023
All Rights Reserved.

A fully qualified net name (as referred to in line 3 above) is formed by prefixing the visible net name (from a label applied to one of the segments of a wire or bus, or an offpage port connected to the net) with the full hierarchical path, separated by periods. At the top level of hierarchy, this is just the visible name.

A fully qualified device name (from line 4 above) is distinguished by specifying the full hierarchical path followed by the device's part reference, separated by period characters. For example, a resistor with part reference R34 inside part Y1 placed on a top-level schematic page is referred to as Y1.R34 when used in an output variable.

A <*pin id*> (from line 4) is uniquely distinguished by specifying the full part name (as described above) followed by a colon, and the pin name. For example, the pins on a capacitor with reference designator C31 placed on a top-level page and pin names 1 and 2 would be identified as C31:1 and C31:2, respectively.

Current

Specify current in the following format:

i[*modifiers*](<*out device*>[:*modifiers*])

where <*out device*> is a fully qualified device name.

Modifiers

The basic syntax for output variables can be modified to indicate terminals of semiconductors and AC specifications. The modifiers come before <*out id*> or <*out device*>. Or, when specifying terminals (such as source or drain), the modifier is the pin name contained in <*out id*>, or is appended to <*out device*> separated by a colon.

Modifiers can be specified as follows:

- For voltage:

  v[*AC suffix*](<*out id*>[, *out id*])
  v[*terminal*]*(<*out device*>)

- For current:

  i[*AC suffix*](<*out device*>[:*terminal*])
  i[*terminal*][*AC suffix*](<*out device*>])

@ 2023 All Rights Reserved.

where

terminal	specifies one or two terminals for devices with more than two terminals, such as D (drain), G (gate), S (source)
AC suffix	specifies the quantity to be reported for an AC analysis, such as M (magnitude), P (phase), G (group delay)
out id	specifies either the *<net id>* or *<pin id>* (*<fully qualified device name>*:*<pin name>*)
out device	specifies the *<fully qualified device name>*

These building blocks can be used for specifying output variables as shown in Table 7-2 (which summarizes the accepted output variable formats) and Tables 7-3 through 7-6 (which

@ 2023 All Rights Reserved.

list valid elements for two-terminal, three- or four-terminal devices, transmission line devices, and AC specifications).

Table 7-2 PSpice A/D output variable formats

Format	Meaning
V[*ac*](< + *out id* >)	voltage at *out id*
V[*ac*](< *+out id* >,< - *out id* >)	voltage across + and - *out id's*
V[*ac*](< *2-terminal device out id >)*	voltage at a *2-terminal device out id*
V[*ac*](< *3 or 4-terminal device out id* >) or V<*x*>[*ac*](< *3 or 4-terminal out device* >)	voltage at non-grounded terminal *x* of a *3 or 4-terminal device*
V<*x*><*y*>[*ac*](< *3 or 4-terminal out device* >)	voltage across terminals *x* and *y* of a *3 or 4-terminal device*
V[*ac*](< *transmission line out id* >) or V<*z*>[*ac*](< *transmission line out device* >)	voltage at one end *z* of a *transmission line device*
I[*ac*](< *3 or 4-terminal out device* >:<*x*>) or I<*x*>[*ac*](< *3 or 4-terminal out device* >)	current through non-grounded terminal *x* of a *3 or 4-terminal out device*
I[*ac*](< *transmission line out device* >:<*z*>) or I<*z*>[*ac*](< *3 or 4-terminal out device* >)	current through one end *z* of a *transmission line out device*
< *DC sweep variable* >	voltage or current source name

Table 7-3 Element definitions for 2-terminal devices

Device type	**<*out id*> or <*out device*> device indicator**	**Output variable examples**
capacitor	C	V(CAP:1) I(CAP)

@ 2023 All Rights Reserved.

Table 7-3 Element definitions for 2-terminal devices

Device type	<*out id*> or <*out device*> device indicator	Output variable examples
diode	D	V(D23:1)
		I(D23)
voltage-controlled voltage source	E	V(E14:1)
		I(E14)
current-controlled current source	F	V(F1:1)
		I(F1)
voltage-controlled current source	G	V(G2:1)
		I(G2)
current-controlled voltage source	H	V(HSOURCE:1)
		I(HSOURCE)
independent current source	I	V(IDRIV:+)
		I(IDRIV)
inductor	L	V(L1:1)
		I(L1)
resistor	R	V(RC1:1)
		I(RC1)
voltage-controlled switch	S	V(SWITCH:+)
		I(SWITCH)
independent voltage source	V	V(VSRC:+)
		I(VSRC)
current-controlled switch	W	V(W22:-)
		I(W22)

@ 2023 All Rights Reserved.

Table 7-4 Element definitions for 3- or 4-terminal devices

Device type	**<*out id*> or <*out device*> device indicator**	*<pin id>*		**Output variable examples**
GaAs MESFET	B	D	(Drain terminal)	V(B11:D)
		G	(Gate terminal)	ID(B11)
		S	(Source terminal)	
Junction FET	J	D	(Drain terminal)	VG(JFET)
		G	(Gate terminal)	I(JFET:G)
		S	(Source terminal)	
MOSFET	M	B	(Bulk, substrate terminal)	VDG(M1)
				ID(M1)
		D	(Drain terminal)	
		G	(Gate terminal)	
		S	(Source terminal)	
bipolar transistor	Q	B	(Base terminal)	V(Q1:B)
		C	(Collector terminal)	I(Q1:C)
		E	(Emitter terminal)	
		S	(Source terminal)	
IGBT	Z	C	(Collector terminal)	V(Z1:C)
		E	(Emitter terminal)	I(Z1:C)
		G	(Gate terminal)	

@ 2023 All Rights Reserved.

Table 7-5 Element definitions for transmission line devices

Device type	**<*out id*> or <*out device*> device indicator**	**<*z*>**	**Output variable examples**
transmission line	T	A (Port A)	V(T32:A+)
		B (Port B)	I(T32:B-)

Table 7-6 Element definitions for AC analysis specific elements

<*ac suffix*> device symbol	**Meaning**	**Output variable examples**
(none)	magnitude (default)	V(V1)
		I(V1)
M	magnitude	VM(CAP1:1)
		IM(CAP1:1)
DB	magnitude in decibels	VDB(R1)
P	phase	IP(R1)
R	real part	VR(R1)
I	imaginary part	VI(R1)

The INOISE, ONOISE, DB(INOISE), and DB(ONOISE) output variables are predefined for use with noise (AC sweep) analysis.

@ 2023 All Rights Reserved.

Creating a New Simulation Profile

A simulation profile (`*.SIM`) saves your simulation settings for an analysis type so you can reuse them easily.

You can use PSpice to create a new simulation profile only by importing settings from a simulation profile that exists in the same project or in another project.

To import settings from an existing simulation profile to create a new simulation profile, do the following:

1. From the *File* menu in PSpice A/D, click *New* and choose *Simulation Profile*.

   The *New Simulation* dialog box appears.

2. In the *Profile name* text box, specify a name for the profile (such as the name of the analysis type for the new profile).

3. Click the Browse button to specify the location and name of an another simulation profile (`.sim`) to import its settings into the new profile.

4. Click *Create* to create the profile and display the *Simulation Settings* dialog box.

   A new simulation profile is created and can be seen under *PSpice Resources – Simulation Profiles* of the project which you had selected to import the existing simulation profile in the *Inherit from an existing profile* text box.

Modifying an Existing Simulation Profile

You can also modify an existing simulation profile to change the simulation settings. You can edit the current simulation profile in PSpice using the following steps:

1. From the *Simulation* menu, choose *Edit Profile* to display the *Simulation Settings* dialog box.

2. Make the required changes in the *Simulation Settings* dialog box.

3. Click *OK* to save the changes.

The following table explains various tabs of the *Simulation Settings* dialog box:

Simulation Settings	Know More
General	General Simulation Settings
Analysis	Analysis Settings

@ 2023 All Rights Reserved.

Configuration Files	Configuration Files include: ■ Library settings for simulation profiles ■ Stimulus settings for simulation profiles ■ Include files settings for simulation profiles
Options tab	Options for Simulation Profiles
Data Collection tab	Setting Data Collection Options
Probe window	By default, the probe data has 64-bit accuracy. You can set the accuracy by selecting a different radio button for Probe Data. A 64-bit accuracy setting ensures that the output does not have ramps. For example, when a small amplitude voltage is superimposed on a large amplitude voltage, the resolution might be lost, resulting in ramps.

Using a Profile for Simulation

You can simulate the circuit using any of the simulation profiles you have created in the project. To use a simulation profile, open it in PSpice using the following steps:

1. From the *File* menu, choose *Open Simulation*.
2. Select the <profile_name>.SIM file and click *Open*.

Let us take an example to make a simulation profile active in Capture schematic application.

To make a simulation profile active in Capture, do one of the following:

- From the PSpice toolbar in Capture, select the simulation profile you want to use. See PSpice toolbar in Capture for more information.
- Select the simulation profile in the Capture Project Manager, and choose *Make Active* from the *PSpice* menu in Capture.

The icon indicates an active simulation profile.

Each simulation profile is associated with a schematic. When you select a simulation profile in Capture, the folder for the associated schematic becomes the root schematic folder. When you run PSpice, only the pages in the root schematic folder are netlisted. If the schematic pages in the root schematic folder refer to pages in another schematic folder (as in the case of hierarchical designs), the pages in that schematic folder also get netlisted.

@ 2023 All Rights Reserved.

Deleting a Simulation Profile

You can delete a simulation profile only from the project manager of the schematic tool application. To delete a simulation profile, do the following:

1. Select the simulation profile in the project manager and press `Delete`.

   Note: When you delete a simulation profile, the profile is deleted from the project but the files related to the simulation profile are not deleted from the project directory. You can manually delete the files from the project directory. For example, if you delete a simulation profile named TRAN, you must delete the TRAN folder located at `<project_directory>\<design_name>-PSpiceFiles\<root_schematic_name>\`.

   Note: If you delete a simulation profile but do not delete the files related to that profile, and later create a profile with the same name for the schematic, the schematic capture tool displays an error message that a profile with the same name exists. You can overwrite the old profile or create another profile with a different name.

General Simulation Settings

Use the General tab of the *Simulation Settings* dialog box to specify the simulation input and output files and save a description of the simulation profile.

@ 2023
All Rights Reserved.

Profile Name

The profile name text box displays the name of the current profile. You can change the name of the simulation profile.

Input Settings

Choose schematic to have the simulation input come from a design (*.DSN file), or choose Circuit File (*.CIR) to have the simulation input come from a circuit file. For either input type, enter the name of the file in the text box.

For a design-based simulation, also select the name of the top-level schematic page. Use the *Analysis* tab of this dialog box to enter simulation settings.

For a circuit-file-based design, the *Analysis* tab settings are unavailable. You can change the simulation settings by editing the commands in the .CIR file.

Output Settings

In the *Output filename* text box, enter a file name (*.OUT) for the simulation output to be saved to.

In the *Waveform data filename* text box, enter a filename (*.DAT) for the simulation waveform data to be saved to.

Note: Use this text box to describe the design or simulation profile.

Related Topics

- .SUBCKT (subcircuit)

Analysis Settings

Use the *Analysis* tab of the *Simulation Settings* dialog box to define the basic analysis type and set up additional advanced analyses and simulation options.

Analysis Types

From the *Analysis Type* drop-down list box, select the basic analysis type you want to run. For more information, refer to Analyses to Run with PSpice A/D.

@ 2023 All Rights Reserved.

Analysis Options

Under *Options* tab, select an advanced analysis you want to use. To know more, refer to Analyses to Run with PSpice A/D.

Configuration Files

You can use the *Configuration Files* tab of the *Simulation Settings* dialog box to configure and edit the following:

- Stimulus Settings
- Library Settings
- Include Files Settings

Stimulus Settings

Use the *Configuration Files* tab of the *Simulation Settings* dialog box to configure global, design and profile level stimulus files.

Click *Stimulus* in the *Category* field in the *Configuration Files* tab, and configure the stimulus files.

Click this button…	To do this…
Browse	Browse directories for libraries to make available to the profile-the selected file name appears in the Filename text box
Add as Global	Add the file listed in the Filename text box to be available to all designs
Add to Design	Add the file listed in the Filename text box to be available to the current design only

@ 2023 All Rights Reserved.

`Add to Profile`	Add the file listed in the Filename text box to be available to the current profile only. A file added at the design level cannot be added at the profile level also. **Note:** Profile specific configuration is available only if the project was created using Capture from release 10.0 and beyond. This feature is also available for the project created using older versions of Capture but converted to the Capture 10.0 format.
`Change`	Change the selected file with the file specified in the Filename text box
	Remove the selected file from the Variant Files list—this does not delete the file from your computer
`and`	Move the selected file up or down in the Variant Files list. PSpice locates the stimuli by searching the stimulus files in the order in which they are listed in the Variant Files list. PSpice locates the stimuli by first searching the stimulus files configured at the profile level, then the stimulus files configured at the design level and finally the stimulus files configured at the global level. **Note:** You can change the search order only within the files configured at the same level. For example, you can change the search order only within the files configured at the profile level.

Library Settings

Use the *Configuration Files* tab of the *Simulation Settings* dialog box configure global, design and profile level model libraries.

1. Click *Library* in the *Category* field in the *Configuration Files* tab, and configure the model libraries.

`Click this button…`	To do this…
`Browse`	Browse directories for libraries to make available to the profile—the selected file name appears in the Filename text box

@ 2023 All Rights Reserved.

`Add as Global`	Add the file listed in the Filename text box to be available to all designs
`Add to Design`	Add the file listed in the Filename text box to be available to the current design only
`Add to Profile`	Add the file listed in the Filename text box to be available to the current profile only A file added at the design level cannot be added at the profile level also. **Note:** Profile specific configuration is available only if the project was created using Capture from release 10.0 and beyond. This feature is also available for the project created using older versions of Capture but converted to the Capture 10.0 format.
`Change`	Change the selected file with the file specified in the Filename text box
	Remove the selected file from the Variant Files list—this does not delete the file from your computer
`and`	Move the selected file up or down in the Variant Files list. PSpice locates models by searching the model libraries in the order in which they are listed in the Variant Files list. PSpice locates models by first searching the model libraries configured at the profile level, then the model libraries configured at the design level and finally the model libraries configured at the global level. **Note:** You can change the search order only within the files configured at the same level. For example, you can change the search order only within the files configured at the profile level.
`Update Index`	If this check box is selected and if there is any modification in the model library file, on simulating the design, PSpice automatically generates a new index file for the model library.

Library Search Path

The *Library Path* text box displays the path that PSpice uses to search for models configured at the global level. You can edit or replace this path by typing a path in this text box, following these rules:

- Use a semicolon (;) to separate two path names.

@ 2023 All Rights Reserved.

- Do not follow the last path name with a semicolon.

Let us take an example.

To search first `C:\Cadence\SPB_17.4\tools\pspice\library` then `C:\MYLIBS`, type the following:

```
"C:\Cadence\SPB_17.4\tools\pspice\library";"c:\mylibs"
```

Include Files Settings

Use the *Configuration Files* tab of the *Simulation Settings* dialog box to configure and edit include files for simulation.

1. Click *Include* in the *Category* field in the *Configuration Files* tab to configure or edit the include files.

Click this button…	To do this…
Browse	Browse directories for files to include—the selected file name appears in the Filename text box
Add as Global	Add the file listed in the Filename text box to be used for all designs
Add to Design	Add the file listed in the Filename text box to be used for the current design only
Add to Profile	Add the file listed in the Filename text box to be used for the current profile only A file added at the design level cannot be added at the profile level also. Note: Profile specific configuration is available only if the project was created using Capture from release 10.0 and beyond. This feature is also available for the project created using older versions of Capture but converted to the Capture 10.0 format.
Edit	Open the selected file for editing in PSpice.
Change	Change the selected file with the file specified in the Filename text box.
	Remove the selected file from the Variant Files list—this does not delete the file from your computer

@ 2023 All Rights Reserved.

and	Move the selected file up or down in the Variant Files list.

Options for Simulation Profiles

Use the *Options* tab of the *Simulation Settings* dialog box to fine-tune how PSpice performs calculations for analog and gate-level digital simulation, as well as what information to save to the simulation output file (*.OUT).

From the *Category* list, select *Analog Simulation*, *Analog Advanced*, *Gate-level Simulation*, or *Output file* to display the settings for each option category.

Analog Simulation Options

Use the *Analog Simulation* settings to fine-tune analog simulation accuracy, set iteration limits, set operating temperature, and specify MOSFET parameters.

The option names in the text box correspond to the option names used in the PSpice .OPTIONS command.

Click this button…	To do this…
`General`	Enter values for speed level, tolerances, and minimum conductance.
`AutoConverge`	Suggest relaxed limits for various options that PSpice can modify during a simulation to achieve convergence.
`MOSFET options`	Enter values for the default drain area, default source area, default length, and default width.
`Advanced options`	Enter values for the total transient iteration limit, relative magnitude for matrix pivot, and absolute magnitude for matrix pivot.

@ 2023 All Rights Reserved.

The following tables defines all the options in the tab for the *Analog Simulation* category:

Flag option	Meaning
ADVCONV	Enables all convergence algorithms, such as Pseudo Tran, STEPGMIN, and step sources. ON by default. Setting this option will increase iterations.
AutoConverge	Suggest relaxed limits for various options that PSpice can modify during a simulation to achieve convergence.
Restart	Restart the convergence calculation
PREODER	Presorts the matrix diagonal by Markowitz counts.

Options	Description	Units	Default
`General section..`			
`SPEED_LEVEL`	increases simulation performance by optimizing switching behavior of models. If increase in simulation performance is not needed, set SPEED_LEVEL=0.	-	3
`RELTOL`	relative accuracy of V and I		0.001
`VNTOL`	best accuracy of voltages	volt	1.0 uV
`ABSTOL`	best accuracy of currents	amp	1.0 pA
`CHGTOL`	best accuracy of charges	coulomb	0.01 pC
`GMIN`	minimum conductance used for any branch	ohm^{-1}	1.0E-12
`Auto Converge section..`			
`ITL1`	Maximum number of iterations of convergence calculation during bias point analysis		150.0
`ITL2`	Maximum number of iterations of convergence calculation during DC analysis		20

@ 2023 All Rights Reserved.

Options	Description	Units	Default
ITL4	Maximum number of iterations of convergence calculation at each time step during transient analysis		10
TNOM	Nominal Temperature value	Celsius	27 degrees Celsius
THREADS	When THREADS=0, PSpice uses the number of threads that are equal to the number of cores detected in the system. When THREADS is anything other than 0, PSpice uses the exact number of threads as specified. It is recommended to set THREADS to 0 for PSpice to decide the optimal number of threads to be used.		0
PIVTOL	Specifies the absolute magnitude required for pivot in matrix solution		1.0E-13
MOSFET Section..			
DEFAD	Specifies the MOSFET default drain area (AD).	$meter^2$	0.0
DEFAS	Specifies the MOSFET default source area (AS).	$meter^2$	0.0
DEFL	Specifies theMOSFET default length (L).	meter	100.0 u
DEFW	Specifies theMOSFET default width (W).	meter	100.0 u

Setting Up Auto Convergence

You can suggest relaxed limits for various options used for simulation. The PSpice engine can select one or more of the options with relaxed limits and change their values during the simulation to converge it.

To set the relaxed limits and specify auto convergence, do the following:

@ 2023 All Rights Reserved.

1. From the *Options* tab of the *Simulation Settings* dialog box, click the *Analog Simulation* tree structure.

2. Select *Auto Converge* in the left panel.
3. Select the *AutoConverge* check box.
4. You can check the options for which you want to set relaxed limit and specify the limit value. For example, if the DC bias bind and iteration limit (ITL1) is set to `150`, you can relax the value to `1000`.

Although you can select any number of options, PSpice Engine decides what options to be changed during simulation.

Caution

You cannot specify a value that is less relaxed then the normal limit. For example, if the limit for ITL1 is set to 150, you cannot specify 120 as the relaxed limit.

1. You can select the *Restart* option to pause PSpice A/D if convergence is not achieved at the end of the simulation time.

   Note: `Restart` is selected by default.

2. Click *OK*.

Note: You can click *Reset* to change limits to the default values.

@ 2023
All Rights Reserved.

When you run the simulation with AutoConvergence set, PSpice A/D initially runs using the normal values for the specified simulation time. However, if the simulation does not converge, PSpice A/D changes the values within the relaxed limit for the parameters selected in the *Autoconvergence* Option.

Note that the *Autoconverge* can also be set from the PSpice Runtime Settings dialog box. Where, you can also set *Enable Advanced Convergence Algorithms*.

If convergence is not achieved at the end of the complete run, PSpice A/D starts simulation with an initial relaxed value if you the *Restart* option is selected.

Note that the PSpice A/D also changes the minimum time step size during a simulation to achieve convergence. Although PSpice A/D starts with the default minimum time step size, it can heuristically decide on a smaller time during the simulation.

Tip

You can specify the relative factor by which the minimum time step size is changed by setting the value of Relative factor for minimum delta (DMFACTOR) in the General node of the Advanced Analog tree structure. You can set any value that is a factor of 10 and is less than or equal to 1, such as .1, .001, or .0001. For example, if the default value of the minimum time step size is 10^{-18} , you can specify `DMFACTOR` as `.1`. This will result in a minimum time step of 10^{-19} .

Setting Autoconvergence from the Runtime Settings Dialog Box

The *RunTime Settings* dialog box allows you to set autoconvergence options during a simulation run. From this dialog box, you can select:

- *Autoconvergence*
- *Enable Advanced Convergence Algorithms*

To enable autoconvergence:

1. Select *Autoconvergence*.
2. You can either use the default relaxed limits for parameters or click *Settings* and change the limits.

To enable advanced convergence algorithms:

1. Select *Transient and Bias Point* options in the *Analog Advanced* tree structure of the *Options* tab.

@ 2023 All Rights Reserved.

2. Set the options.

The options are enabled according to the running simulation. For example, in the Bias Point mode, the GMIN, Source-stepping and Pseudo-Tran options are available, while in the Transient mode, the TRTOL and METHOD options are available.

@ 2023
All Rights Reserved.

Advanced Analog Options

You can use the analog advanced tree structure to set various analog simulation parameters, which are useful to:

- Improve convergence for both bias point and transient analyses
- Assist debugging of convergence failure by printing simulation data
- Improve performance and remove overflow errors
- Control simulation parameters

@ 2023 All Rights Reserved.

- Improve accuracy of simulation results.

@ 2023
All Rights Reserved.

The Analog Advanced tree structure displays default values for all the options and on checking some flag options - such as DIODERS, CSHUNT, BJTCJ, and DIODECJO, recommended values are also displayed.

Note: You can access the *Advanced Analog* Options from the *Options* tab of the *Simulation Settings* dialog box.

Use the *Analog Simulation* settings to enter values for the total transient iteration limit, relative magnitude for matrix pivot, and absolute magnitude for matrix pivot.

The option names shown in the text box correspond to the option names used in the PSpice .OPTIONS command.

Click this…	To do this…
General	Enter values for speed level, tolerances, and minimum conductance.
Bias Point	Suggest relaxed limits for various options that PSpice can modify during a simulation to achieve convergence.
Transient	Enter values for the default drain area, default source area, default length, and default width.

@ 2023
All Rights Reserved.

The following tables defines all the options in the tab for the *Analog Advanced* category:

Flag Option	Description
General	
NOGMINI	Specifies not to add GMIN across current sources. Not selected by default.
BRKDEPSRC	Sets automatic break-points for behavioral sources. Not selected by default.
CONVAID	Generates .1OP file for debugging purpose when convergence fails.Not selected by default.
Bis Point	
STEPGMIN	Enables GMIN stepping. This causes a GMIN stepping algorithm to be applied to circuits that fail to converge. GMIN stepping is applied first, and if that fails, the simulator falls back to supply stepping. Not selected by default.
NOSTEPSRC	Do not run source stepping algorithm for bias point convergence. Not selected by default.
NOSTEPDEP	Do not step dependent sources during source stepping algorithm for bias point convergence. Not selected by default.
GMINSRC	Enables step GMIN inside source-stepping. Not selected by default.
PSEUDOTRAN	Uses Pseudo-Transient Method. Not selected by default.
Transient	
TRANCONV	Enables continuation methods. Not selected by default.

Options	Description	Units	Default
General			
ITL5	Specifies the total repeating limit for all points for transient analysis. The default value of 0 specified infinity.	-	0.0
PIVREL	Specifies the relative magnitude required for pivot in matrix solution.	-	1.0E-3

@ 2023 All Rights Reserved.

PIVTOL	Specifies the absolute magnitude required for pivot in matrix solution	-	1.0E-13
SOLVER	Specifies the performance package solution algorithm. Select from default, 0, or 1. 0 selects the original solution algorithm and 1selects the advanced solution algorithm.	-	default
DMFACTOR	Sets the relative factor for minimum delta. The value specifies the relative value by which the minimum time step size is changed. The value should be less than or equal to 1 and a factor of 10, such as .1, .001, or .0001.		1
WCDEVIATION	Specifies the worst case deviation. It can have double values between 0 and 1.	-	0 (Same as RELTOL)
LIMIT	Specifies the absolute voltage limit. The default, 0, specifies that there is no limit on data values. You can modify it to a large value, such as 1e12, to eliminate overflow errors, especially when using exponential sources.	-	0
DIODERS	Sets the minimum value for Diode ohmic resistance (Rs). Not selected by default.	ohm	0
DIODECJO	Sets the minimum value for Diode junction capacitance (Cjo). Not selected by default.	ohm	0
BJTCJ	Specifies the minimum value for BJT Base-collector zero-bias depletion capacitance (Cjc), Base-emitter zero-bias depletion capacitance (Cje), and zero-bias collector substrate capacitance (Cjs). Not selected by default.	farad	0
Bias Point			
GMINSTEPS	Specifies the GMIN stepping size in integer (any positive value). Set to 0 for engine default.	-	0 (Same as ITL1)
ITL6	Specifies the number of steps of the source stepping algorithm. Can have any positive integer value. Set to 0 for engine default.	-	0 (Same as ITL1)

@ 2023 All Rights Reserved.

PTRANSTEP	Specifies the number of steps for a pseudo transient analysis to find the operating point. Can be any positive integer value. Set to 0 for engine default.	-	0 (Same as ITL1)
Transient			
METHOD	Specifies the integration method (values can be either TRAPEZOIDAL or GEAR)	-	-
TRTOL	Specifies the tolerance for integration error calculated using transient analysis. It is a relative tolerance where a higher TRTOL value results in bigger time steps and reduced accuracy. The TRTOL value should NOT be greater than 1/RELTOL.	-	7
CSHUNT	shunt capacitance added from all nodes of the design to GND. Recommended value is 1pF. Not selected by default.	farad	
TRANCONV	Enables alternate path search if transient simulation fails.		
IGMOS	Sets stringent convergence criteria for MOSFET devices. Set to overcome gate current inaccuracies when step size is small. Not selected by default.		
DEBUG	Set the DEBUG mode for report of diagnostic data. Select 0 for no diagnosis. Select 1 for a diagnostic Excel sheet with accepted/rejected time points. Select 2 for a diagnostic Excel sheet with accepted/rejected time points along with iterations data in CSV (comma-separated value) format. Select 3 for diagnostic data in the CSV format. **Note:** In case of convergence error for DEBUG mode 2 and 3, CSV file will bot be created. To generate the CSV file, you will have to stop the simulation.		0

@ 2023 All Rights Reserved.

DEBUG_START	Specify the start time for diagnostic report data. Default value is 0s.
DEBUG_END	Specify the end time for diagnostic report data. Default value is 1us.

Setting up Solver Algorithms

You can specify solver algorithms for performance package using the PSpice Advanced Analog option. PSpice A/D includes two solution algorithms: Solver 0 and Solver 1.

To choose a solver setting, do the following:

1. From the design entry tool's *PSpice* menu, choose *Edit Simulation Profile*.

   Alternatively, choose *Edit Profile* from the *PSpice Simulation* menu.

   The *Simulation Settings* dialog box appears.

2. Click the *Options* tab, then click the *Analog Advanced* tree structure.
3. Select the *General* option in the *Analog Advanced* tree.

4. In the *Solver* drop-down list, select one of the following choices:

   - 0 - the PSpice simulation engine uses the original PSpice solution algorithm.

@ 2023 All Rights Reserved.

- 1 - the PSpice simulation engine uses an advanced solution algorithm that provides significant speed improvements. This is default selection.

  Solver 1 is particularly useful for larger MOS and bipolar circuits with substantial runtimes.

5. Click *OK*.

Gate-level Simulation Options

Use the *Gate-level Simulation* settings to set timing, I/O levels for interfaces, drive strength, and error message limits.

Click this button…	To do this…
`General`	Enter values to set delay or initial state in flip-flops or latches.
`Advanced options`	Enter values for the minimum output drive resistance, maximum output drive resistance, overdrive ratio, default delay calculation, and error message limits.

The following tables defines all the options in the tab for the *Gate Level Simulation* category:

Flag Option	Description
`NOPRBMSG`	Suppresses simulation error messages in Probe data file.

`DIGMNTYMX`	default delay selector: 1=min, 2-typical, 3=max, 4=min/max	-	2.0
`DIGINITSTATE`	sets initial state of all flip-flops and latches in circuit: 0=clear, 1=set, 2=X	-	2.0
`DIGIOLVL`	default digital I/O level: 1-4;	-	1.0
`DIGDRVF`	minimum drive resistance (Input/Output UIO type model, DRVH (high) and DRVL (low) values)	ohm	2.0
`DIGDRVZ`	maximum drive resistance (UIO type model, DRVH and DRVL values)	ohm	20K

@ 2023 All Rights Reserved.

DIGOVRDRV	ratio of drive resistances required to allow one output to override another driving the same node	-	3.0
DIGMNTYSCALE	scale factor used to derive minimum delays from typical delays	-	0.4
DIGTYMXSCALE	scale factor used to derive maximum delays from typical delays	-	1.6
DIGERRDEFAULT	default error limit per digital constraint device	-	20.0
DIGERRLIMIT	maximum digital error message limit	-	0

Output File Options

Use the *Output File* settings to select the types of information PSpice saves to the simulation output file.

The following tables defines all the options in the tab for the *Output File* category:

Flag Option	Description
ACCT	Summary and accounting information is printed at the end of all the analyses.
EXPAND	Lists devices created by subcircuit expansion and lists contents of the bias point file.
LIBRARY	Lists lines used from library files.
LIST	Lists a summary of the circuit elements (devices).
NOBIAS	Suppresses the printing of the bias point node voltages.
NODE	Lists a summary of the connections (node table).
NOECHO	Suppresses a listing of the input file(s).
NOMODE	Suppresses listing of model parameters and temperature updated values.
NOOUTMSG	Suppresses simulation error messages in output file.
NOPAGE	Suppresses paging and the banner for each major section of output.
OPTS	Lists values for all options.
NUMDGT	Number of digits in printed values. This is 4 by default.

@ 2023 All Rights Reserved.

Flag Option	Description
WIDTH	same as the .WIDTH OUT= statement (can be set to either 80 or 132) Default is 80.

ACCT

Points to be noted when simulating a design with the ACCT option enabled:

- PSpice uses an industry-standard sparse matrix solver, and a number of these terms are specific to the way sparse solvers compute matrix inverse.
- All of these parameters get fixed during circuit setup, and don't get changed during simulation.
- Fill-in is the term used for entries that change from an initial zero to a non-zero value during the execution of an algorithm (matrix solve).
- Sparse matrix solvers use LU decomposition as an intermediate step, to compute inverse efficiently. LU decomposition algorithms make use of exchange of rows and columns to minimize fill-ins.

@ 2023 All Rights Reserved.

The following table shows some of the matrix and analysis statistics shown in the output file when simulating a design with ACCT option enabled:

Statistics	Description
NSTOP	Circuit matrix is conceptually of dimension NSTOP x NSTOP. The Sparse matrix storage and computational complexity depends on count of actual matrix entries, and not matrix size.To map to a circuit, Matrix size relates to number of nodes in a circuit (NSTOP * NSTOP, NSTOP = number of nodes), while number of matrix entries is roughly proportional to the number of connections in the circuit (like no connection from N2->N3 in below diagram). The number of connections are typically much smaller than NSTOP * NSTOP.
NTTAR	NTTAR is the earlier saved value of NTTBR before any fill-ins (see definition of fill-in above).
NTTBR	NTTBR is the actual value used during simulation, since this also accounts for the fill-ins added during matrix setup.

@ 2023 All Rights Reserved.

Statistics	Description
NTTOV	Number of terms (locations) in circuit matrix that come from more than 1 device (also called overflows). Example, For this circuit, Node N1 is connected to 4 resistors. All 4 resistors will update the values at matrix locations (N1,N2), (N1, 0) and (N1, N3). For R1 - matrix stamps are for locations (N2,N1), (N2, N2), (N1, N2), (N1, N1) For R2 - matrix stamps are for locations (N1, 0), (0, 0), (0, N1), (N1, N1) So, Location (N1, N1) comes from more than 1 device.
IFILL	Difference between NTTAR and NTTBR. Basically, Number of fillins.
IOPS	LUOperationCount - Number of multiplication/division operations needed for 1 LU decomposition.
PERSPA	Percent sparsity of circuit matrix, which is the number of zero-valued elements divided by the total number of elements in a matrix. This depends on circuit connectivity, and gives a pointer towards complexity of sparse solver operations.
NUMTTP	Sum of accepted and rejected time points. This is the count of total time points actually evaluated, so gives a better indication towards performance than only the accepted time points.

@ 2023 All Rights Reserved.

Statistics	Description
NUMRTP	Number of rejected time points. Time points may get rejected due to local truncation error, or convergence error in PSpice inner iteration. When a time point gets rejected, current DELTA is reduced to DELTA/8, and simulation is run on the new time point.
NUMNIT	Total number of inner iterations for all time points. For each iteration, matrix solver runs decomposition and solution of matrix.

@ 2023 All Rights Reserved.

trans.dat (active) | trans.out.1

```
103
104
105  Node counts:
106    Top level (NUNODS)            =        3
107    External (NCNODS)             =        3
108    Total (NUMNOD)                =        4
109
110  Total device count (NUMEL)      =        3
111    Diodes (D)                    =        1
112    Resistors (R)                 =        1
113    Voltage Sources (V)           =        1
114
115  Number of subcircuits (X)       =        0
116
117  Matrix statistics:
118    Matrix size (NSTOP)           =        5
119    Initial no. elements (NTTAR)  =        9
120    No. elements w/ fillin (NTTBR) =       9
121    No. fillins (IFILL)           =        0
122    No. overflows (NTTOV)         =        1
123    No. LU operations (IOPS)      =        4
124    Percent sparsity (PERSPA)     =   64.000
125
126  Analysis statistics:
127    No. total time points (NUMTTP) =    14415
128    No. rejected time points (NUMRTP) =   833
129    No. iterations (NUMNIT)       =    49220
130
131  Size of Probe file              =  1610445
132  Load Threads                    =        1
133
134  Runtime statistics:                Seconds     Iterations
135    Matrix load                   =      .03
136    Matrix solution               =     0.00            3
137    Readin                        =     0.00
138    General setup                 =      .08
139    CMI setup                     =     0.00
140    Setup                         =     0.00
141    DC sweep                      =     0.00            0
142    Bias point                    =     0.00            7
143    AC and noise                  =     0.00            0
144    Total transient analysis      =      .05
145    Transient analysis w/o bias point =   .05        49220
146    Output                        =     0.00
147    Overhead                      =      .91
148    Total job time (using Solver 1) =    .13
149
```

Related Topics

- .OPTIONS

@ 2023 All Rights Reserved.

Data Collection Options

Use the *Data Collection* tab of the *Simulation Settings* dialog box to restrict the captured simulation data. This is especially useful for large circuit designs that produce more data than you need for waveform analysis.

You can also set the probe data accuracy from this tab. By default, probe data has 64-bit accuracy but you can choose a lower 32-bit accuracy.

You can choose options and apply them to Voltages, Currents, Power, Digital or Noise data.

Setting Data Collection Options

One reason that waveform data files are large is that, by default, PSpice A/D stores all net voltages and device currents for each step (for example, time or frequency points). However, if you have placed markers on your schematic prior to simulation, PSpice A/D saves only the results for the marked wires and pins.

You can limit the file size by setting the collection options from the *Simulation Settings* dialog box.

To limit the file size, do the following:

1. Choose *PSpice – Edit Simulation Profile* to display the Simulation Settings dialog box.
2. Click the *Data Collection* tab.

@ 2023 All Rights Reserved.

3. In the *Data Collection Options* section, choose the desired option for each type of marker (Voltages, Currents, Power, Digital, Noise).

Option	Description
All	All data will be collected and stored. (This is the default setting.)
All but Internal Subcircuits	All data will be collected and stored except for internal subcircuits of hierarchical designs (top level data only).
At Markers Only	Data will only be collected and stored where markers are placed.
None	No data will be collected.

4. Select *Save data in the CSDF format (.CSD)* if you want the data to be stored in this format.

   Note: By default, the probe data has an accuracy of *64-bit*. You can change this to *32-bit*.

5. Click *OK* to close the Simulation Settings dialog box.
6. From the *PSpice* menu, point to Markers, then choose the marker type you want to place.
7. Point to the wires, pins or devices you wish to mark and click to place the chosen markers.
8. Right-click and select End Mode to stop placing markers.
9. From the *PSpice* menu, choose Run to start the simulation.

When the simulation is complete, the corresponding waveforms for the marked nodes or devices will be displayed in Probe.

Waveform Data File Format Option

Select the *Save data in the CSDF format (.CSD)* check box to save the waveform data file in the ASCII Common Simulation Data Format instead of in the default binary format.

Probe Data Accuracy Options

By default, the probe data has 64-bit accuracy. You can set the accuracy by selecting a different radio button for *Probe Data*. A 64-bit accuracy setting ensures that the output does

@ 2023 All Rights Reserved.

not have ramps. For example, when a small amplitude voltage is superimposed on a large amplitude voltage, the resolution might be lost, resulting in ramps.

Probe Windows Options

You can use the *Probe Windows* tab of the *Simulation Settings* dialog box to set up how Probe windows are displayed for a simulation profile.

Probe Window Display Options

Option	To do this…
Display Probe window when profile is opened	Display the Probe windows that were displayed the last time the profile was opened.
Display Probe window:	Display the Probe windows when:
■ during simulation ■ after simulation has completed.	■ the simulation is running, and update the waveforms as the simulation progresses. ■ the simulation is finished.
Show	Show the traces:
■ All markers on open schematics	■ For all the markers that are placed on currently open designs in the design entry program.
■ Last plot	■ That were used the last time the profile was opened.

Setting Up Probe Window Options

To set Probe window options, do the following:

1. From the *Tools* menu, select *Options* to display the *General* tab of the *Probe Settings* dialog box.
2. In the Use Symbols frame, choose one of the following options:
   - ❑ Attributes
   - ❑ Never
   - ❑ Always
3. In the Trace Color Scheme frame, choose one of the following options:

@ 2023 All Rights Reserved.

- ❑ Normal
- ❑ Match Axis
- ❑ Sequential Per Axis
- ❑ Unique By File

4. In the Use ScrollBars frame, choose one of the following options:

- ❑ Auto
- ❑ Never
- ❑ Always

5. In the Auto-Update Interval frame, choose one of the following options:

- ❑ Auto
- ❑ Every sec
- ❑ Every %

6. In the Number of Histogram Divisions text box, enter the number of histogram divisions used when PSpice A/D displays Performance Analysis of a Monte Carlo run.

7. In the Number of Cursor Digits text box, enter the number of digits PSpice A/D displays when the cursor position is shown.

8. Select any or all of the following options:

- ❑ Mark Data Points
- ❑ Display Evaluation
- ❑ Display Statistics
- ❑ Highlight Error States

9. Click *OK* to close the *Probe Settings* dialog box.

Starting a Simulation

After you have used a schematic capture applications to enter your circuit design and have set up the analyses to be performed, you can start a simulation by choosing *Run* from the *PSpice* menu. When you enter and set up your circuit this way, the schematic capture application automatically generates the simulation files and starts PSpice A/D.

@ 2023 All Rights Reserved.

There may be situations, however, when you want to run PSpice A/D outside of any schematic capture application. You may want to simulate a circuit that was not created in any schematic capture application, for example, or you may want to run simulations of multiple circuits in batch mode.

After you have set up the analyses for the circuit, you can start a simulation from a schematic capture application in either of the following ways:

- From the *PSpice* menu select *Run*.
- Click the *Simulate* button on the PSpice toolbar.

To start PSpice A/D outside of any schematic capture application such as Capture, do the following:

1. From the *Start* menu, point to the installed release, then choose *PSpice*.
2. From the *File* menu, choose *Open Simulation*.
3. Do one of the following:
   - Double-click on the simulation profile filename (`*.SIM`) in the list box.
   - Enter the simulation profile filename (`*.SIM`) in the File name text box and click Open.
4. From the *Simulation* menu, choose *Edit Settings* to modify any of the analysis setup parameters.
5. From the *Simulation* menu, choose *Run* (or click the Run toolbar button) to begin the simulation.

Batch Simulations

Multiple simulations can be run in batch mode when starting PSpice A/D directly with circuit file input. You can use batch mode to run a number of simulations overnight. Following are the two ways to do this:

- Multiple simulation setups within one circuit file
- Running simulations with multiple circuit files

@ 2023 All Rights Reserved.

Multiple Simulation Setups within One Circuit File

Multiple circuits description can be concatenated into a single circuit file and simulated simultaneously with PSpice A/D. Each circuit description in the file must begin with a title line and end with a `.END` statement.

The simulator reads all the circuits in the circuit file and processes each in sequence. The data file and simulation output file contain the outputs from each circuit in the same order as they appeared in the circuit file. The effect is the same as if you had run each circuit separately and concatenated all the outputs.

Running Simulations with Multiple Circuit Files

You can direct PSpice A/D to simulate multiple circuit files using either of the following methods:

Using Open Simulation Dialog Box

1. From the Start menu, launch PSpice A/D.
2. In the PSpice A/D window, select *Open Simulation* from the *File* menu.
3. Do one of the following:
   a. Type each file name enclosed in double quotation marks in the File Name text box separated by a space.
   b. Use the combination keystrokes and mouse clicks in the list box as follows: *Ctrl*+click to select file names one at a time, and *Shift*+click to select groups of files.

Using Command Line

1. From the Start menu, launch PSpice A/D.
2. Update the command line by including a list of circuit file names separated by spaces.

   Note: Circuit file names can be fully qualified or can contain the wild card characters (* and ?).

Viewing the Simulation Queue

Use the *Simulation Queue* dialog box to set the order of a batch simulation and to view the progress of a batch simulation.

@ 2023 All Rights Reserved.

To set up a batch simulation, do the following:

1. From the *File* menu, choose *Open Simulation*.
2. Select multiple .SIM or .CIR files by doing one of the following:
   - Press `CTRL` to select individual files.
   - Press `SHIFT` to select a range of files.

Note: If the files you want to select are located in different directories, click Add in the Simulation Queue dialog box to find them in the various directories. The *Simulation Queue* dialog box will appear after you click *Open*.

To set the batch simulation order, do the following:

1. After selecting the files for a batch simulation, click *Open*.

   The *Simulation Queue* dialog box appears. The file at the top of the *Pending Simulations* list is set to be simulated first.
2. To change the order, do one of the following:
   - Select a filename in the list and click either or to move it up or down.
   - Click the Reset button to reset the order.
   - Click Add to add more files to the simulation.
3. Click the *Settings* button to display the *Simulation Settings* dialog box, where you can view or change the simulation profile settings.
4. Click the Start button to start the simulations.

PSpice A/D Simulation Window

The PSpice A/D Simulation Window is an MDI (Multiple Document Interface) application. This implies that you can open and display multiple files at the same time in this window. For instance, you can have a waveform file (.DAT), a circuit file (.CIR), and a simulation output file (.OUT) open and displayed in different child windows within this one window.

The PSpice A/D Simulation Window consists of three sections: the main window section where the open files are displayed, the output window section where output information such as informational, warning, and error messages from the simulator are shown, and the simulation status window section where detailed status information about the simulation are shown. These three sections are shown in the following figure.

@ 2023 All Rights Reserved.

The windows in these sections may be resized, moved, and reordered as needed.

The simulation window also includes a menu bar and toolbars for controlling the simulation and the waveform display.

Title Bar

The title bar of the simulation window (the area at the top of the window) identifies the name of the currently open simulation (either simulation profile or circuit file) and the name of the currently active document displayed in the main window area. For example, the simulation window shown in the following figure indicates that simulation profile SCHEMATIC1-Transient is currently open and the active document displayed is Transient.DAT.

Menus and Toolbars

The menus accessed from the menu bar include commands to set up and control the simulator, customize the window display characteristics, and configure the way the waveforms are displayed. The toolbar buttons duplicate many of the more frequently used commands.

Figure 7-1 PSpice A/D simulation window

@ 2023
All Rights Reserved.

Main Window Section

The top central portion (by default) of the simulation window is the main window section where documents (such as waveforms, circuit description, output information etc.) are displayed within child windows. These windows are tabbed by default. The tabs at the bottom left show the names of the documents that each child window contains. Clicking on a tab brings that child window to the foreground. The following figure shows the tabbed document windows for Transient.DAT and Transient.OUT.

You can configure the display of these windows to suit your preferences and to make the analysis of the circuit quick and readily understandable. These windows can also be resized, moved, and reordered to suit your needs.

Output Window Section

The lower left portion of the simulation window provides a listing of the output from the simulation. It shows informational, warning, and error messages from the simulation. You can resize and relocate this window to make it easier to read.

Simulation Status Window Section

The lower right portion of the simulation window presents a set of tabbed windows that show detailed status about the simulation. There are three tabbed windows in this section: the Analysis window, the Watch Variable window, and the Devices window. The Analysis window provides a running log of values of simulation variables (parameters such as Temperature, Time Step, and Time). The Watch Variable window displays watch variables and their values. These are the variables setup to be monitored during simulation. The Devices window displays the devices that are being simulated.

@ 2023 All Rights Reserved.

Interaction with Simulation

PSpice A/D includes options for interacting with a simulation by changing certain runtime parameters in the course of the analysis. With the interactive simulation feature, you can do the following:

- Extend a transient analysis after TSTOP has been reached in order to achieve the desired results.
- Interrupt a bias or transient analysis, change certain runtime parameters, and then resume the simulation with the new settings.
- Schedule changes to certain runtime parameters so that they are made automatically during a simulation.

Note: The ability to interact with a simulation only applies to bias point and transient analyses. You cannot interact with other analysis types.

The following table identifies what interactive functionality is available with each version of PSpice A/D.

PSpice A/D version	Interactive Simulation Functionality
PSpice	Extend transient analysis
PSpice	Interrupt a simulation, change parameters, and resume the simulation
	Schedule automatic changes to parameters during simulation

Extending a Transient Analysis

Often, a long transient analysis will run to the completion time (TSTOP) without achieving the desired simulation results (achieving a steady state, for instance). To achieve better results, the value for TSTOP would have to be increased and the entire simulation would have to be rerun from the beginning. This was time-consuming and inefficient for large simulations.

A transient analysis will automatically pause rather than stop when it reaches the TSTOP value. Once paused, you can review the results and determine if the simulation should run

@ 2023 All Rights Reserved.

longer. If desired, you can increase the value of TSTOP and resume the transient analysis from the point at which it paused, thus saving a good deal of processing time.

To help clarify under what conditions simulations will either be terminated or paused, the following table explains the different behaviors of PSpice A/D for particular simulation scenarios:

Simulation Scenario	Behavior of PSpice
Running a single transient simulation using a profile or a circuit file containing one circuit.	PSpice A/D will pause after a successful simulation, or if a convergence error occurs, allowing you to change certain runtime parameters and resume the analysis.
Running a single AC or DC simulation using a profile or a circuit file containing one circuit.	PSpice A/D will stop (terminate) after a successful simulation. -or- PSpice A/D will pause if a convergence error occurs, allowing you to change certain runtime parameters and resume the analysis.
Running a single simulation with a profile or a circuit file containing outer loops.	PSpice A/D will stop (terminate) after a successful simulation, or if a convergence error occurs.
Running a queued simulation.	PSpice A/D will stop (terminate) after a successful simulation, or if a convergence error occurs.
Launching a new simulation when another one is already active in PSpice A/D.	If the old simulation has completed, PSpice A/D will load the new simulation and run it. -or- If the old simulation is running or paused, PSpice A/D will prompt you to choose whether to run the new simulation instead, place it in the queue or cancel it.

To extend a transient analysis, do the following:

@ 2023 All Rights Reserved.

1. After you pause a transient analysis, click in the Run For text box on the PSpice toolbar.

2. Enter a new value for TSTOP.
3. Click on the Run toolbar button to resume the simulation.

   The simulation will resume from the point at which it last paused, and then run for the amount of time specified in the RunFor text box, at which point it will pause again.

Note: Each time you resume the simulation after changing TSTOP, the transient analysis will always pause when completed. In this way, you can continue extending the analysis indefinitely. If the simulation is paused before TSTOP is reached, and you enter a value in the RunFor text box and click the Run toolbar button, PSpice A/D will run for the time specified and then pause. If you click the Run button and if the total time has not yet reached TSTOP, PSpice A/D will run until TSTOP. If you pause the simulation while it is in the middle of a RunFor operation and then resume the simulation, PSpice A/D will complete the RunFor operation. If you click the Run button while the simulation is paused in the middle of a RunFor operation, PSpice A/D will run until TSTOP is reached.

@ 2023 All Rights Reserved.

Interrupting a Simulation

In PSpice A/D, you have the ability to interrupt (pause) a simulation, change certain runtime parameters, and then resume the simulation from the point at which it was paused using the new parameters.

Note: The new parameters are temporary values and are not saved in the simulation profile. However, they are logged in the output file so that you can refer to them later.

When a simulation has been paused, you can change the following runtime parameters in the *Edit Runtime Settings* dialog box:

- RELTOL
- ABSTOL
- VNTOL
- GMIN
- TSTOP
- TMAX
- ITL1
- ITL2
- ITL4

The *PSpice Runtime Settings* dialog box will appear automatically whenever a simulation fails to converge. (In such cases, the simulation will be paused automatically.) It will also appear if you attach PSpice A/D to a simulation that was paused in the background.

To interrupt a simulation and change the parameter, you need to perform the following steps:

1. In PSpice A/D, from the *Simulation* menu, choose *Edit Runtime Settings*.

   The *PSpice Runtime Settings* dialog box appears.

2. If you want to use the original value for a particular parameter, click the *Use Original Value* check box for that parameter.

   The original parameter values are derived from the simulation profile. By default, the *Use Original Value* check boxes are checked (enabled).

3. If you want to change one or more parameters, enter new values for each of the runtime parameters you want to change in the text boxes under the column *Change To*.

@ 2023 All Rights Reserved.

If the *Change To* text box is grayed out, uncheck the *Use Original Value* check box.

4. Click *OK* and Resume Simulation to resume the simulation with the new parameters.

Note: Simulation is an iterative process in which previous states are used to calculate the current state. When you change a value and resume simulation, the previous state calculated with the original value affects the current state. If you want the changed values to become default values, use the .options command in the profile.

Schedule Changes to Runtime Parameters

You may want to predefine a set of values for a parameter and schedule these values to take effect at various time intervals during a long simulation. For instance, you may want to use a smaller time step value during periods where the input stimulus changes rapidly, but otherwise use a larger value.

You can set up automatic changes to certain runtime parameters that will occur at scheduled times during a simulation. By scheduling the changes, you don't have to interrupt the simulation manually, and can even run it in a batch mode in the background.

The following runtime parameters can be changed at scheduled times during a simulation. Note that these only apply to transient analysis; you cannot interact with other analysis types.

- RELTOL
- ABSTOL
- VNTOL
- GMIN
- ITL4

PSpice Command Syntax for Scheduling Parameter Changes

You can schedule parameter changes by entering them either in the Maximum Step Size text box in the Simulation Profile or in a text file using the new expression SCHEDULE, and then including that file in the simulation profile settings.

The expression SCHEDULE is a piecewise constant function (from time x forward use y) and takes the form:

$$\text{SCHEDULE}(x_1,y_1,x_2,y_2\ldots x_n,y_n)$$

where x is the time value, which must be x >= 0, and y is the value of the associated parameter. You must include an entry for time = 0.

@ 2023 All Rights Reserved.

When used with the .OPTIONS command, the syntax is as follows:

> .OPTIONS <Parameter Name>= {SCHEDULE(<time-value>, <parameter value>, <time-value>, <parameter value>, …)}

For example,

.OPTIONS RELTOL={SCHEDULE(0s,.001,2s,.005)}

indicates that RELTOL should have a value of 0.001 from time 0 up to time 2s, and a value of 0.005 from time 2s and beyond (that is: RELTOL=.001 for t, where 0 <= t < 2s, and RELTOL=.005 for t, where t >= 2s).

Scheduling Runtime Parameters Changes

To schedule changes to runtime parameters, do the following:

1. Open a standard text editor (such as Notepad) and create a text file with the command syntax shown above, using the appropriate values for the different parameters.
2. In a schematic capture application, open the design you want to simulate.
3. From the *PSpice* menu, choose *Edit Simulation Profile*.

   The *Simulation Settings* dialog box appears.
4. Click on the *Configuration* Files tab
5. Click Include in the *Category* field to display the Include files list.
6. Under the Filename text box, enter the name of the text file you created in Step 1, or click the Browse button to locate the file and enter the full path and filename.
7. Click the *Add to Design* button to include the file as part of the circuit.
8. Click *OK*.

   When you run the simulation, the scheduled parameter changes will be included as part of the circuit file and the simulation will run to completion automatically.

Related Topics

- Analog Simulation Options
- Schedule Changes to Runtime Parameters
- Setting Autoconvergence from the Runtime Settings Dialog Box

@ 2023 All Rights Reserved.

Using the Simulation Manager

PSpice A/D includes a new Simulation Manager that provides enhanced control over how multiple simulations are processed. You can preempt the current simulation to run another one first or, you can use the Simulation Manager to monitor the progress of a set of batch simulations that were set up and launched earlier.

Note: None of the earlier functionality of batch processing has been lost.

The PSpice Simulation Manager provides a familiar, easy-to-use interface for controlling how multiple simulations are processed.

The Simulation Manager allows you to do the following:

- add or delete simulations
- start, stop or pause simulations
- rearrange the order of the simulations in the queue
- attach PSpice to a simulation to make it the active display
- view the status and progress of simulations running in the background

You can accomplish most of these functions by selecting the desired simulation in the list, then clicking on the appropriate toolbar button to execute the command.

Note that for simulations that are queued in the Simulation Manager, the setting in the Simulation Profile to start probe automatically is ignored. When a queued simulation runs to completion and finishes, it will not be loaded into probe. You must do this manually if you want to see the results of that simulation.

@ 2023 All Rights Reserved.

Access to Simulation Manager

The Simulation Manager is invoked whenever you start a new simulation, either from PSpice A/D or from a front-end design entry tool. Since it is active as long as a simulation is running in the background, you can also call up the Simulation Manager from the Windows system tray.

You can also launch the Simulation Manager by itself from the Windows Start menu. You do not need to have PSpice A/D running in order to work with the Simulation Manager.

Information in the Simulation Manager

Every job listed in the Simulation Manager will have a specific entry for Schedule, Status and Percent Complete. In addition, certain color-coded icons are shown to the left of each simulation file name to indicate their current state. A quick glance over the list of jobs will tell

@ 2023 All Rights Reserved.

you immediately where any particular job is and how it will be processed. The following tables explain the meanings of the various categories and states.

Icon	Explanation
	The simulation is either in the queue and has not been run yet, or has been run to completion.
	The simulation is currently running.
	The simulation has been paused and is on hold, waiting to either be continued or stopped.
	The simulation has been stopped and is not completed.

Schedule	Explanation
queued	The simulation is in the queue. It will be run in the order in which it is listed in the queue. (This is the default setting.)
running	The simulation is currently running and ongoing status information is displayed.
on hold	The simulation has been paused.
stopped	The simulation has been run completely, or was stopped because of an error. **Note:** You must manually restart a stopped simulation if you want it to run again at a later time.

Status	Explanation
not run	The simulation has not been started yet. (This is the default setting.)

@ 2023 All Rights Reserved.

<status>	Basic status information about the progress of the analysis will be displayed for a simulation that is currently running.
paused	The simulation has been paused either manually or automatically by the Simulation Manager. **Note:** If you change the default option that automatically resumes paused simulations in the queue, then you must remember to manually resume a paused simulation if you want it to continue at a later time.
complete – no errors	The simulation has run to completion and no errors were encountered.
errors	The simulation ran partially but stopped automatically because errors were encountered.

Percent	**Explanation**
<%>	The percentage of completion for a simulation. This number increases as a simulation progresses.

What the various versions of PSpice A/D support

The following table identifies what functionality in the Simulation Manager is available with each version of PSpice A/D.

PSpice A/D version	**Functionality of Simulation Manager**
PSpice	One simulation may be running and multiple simulations may be paused.
PSpice	The queue is run sequentially.

@ 2023 All Rights Reserved.

Error Conditions During Simulation

Since each simulation that runs in the background runs independently, an error that occurs during one simulation will not prevent the remaining jobs in the queue from running subsequently, in order.

The following common error conditions may arise, but these will not prevent the Simulation Manager from running the remaining simulations pending in the queue.

- Simulation crash: If a simulation crashes for whatever reason, the Simulation Manager will stop receiving progress updates. After a certain period, the Simulation Manager will stop that simulation and will automatically start the next job in the queue.
- Simulation pause: If a simulation pauses automatically and requires manual intervention to continue, the Simulation Manager will automatically start the next job in the queue.
- Simulation stop: If a simulation stops automatically, the Simulation Manager will automatically start the next job in the queue.

Note: You must manually restart a stopped or paused simulation if you want it to run again at a later time. You will not be able to shut down the Simulation Manager until all stopped and paused simulations have been deleted.

Related Topics

- Batch Simulations

Setting up Multiple Simulations

With the Simulation Manager, you can set up any number of batch simulations to be run sequentially in the background while you do other work in PSpice A/D. Each new simulation that you set up will be added to the bottom of the simulation queue and will be assigned the schedule category “queued”. It will be run after all other queued jobs ahead of it have been run.

Once a job has been added, you can change its position in the queue, start, stop or pause it, or make other modifications to its status.

To add a simulation to the queue, do the following:

1. From the *File* menu, choose *Add Simulation* or click the *Add Simulation* button on the toolbar.
2. Locate the file (.SIM, .CIR) you wish to add to the queue.

@ 2023 All Rights Reserved.

Alternately, you can add a simulation to the queue by starting the PSpice simulation directly from within the front-end tool you are using, such as OrCAD Capture.

Note: If one simulation is already running in the Simulation Manager and you start another one, you will be prompted to direct the Simulation Manager in how to proceed with the new simulation.

Related Topics

- Setting Options in the Simulation Manager

Starting, Stopping, and Pausing Simulations

In the Simulation Manager, you can easily manage the various batch simulations in the queue. The most fundamental controls that are provided are the ability to start a simulation, stop it, or pause it temporarily.

To start a simulation from the Simulation Manager

1. Select a simulation in the list.
2. From the *Simulation* menu, choose Run or click the Run Selected button on the toolbar.

To stop a simulation from the Simulation Manager

1. Select the simulation that is currently running.
2. From the *Simulation* menu, choose Stop or click the Stop Selected button on the toolbar.

To pause a simulation from the Simulation Manager

1. Select the simulation that is currently running.
2. From the *Simulation* menu, choose Pause or click the Pause Selected button on the toolbar.

Attaching PSpice A/D to a Simulation

A simulation that is running in the Simulation Manager will not be loaded into PSpice A/D or displayed in Probe while it is running. This allows you to work on a different design in the PSpice A/D application while a simulation is running in the Simulation Manager.

@ 2023 All Rights Reserved.

Note: If you start a new simulation from within PSpice A/D while another is running in the queue in the Simulation Manager, the Simulation Manager must decide how to treat the new job. You will be prompted to choose whether you want the new job to preempt the current simulation and start running immediately.

If you want to display a different simulation in PSpice A/D by choosing from the list of jobs in the Simulation Manager, you can attach PSpice A/D to a particular job in the queue.

To attach PSpice A/D to a simulation, do the following steps:

1. Select the simulation you want to attach PSpice A/D.
2. From the *View* menu, choose *Simulation Results*.

   The PSpice A/D program will activate and the results of the simulation you selected will become the current display in Probe. If the simulation is currently running, you will be able to view the marching waveforms.

Setting Options in the Simulation Manager

Each time you add a new simulation while another is running, the Simulation Manager must decide how to treat the new job. The default setting is to add the new simulation to the bottom of the queue and continue running whatever job is currently being simulated.

You can change this default so that the Simulation Manager will start each new simulation immediately and either stop or pause whatever job is currently running. The options you can choose from are explained in the procedure.

You can also choose to have the Options dialog box display each time you add a new simulation, or not show this anymore. If you disable the prompting, you can always enable it again using the following procedure.

In addition, you can define how paused simulations should be handled by the Simulation Manager. You can configure them to be resumed automatically after the previous simulation stops, or you can choose to leave them in a paused state until you manually resume them.

To set the default options for the Simulation Manager, do the following steps:

1. From the *Tools* menu, choose *Options*.

   The *Options* dialog box appears.

@ 2023 All Rights Reserved.

2. In the top frame dealing with simulations that are already running, click the appropriate radio button for the option you wish to set.

Radio Button	Function
`Display the simulation in the queue.`	The simulation that is currently running will be displayed in PSpice. The new simulation will be added to the bottom of the queue and will be run after all other jobs in the queue have been run. (This is the default setting.)
`Pause the current simulation and run the new one.`	The simulation that is currently running will be paused. The new simulation will be started immediately. You must remember to resume the paused simulation later if you want it to continue.
`Stop the current simulation and run the new one.`	The simulation that is currently running will be stopped. The new simulation will be started immediately. You must remember to restart the stopped simulation later if you want it to run again.

@ 2023 All Rights Reserved.

3. If you want the *Options* dialog box to appear as a reminder each time you add a new simulation, be sure to check the Always Prompt box. (The default setting is to enable this feature.)

4. In the bottom frame dealing with paused simulations, click the appropriate radio button for the option you wish to set.

Radio button	Function
`Resume simulating.`	The first paused simulation in the list will automatically resume after the previous simulation has stopped. (This is the default setting.)
`Wait for user intervention.`	The Simulation Manager will not resume any paused simulations automatically. You must intervene manually to resume them. **Note:** If you enable this radio button, you must remember to intervene manually if you want paused simulations to resume later.

5. Click *OK* to save the settings.

@ 2023 All Rights Reserved.

@ 2023 All Rights Reserved.

8

DC Analyses

The DC analyses evaluate circuit performance in response to a direct current source. The following table summarizes what PSpice A/D calculates for each DC analysis type.

DC Analysis	PSpice A/D Computes
DC sweep	Steady-state voltages, currents, and digital states when sweeping a source, a model parameter, or temperature over a range of values
Bias point detail	Bias point data in addition to what is automatically computed in any simulation
DC sensitivity	Sensitivity of a net or part voltage as a function of bias point
Small-signal DC transfer	Small-signal DC gain, input resistance, and output resistance as a function of bias point

To calculate the DC response of an analog circuit, PSpice A/D removes time from the circuit. This is done by treating all capacitors as open circuits, all inductors as shorts, and using only the DC values of voltage and current sources. A similar approach is used for digital devices. All propagation delays are set to zero, and all stimulus generators are set to their time-zero values.

To solve the circuit equations, PSpice A/D uses an iterative algorithm. For analog devices, the equations are continuous, and for digital devices, the equations are Boolean. If PSpice A/D cannot get a self-consistent result after a certain number of iterations, the analog or digital devices are forced to the X value, and more iterations are done. Since X as input to a digital component gives X as output, the Boolean equations can always be solved this way.

If a digital node cannot be driven by known values during the DC iterations (for instance, the output of a flip-flop with the clock line held low), then its DC state will be X. Depending on the circuit, some, none, or all of the digital nodes may have the state X when the bias point is calculated.

@ 2023 All Rights Reserved.

Related Topics

- DC Sweep
- Bias Point
- Small-Signal DC Transfer
- DC Sensitivity

@ 2023 All Rights Reserved.

DC Sweep

The DC sweep analysis causes a DC sweep to be performed on the circuit. DC sweep allows you to sweep a source (voltage or current), a global parameter, a model parameter, or the temperature through a range of values. The bias point of the circuit is calculated for each value of the sweep. This is useful for finding the transfer function of an amplifier, the high and low thresholds of a logic gate, and so on.

For the DC sweep analysis specified in the following figure, the voltage source V1 is swept from -0.125 volts to 0.125 volts by steps of 0.005. This means that the output has (0.125 + 0.125)/0.005 +1 = 51 steps or simulation points.

A source with a DC specification (such as VDC or IDC) must be used if the swept variable is to be a voltage type or current source. You can set the DC value from *Properties* of the *Edit* menu of your schematic capture tool. You must click on the cell under the DC column and type in its value.

The default DC value of V1 is overridden during the DC sweep analysis and is made to be the swept value. All the other sources retain their values.

@ 2023 All Rights Reserved.

After running the analysis, the simulation output file (`EXAMPLE.OUT` for the `EXAMPLE.OPJ` circuit in the following figure) contains a table of voltages relating V1, node OUT1, and node OUT2.

Figure 8-1 Example schematic EXAMPLE.OPJ.

Note: The example circuit EXAMPLE.OPJ is provided with the program installation.

The following table summarizes what are the minimum requirements to run a DC sweep analysis:

Swept Variable Type	Requirement
voltage source	voltage source with a DC specification (VDC, for example)
temperature	none
current source	current source with a DC specification (IDC, for example)
model parameter	PSpice A/D model (.MODEL)
global parameter	global parameter defined with a parameter block (.PARAM)

@ 2023 All Rights Reserved.

Setting up Simulation Profile to Run DC Sweep

Before you run a DC Sweep analysis, you need to set up the simulation profile using the following steps:

1. In your schematic capture tool, select *New Simulation Profile* or *Edit Simulation Profile* from the *PSpice* menu. If this is a new simulation, enter the name of the profile and click *OK*.

   The *Simulation Settings* dialog box appears.

2. Under *Analysis Type*, select *DC Sweep*.
3. For the *Primary Sweep* option, enter the necessary parameter values and select the appropriate check boxes to complete the analysis specifications.
4. Click *OK* to save the simulation profile.
5. Select *Run* under the *PSpice* menu to start the simulation.

Note that you do not require to specify a DC sweep and a parametric analysis for the same variable.

Note: To run a small-signal DC transfer analysis, you need to place and connect one or more independent sources and then set the DC voltage or current level for each source.

@ 2023
All Rights Reserved.

Setting up a DC Stimulus

To set up a DC stimulus, do the following:

1. Place and connect one of these symbols in your schematic:

For Voltage Input	
Input	**Analysis Type**
VDC	A DC Sweep and/or Bias Point (transfer function) analysis only.
VSRC	Multiple analysis types including DC Sweep and/or Bias Point (transfer function).

For Current Input	
Input	**Analysis Type**
IDC	A DC Sweep and/or Bias Point (transfer function) analysis only.
ISRC	Multiple analysis types including DC Sweep and/or Bias Point (transfer function).

2. Double-click the symbol instance to display the Parts spreadsheet appears.
3. Click in the cell under the DC column to edit its value.
4. Define the DC specification as shown in the following table:

Attribute	Value
DC	*DC_level* where *DC_level* is in volts or amps (units are optional).

5. Click *OK* twice to exit the dialog boxes.

Setting up a Nested DC Sweep

After a primary sweep value, a second sweep variable can be specified in the *DC Sweep* dialog box. When you specify a secondary sweep variable, it forms the outer loop for the

@ 2023 All Rights Reserved.

analysis. That is, for every increment of the second sweep variable, the first sweep variable is stepped through its entire range of values.

To set up a nested sweep, do the following:

1. Under *Options*, select the *Secondary Sweep* box for the *DC Sweep* Analysis type.
2. Enter the necessary parameter values and select the appropriate check boxes to complete the analysis specifications.

Curve Families for DC Sweeps

When a nested DC sweep is performed, the entire curve family is displayed. That is, the nested DC sweep is treated as a single data section or you can think of it as a single PSpice A/D run.

@ 2023
All Rights Reserved.

Let us take an example of the circuit as shown in the figure to set up a DC sweep analysis with an outer sweep of the voltage source VG and an inner sweep of the voltage source VD as listed in the table.

Figure 8-2 Curve family example schematic.

	Primary Sweep	**Secondary Sweep**
Swept Variable Type	Voltage Source	Voltage Source
Sweep Type	linear	linear
Name	VD	VG
Start Value	0	0
End Value	5	2
Increment	0.1	0.5

@ 2023 All Rights Reserved.

When the DC sweep analysis is run, you can add a current marker at the drain pin of M1 and display the simulation results in PSpice A/D. The result is displayed in the following figure:

Figure 8-3 Device curve family.

Note: In a schematic capture application such as Capture, from the *PSpice* menu, point to *Markers*, you can choose *Current Into Pin* to add a current marker.

To add a load line for a resistor, you can add a trace that computes the load line from the sweep voltage. Assume that the X axis variable is the sweep voltage V_VD, which runs from 0 to 5 volts. The expression which will add a trace that is the load line for a 50 kohm resistor is:

```
(5V-V_VD)/50K
```

Note: V_VD is the hierarchical name for VD created by netlisting the schematic.

@ 2023 All Rights Reserved.

This can be useful for determining the bias point for each member of a curve family as shown in the figure.

Figure 8-4 Operating point determination for each member of the curve family.

Related Topics

- .DC (DC analysis)

@ 2023 All Rights Reserved.

Bias Point

The bias point is calculated for any analysis whether or not the Bias Point analysis is enabled in the *Simulation Settings* dialog box. However, additional information is reported when the Bias Point analysis is enabled.

When Bias Point analysis is not enabled, only analog node voltages and digital node states are reported to the output file.

When the Bias Point analysis is enabled, the following information is reported to the output file:

- a list of all analog node voltages
- a list of all digital node states
- the currents of all voltage sources and their total power
- a list of the small-signal parameters for all devices

@ 2023 All Rights Reserved.

If Bias Point is enabled, you can suppress the reporting of the bias point analog and digital node values. To do that, uncheck the box for bias point node voltages (NOBIAS) from *Output file* under the *Options* tab in the *Simulation Settings* dialog box.

@ 2023
All Rights Reserved.

Setting up Simulation Profile to Calculate Bias Point

1. Under *Analysis* type in the *Simulation Settings* dialog box, select *Bias Point*.
2. For the *General* settings option, enter the necessary parameter values and select the appropriate check boxes to complete the analysis specifications.
3. Click *OK* to save the simulation profile.
4. Start the simulation from your schematic capture application.

Related Topics

- .OP (bias point)

@ 2023 All Rights Reserved.

Small-Signal DC Transfer

The small-signal DC transfer analysis calculates the small-signal transfer function by transforming the circuit around the bias point and treating it as a linear circuit. The small-signal gain, input resistance, and output resistance are calculated and reported.

The digital devices themselves are not included in the small-signal analysis. A gate, for example, does not have a frequency response. Instead, all the digital devices hold the states that were calculated when solving for the bias point. However, for N and O devices in the analog or digital interface sub-circuits, the analog side has a well-defined linear equivalent.

Setting up a Small-Signal DC Transfer Analysis

To set up a small-signal DC transfer analysis, do the following:

1. Under *Analysis Type* in the *Simulation Settings* dialog box, select *Bias Point*.
2. Specify the name of the input source desired.

Note: The minimum requirements to run a small-signal DC transfer analysis is that the circuit should contain an input source, such as VSRC.

3. Click *OK* to save the simulation profile.
4. Start the simulation from your schematic capture application.

Calculating Small-Signal Gain, Input Resistance, and Output Resistance

To calculate a small-signal gain along with the input and output resistance, do the following:

1. In the *Bias Point* analysis type, select *Calculate small-signal DC gain (.TF)*.
2. Specify the value for either an output voltage or the current through a voltage source in the *To Output variable* text box.

   For example, entering V(a,b) as the output variable specifies that the output variable is the output voltage between two nets, a and b. Entering I(VDRIV) as the output variable specifies that the output variable is the current through a voltage source VDRIV.

3. Specify the input source name for *Calculate small-signal DC gain (.TF)* section.

   The gain from the input source to the output variable is calculated along with the input and output resistances.

   For example, if you enter `V(OUT2)` as the output variable and `V1` as the input source, the input resistance for V1 is calculated, the output resistance for V(OUT2) is calculated, and

@ 2023 All Rights Reserved.

the gain from V1 to V(OUT2) is calculated. All calculations are reported to the simulation output file.

Note: To run a small-signal DC transfer analysis, you need to place and connect one or more independent sources and set the DC voltage or current level for each source.

Related Topics

- .TF (transfer)

@ 2023
All Rights Reserved.

DC Sensitivity

DC sensitivity analysis calculates and reports the sensitivity of one node voltage to each device parameter for the following device types:

- resistors
- independent voltage and current sources
- voltage and current-controlled switches
- diodes
- bipolar transistors

The sensitivity is calculated by linearizing all devices around the bias point. Purely digital devices hold the states calculated when solving for the bias point

Note: There is no minimum requirements to run a DC sensitivity analysis.

Setting up a DC Sensitivity Analysis

To set up a DC sensitivity analysis, do the following:

1. Select *Analysis Type* in the *Simulation Settings* dialog box.
2. Choose *Perform Sensitivity analysis (.SENS)* under *General* settings.
3. Enter the required value(s) in the *Output variable(s)* box.
4. Click *OK* to save the simulation profile.

@ 2023 All Rights Reserved.

5. Start the simulation from your schematic capture application.

Related Topics

- .SENS (sensitivity analysis)

@ 2023 All Rights Reserved.

@ 2023 All Rights Reserved.

9

AC Analysis

The AC analyses evaluate circuit performance in response to a small-signal alternating current source. The following table summarizes what PSpice A/D calculates for each AC analysis type:

AC Analysis	PSpice A/D Computes
AC Sweep	Small-signal response of the circuit (linearized around the bias point) when sweeping one or more sources over a range of frequencies. Outputs include voltages and currents with magnitude and phase; you can use this information to obtain Bode plots.
Noise	For each frequency specified in the AC analysis: ■ Propagated noise contributions at an output net from every noise generator in the circuit ■ RMS sum of the noise contributions at the output ■ Equivalent input noise

Caution

To run a noise analysis, you must also run an AC sweep analysis.

Setting up an AC Analysis

To set up the AC analysis, do the following:

1. From the *PSpice* menu, choose *New Simulation Profile* or *Edit Simulation Profile*. (If this is a new simulation, enter the name of the profile and click *OK*.)

   The *Simulation Settings* dialog box appears.

2. Choose *AC Sweep* or *Noise* in the *Analysis Type* list box.

@ 2023 All Rights Reserved.

3. Under *Options*, select *General Settings* if it is not already enabled.

4. Set the number of sweep points as follows:

Sweep Frequency	Do this
linearly	Under *AC Sweep* Type, click *Linear*, and enter the total number of points in the sweep in the *Total Points* text box.
logarithmically by decades	Under *AC Sweep* Type, click *Logarithmic*, select `Decade` (default), and enter the total number of points per decade in the *Total Points* text box.
logarithmically by octaves	Under *AC Sweep* Type, click *Logarithmic*, select `Octave`, and enter the total number of points per octave in the *Total Points* text box.

5. In the *Start Frequency* and *End Frequency* text boxes, enter the starting and ending frequencies, respectively, for the sweep.

6. Click *OK* to save the simulation profile.

@ 2023
All Rights Reserved.

AC Sweep Setup in example.opj

Let us take an example of a circuit, `EXAMPLE.OPJ`, provided with your installed programs, and set up an AC analysis for this as shown in the following figure:

Figure 9-1 Circuit diagram for EXAMPLE.OPJ.

@ 2023 All Rights Reserved.

Note: The source, V1, is a VSIN source that is normally used for setting up sine wave signals for a transient analysis. It also has an AC property so that you can use it for an AC analysis.

Figure 9-2 AC analysis setup for EXAMPLE.OPJ.

Frequency is swept from 100 kHz to 10 GHz by decades, with 10 points per decade. The V1 independent voltage source is the only input to an amplifier, so it is the only AC stimulus to this circuit. Magnitude equals 1 V and relative phase is left at zero degrees (the default). All other voltage sources have zero AC value.

Related Topics

- Noise Analysis with PSpice

AC Sweep Analysis

AC sweep is a frequency response analysis. PSpice calculates the small-signal response of the circuit to a combination of inputs by transforming it around the bias point and treating it as a linear circuit. Here are a few things to note:

- Nonlinear devices, such as voltage- or current-controlled switches, are transformed to linear circuits about their bias point value before PSpice runs the linear (small-signal) analysis.

@ 2023 All Rights Reserved.

- Digital devices hold the states that PSpice calculated when solving for the bias point.
- Because AC sweep analysis is a linear analysis, it only considers the gain and phase response of the circuit; it does not limit voltages or currents.

The best way to use AC sweep analysis is to set the source magnitude to one. This way, the measured output equals the gain, relative to the input source, at that output.

Setting Up AC Sweep Analysis

To set up and run an AC sweep, do the following:

1. Place and connect a voltage or current source with an AC input signal.
2. From the *PSpice* menu, select *New Simulation Profile* or *Edit Simulation Profile*. If this is a new simulation, enter the name of the profile and click *OK*.

   The *Simulation Settings* dialog box appears.
3. Choose *AC Sweep* in the *Analysis* type list box.
4. Specify the required parameters for the AC sweep/Noise analysis you want to run.
5. Click *OK* to save the simulation profile.
6. From the *PSpice* menu, select *Run* to start the simulation.

Setting up an AC Stimulus

To run an AC sweep analysis, you need to place and connect one or more independent sources and set the AC magnitude and phase for each source.

Note that unlike DC sweep, the AC Sweep or Noise dialog box does not include an input source option. Instead, each independent source in your circuit contains its own AC specification for magnitude and phase.

To set up an AC stimulus, do the following:

1. Place and connect one of these symbols in your schematic:

For Voltage Input	
Input	**Analysis**
VAC	An AC sweep analysis only.

@ 2023 All Rights Reserved.

For Voltage Input	
Input	**Analysis**
VSRC	Multiple analysis types including AC sweep.

For Current Input	
Input	**Analysis**
IAC	An AC sweep analysis only.
ISRC	Multiple analysis types including AC sweep.

2. Double-click the symbol instance to display the parts spreadsheet.
3. Click in the cell under the appropriate property column to edit its value. Depending on the source symbol that you placed, define the AC specification as follows:

For VAC or IAC	
Property	**Value**
ACMAG	AC magnitude in volts (for VAC) or amps (for IAC); units are optional.
ACPHASE	Optional AC phase in degrees.

For VSRC or ISRC	
Property	**Value**
AC	*Magnitude_value* [*phase_value*] where *magnitude_value* is in volts or amps (units are optional) and the optional *phase_value* is in degrees.

@ 2023 All Rights Reserved.

Transforming a Device into a Linear Circuit

An AC Sweep analysis is a linear or small-signal analysis. This means that nonlinear devices must be linearized to run the analysis.

To transform a device (such as a transistor amplifier) into a linear circuit, you must do the following:

1. Compute the DC bias point for the circuit.
2. Compute the complex impedance and/or transconductance values for each device at this bias point.
3. Perform the linear circuit analysis at the frequencies of interest by using simplifying approximations.

   For example - Replace a bipolar transistor in common-emitter mode with a constant transconductance (collector current proportional to base-emitter voltage) and a number of constant impedances.

PSpice automates this process for you. PSpice computes the partial derivatives for nonlinear devices at the bias point and uses these to perform small-signal analysis.

Let us take an example of nonlinear behavioral modeling block.

Suppose you have an analog behavioral modeling block that multiplies V(1) by V(2). Multiplication is a nonlinear operation. To run an AC sweep analysis on this block, the block needs to be replaced with its linear equivalent. To determine the linear equivalent block, PSpice needs a known bias point.

@ 2023 All Rights Reserved.

Consider the circuit with a DC source as shown in the figure.

At the DC bias point, PSpice calculates the partial derivatives which determine the linear response of the multiplier as follows:

$$V(Out) = \frac{\partial V(Out)}{\partial V(In1)} + \frac{\partial V(Out)}{\partial V(In2)}$$

$$= 2.V(In1) + 2.0$$

For this circuit, this equation reduces to:

$$V(Out) = 2V(In1)$$

This means that the multiplier acts as an amplifier of the AC input with a gain that is set by the DC input.

@ 2023 All Rights Reserved.

Caution: Multiplying AC Sources

Suppose that you replace the 2 volt DC source in this example with an AC source with amplitude 1 and no DC value (DC=0). When PSpice computes the bias point, there are no DC sources in the circuit, so all nodes are at 0 volts at the bias point. The linear equivalent of the multiplier block is a block with gain 0, which means that there is no output voltage at the fundamental frequency. This is exactly how a double-balanced mixer behaves. In practice, this is a simple multiplier.

Note that a double-balanced mixer with inputs at the same frequency would produce outputs at DC at twice the input frequency, but these terms cannot be seen with a linear, small-signal analysis.

@ 2023 All Rights Reserved.

Noise Analysis with PSpice

When running a noise analysis, PSpice calculates and reports the following for each frequency specified for the AC Sweep/Noise analysis:

- Device noise, which is the noise contribution propagated to the specified output net from every resistor and semiconductor device in the circuit; for semiconductor devices, the device noise is also broken down into constituent noise contributions where applicable.

  Example: Diodes have separate noise contributions from thermal, shot, and flicker noise.

- Total output and equivalent input noise.

Value	Description
Output noise	RMS sum of all the device contributions propagated to a specified output net
Input noise	equivalent noise that would be needed at the input source to generate the calculated output noise in an ideal (noiseless) circuit

To calculate total noise at an output net, PSpice computes the RMS sum of the noise propagated to the net by all noise-generating devices in the circuit.

To calculate the equivalent input noise, PSpice then divides total output noise by the gain from the input source to the output net. This results in the amount of noise which, if injected at the input source into a noiseless circuit, would produce the total noise originally calculated for the output net.

@ 2023 All Rights Reserved.

Setting up and Running a Noise Analysis

The following procedure describes the minimum setup requirements for running a noise analysis.

To set up and run a Noise analysis, do the following:

1. Place and connect a voltage or current source with an AC input signal.
2. Set up the AC sweep simulation specifications.
3. Set up the noise simulation specifications and enable the analysis in the AC Sweep/Noise portion of the *Simulation Settings* dialog box.

   From the *PSpice* menu, choose *New Simulation Profile* or *Edit Simulation Profile*. (If this is a new simulation, enter the name of the profile and click *OK*).

   The *Simulation Settings* dialog box appears.
4. Click *OK* to save the simulation profile.
5. From the *PSpice* menu, choose Run to start the simulation.

6. Choose *AC Sweep/Noise* in the *Analysis Type* list box.
7. Under *Options*, select *General Settings* if it is not already enabled.

@ 2023
All Rights Reserved.

8. Specify the AC sweep analysis parameters as described in Setting up an AC analysis.
9. Select the *Noise Analysis* check box.
10. Enter the noise analysis parameters as follows:

Parameter	Description
Output Voltage	A voltage output variable of the form `V`(*node*, [*node*]) where you want the total output noise calculated.
I/V Source	The name of an independent current or voltage source where you want the equivalent input noise calculated. **Note:** If the source is in a lower level of a hierarchical schematic, separate the names of the hierarchical devices with periods (.). Example: U1.V2
Interval	An integer *n* designating that at every n^{th} frequency, you want to see a table printed in the PSpice output file (`.OUT`) showing the individual contributions of all of the circuit's noise generators to the total noise. **Note:** In the Probe window, you can view the device noise contributions at every frequency specified in the AC sweep. The Interval parameter has no effect on what PSpice writes to the Probe data file.

11. Click *OK* to save the simulation profile.

@ 2023 All Rights Reserved.

Analyzing Noise in Probe Window

You can use output variable formats to view traces for device noise contributions and total input or output noise at every frequency in the analysis.

To View	Output Variable	Equation
Flicker noise for a device	NFID(*device_name*) NFIB(*device_name*)	noise $\propto k_f \cdot \frac{I^{a_f}}{f^b}$
Shot noise for a device	NSID(*device_name*) NSIB(*device_name*) NSIC(*device_name*)	For diodes and BJTs: noise $\propto 2qI$ For GaAsFETs, JFETs, and MOSFETs: noise $\propto 4kT \cdot \frac{dI}{dV} \cdot \frac{2}{3}$
Thermal noise for the RB, RC, RD, RE, RG, or RS constituent of a device, respectively	NRB(*device_name*) NRC(*device_name*) NRD(*device_name*) NRE(*device_name*) NRG(*device_name*) NRS(*device_name*)	noise $\propto \frac{4kT}{R}$
Thermal noise generated by equivalent resistances in the output of a digital device	NRLO(*device_name*) NRHI(*device_name*)	noise $\propto \frac{4kT}{R}$
Total noise for a device	NTOT(*device_name*)	Sum of all contributors in *device_name*
Total output noise for the circuit	NTOT(ONOISE)	$\sum_{device} NTOT(device)$
RMS-summed output noise for the circuit	V(ONOISE)	RMS sum of all contributors $(\sqrt{NTOT(ONOISE)})$
Equivalent input noise for the circuit	V(INOISE)	$\frac{V(ONOISE)}{gain}$

@ 2023
All Rights Reserved.

Noise Units

Table 9-1

Noise Output Variable	Units
Device contribution of the form N*xxx*	$(volts)^2/(Hz)$
Total input or output noise of the form V(ONOISE) or V(INOISE)	$(volts)/(\sqrt{Hz})$

Let us take an example to run a noise analysis on the `EXAMPLE.OPJ` circuit in a schematic capture application, say, Capture.

In Capture, the example circuit is provided in the `<Install_directory>\tools\pspice\capture_samples\anasim\example` subdirectory.

To run a noise analysis on this example circuit, do the following:

1. Launch the schematic capture tool, Capture.
2. From the *PSpice* menu, choose *New Simulation Profile* or *Edit Simulation Profile*. (If this is a new simulation, enter the name of the profile and click *OK*.)

   The *Simulation Settings* dialog box appears.
3. Choose AC Sweep/Noise in the Analysis type list box.
4. Under *Options*, select *General Settings* if it is not already enabled.
5. Enable the *Noise Analysis* check box.
6. Enter the following parameters for the noise analysis:

Output Voltage	V(OUT2)
I/V Source	V1
Interval	30

These settings mean that PSpice will calculate noise contributions and total output noise at net OUT2 and equivalent input noise from V1.

The following figure shows Probe traces for Q1's constituent noise sources as well as total noise for the circuit after simulating:

@ 2023 All Rights Reserved.

Notice that the trace for RMSSUM (at the top of the plot), which is a macro for the trace expression

SQRT(NTOT(Q1) + NTOT(Q2) + NTOT(Q3) + ...),

exactly matches the total output noise, V(ONOISE), calculated by PSpice.

@ 2023
All Rights Reserved.

Figure 9-3 Device and total noise traces for EXAMPLE.OPJ.

Note: The source, V1, is a VSIN source that is normally used for setting up sine wave signals for a transient analysis. It also has an AC property so that you can use it for an AC analysis.

Frequency is swept from 100 kHz to 10 GHz by decades, with 10 points per decade. The V1 independent voltage source is the only input to an amplifier, so it is the only AC stimulus to this circuit. Magnitude equals 1 V and relative phase is left at zero degrees (the default). All other voltage sources have zero AC value.

Related Topics

- Setting up an AC Analysis
- Trace Expressions

@ 2023 All Rights Reserved.

10

Time Domain Analysis

The time-based analyses evaluate circuit performance in response to time-varying sources. The following table summarizes what PSpice A/D calculates for each time-based analysis type.

Time-Based Analysis	PSpice A/D Calculates
`Transient`	Voltages, currents, and digital states tracked over time For digital devices, you can set the propagation delays to minimum, typical, and maximum. If you have enabled digital worst-case timing analysis, PSpice A/D considers all possible combinations of propagation delays within the minimum and maximum range
`Fourier`	DC and Fourier components of the transient analysis results. **Note:** To run a Fourier analysis, you must also run a transient analysis.

To run a transient analysis, circuit design should contain one of the following to meet the minimum requirements:

- An independent source with a transient specification.
- An initial condition on a reactive element
- A controlled source that is a function of time

Setting Up a Transient Analysis

To set up a transient analysis, do the following:

1. From your schematic capture tool, select *PSpice*.
2. Choose *New Simulation Profile* or *Edit Simulation Profile*. (If this is a new simulation, enter the name of the profile)

@ 2023 All Rights Reserved.

3. click *OK*.

   The *Simulation Settings* dialog box is displayed.

4. Click the *Analysis* tab.
5. From the *Analysis* type list box, select *Time Domain* (Transient).
6. Specify the required parameters for the transient analysis you want to run.
7. Click *OK* to save the simulation profile.
8. From the *PSpice* menu, choose *Run* to start the simulation.

Check Point Options

You can save the state of a transient simulation at different moments as checkpoints. You can specify any check point as a restart point and rerun the simulation from that point. As a result, you can avoid running a long simulation from the start and run only the portion that is expected to have the outputs of interest.

For example, you simulate an SMPS design and determine a convergence error toward the end of the simulation. You can change design settings and rerun the portion of the simulation that throws the error instead of simulating the beginning.

To use Check Point, you need to perform the following steps:

@ 2023 All Rights Reserved.

1. Specify the CheckPoints.
2. Run the initial analysis.
3. Specify the restart point.
4. Restart the analysis.

You need to define the following for CheckPoints:

- Simulation time interval: The interval in simulation time between two CheckPoints. The default unit is seconds (s) but you can also specify the time in all standard scale modifiers, such as microseconds (ms) and nanoseconds (ns).
- Real time interval: The interval between two CheckPoints in real time. The default is minutes (min) but you can also specify intervals in hours (hrs).
- Time points: Specific times when CheckPoints are created.

  Note: Note that in case of digital circuits, CheckPoint will not be generated at a timepoint if there are no events. For example, if a digital circuit changes state from 0 to 1 at a specified timepoint, a checkpoint will be generated. But if there is no transition, checkpoint will not be generated at that timepoint.
- Directory location: The location where CheckPoint data is stored. The default location is the transient simulation profile directory.

You can define CheckPoints either from the *Simulation Settings* dialog box or by editing the circuit file (.cir).

Defining Checkpoints

You can specify check points using the *Simulation Settings* dialog box of your schematic capture application.

To define checkpoint, do the following:

1. Open the *Simulation Settings* dialog box.

@ 2023 All Rights Reserved.

2. In the *Analysis* tab, select *Save Check Point*.

3. In the *Check Point Location* box, specify the location where the Check Point data will be stored. You can specify more than one directory and change the order of directories in the *Directories* field using the buttons.
4. In the *Simulation Interval* box, specify the interval in seconds.
5. In the *Real Time Interval* box, specify the real time in minutes or hours.
6. In the *Time Points* box, specify the time points where checkpoints are to be created.

Important

The time points must be separated by either space or comma.

You can edit the circuit file to specify CheckPoints. The syntax for Check Point is:

```
.chkpt CheckPoint_name time_interval_type time_interval_value
[time_interval_type time_interval_value ][TSTEP 0/1]
```

Following can be the TSTEP values:

0	generate CheckPoints closest to the specified time points using default time step of PSpice engine
1	generate CheckPoints at the specified time points

@ 2023
All Rights Reserved.

The value for the parameter time_interval_type can be:

SINT	Specifies simulation time interval
RINT	Specifies real time interval
TP	Specifies time points

For example, the following specifies a Check Point to be saved as `D:\simdata\checkset1` with simulation time interval of 1ms, real time interval 10 minutes, and time points 2.5ms and 7.5ms:

```
.chkpt "D:/simdata/checkset1" SINT 1ms RINT 10min TP 2.5ms,7.5ms
```

Ensure that you have a transient analysis defined before running the simulation.

Restarting Simulation from a Saved Check Point

You might want to restart a simulation from a saved check point after changing the design. You can view the CheckPoint locations on the traces displayed in the Probe window. This will enable you to choose the CheckPoint to restart the simulation.

You can change the following in the schematic before restarting a simulation:

- Component values
- Parameter values
- Simulation options
- CheckPoint restart options
- Data save options

Caution

Before restarting simulations, note the following:

- Add or remove components before restarting simulation
- Change device name or order in the circuit file
- Change initial condition of device such as capacitor
- Change multi-analysis options, such as temperature, parameter, or MC sweep
- Topology changes for digital and mixed signal circuits:

@ 2023 All Rights Reserved.

- Changes in IO Model
- Changes in IO_LEVEL
- Changes in Timing models
- Changes in minimum time and maximum delay properties
- Change in the stimulus source

Important

Multi analysis options are not supported currently. If you select multi analysis and enable *Save Check Point* option, the simulation will run but checkpoint states will not be saved.

To restart simulation from a check point, do the following:

1. Choose *PSpice – Edit Simulation Profile*.
2. In the *Analysis* tab, select the *Restart Simulation* option.

3. Ensure that the correct check point location is selected in the *Check Point Location* section.
4. Specify the following options:
   - *Simulation Interval*
   - *Real Time Interval*

@ 2023
All Rights Reserved.

- *Time Points*

5. Click *OK*.
6. Run the simulation.

You can also edit the circuit file to specify the restart point. The syntax for the restart option is:

```
.restart CheckPoint_name state_number [0/1]
```

Where,

0	Original DAT file is overwritten
1	Original DAT file is saved as `checkpoint_<DAT_file_name>.dat`

You can also save checkpoint states either at *User specified time points* or *PSpice calculated time points*.

For example, to restart a simulation from the 20th saved state of the CheckPoint *D:/simdata/checkset1*:

```
.restart "D:/simdata/checkset1" state20
```

Stimulus Editor Utility

The Stimulus Editor is a utility that allows you to quickly set up and verify the input waveforms for a transient analysis. You can create and edit voltage sources, current sources, and digital stimuli for your circuit. Menu prompts guide you to provide the necessary parameters, such as the rise time, fall time, and period of an analog repeating pulse, or the complex timing relations with repeating segments of a digital stimulus. Graphical feedback allows you to quickly verify the waveform.

Note that if you are using a PSpice product that does not include the Stimulus Editor, you must use the characterized-by-property sources.

Stimulus Files Configuration

The Stimulus Editor produces a file containing the stimuli with their transient specification. These stimuli are defined as simulator device declarations using the V (voltage source), I (current source), and U STIM (digital stimulus generator) forms. Since the Stimulus Editor

@ 2023 All Rights Reserved.

produces these statements automatically, you will never have to be concerned with their syntax.

The Stimulus files list in the *Configuration Files* tab of the *Simulation Settings* dialog box allows you to view the list of stimulus files related to your current schematic.

You can also manually add, delete, or change the stimulus file configuration in this tab dialog box. The list box displays all of the currently configured stimulus files. One file is specified per line. Files can be configured as either global to the schematic capture environment, local to the current design, or only for the current profile. Global files are marked with the icon before the file name.

When starting the Stimulus Editor from the schematic capture application, stimulus files are automatically configured (added to the list) as local to the current design. Otherwise, new stimulus files can be added to the list by entering the file name in the Filename text box and then clicking the *Add to Profile* (profile specific configuration), *Add to Design* (local configuration) or *Add as Global* (global configuration) button.

@ 2023 All Rights Reserved.

Running the Stimulus Editor

The Stimulus Editor is fully integrated with the schematic capture tools such as Capture and can be run from either the schematic editor or symbol editor.

You can start the *Stimulus Editor* by doing the following:

1. Select one or more stimulus instances in the schematic.
2. From the *Edit* menu, choose *PSpice Stimulus*.

When you first start the Stimulus Editor, you may need to adjust the scale settings to fit the trace you are going to add. You can use Axis Settings on the Plot menu or the corresponding toolbar button to change the displayed data, the extent of the scrolling region, and the minimum resolution for each of the axes. Displayed Data Range parameters determine what portion of the stimulus data set will be presented on the screen. Extent of Scrolling Region parameters set the absolute limits on the viewable range. Minimum Resolution parameters determine the smallest usable increment (example: if it is set to 1 msec, then you cannot add a data point at 1.5 msec).

@ 2023 All Rights Reserved.

Defining Stimuli

To define a stimulus in Stimulus Editor, do the following:

1. Place stimulus part instances from the symbol set: VSTIM, ISTIM and DIGSTIMn (found in the SOURCSTM.OLB part library).
2. Click the source instance to select it.
3. From the *Edit* menu, choose *PSpice Stimulus* to start the Stimulus Editor.
4. Fill in the transient specification according to the dialogs and prompts.

   Piecewise linear and digital stimuli can be specified by direct manipulation of the input waveform display.
5. From the *File* menu, choose *Save* to save the edits.
6. Click *Yes* to update the schematic.

Let us take an example of a piecewise linear stimulus.

1. Open an existing schematic or start a new one.
2. From the *Place* menu, choose Part and browse the `SOURCSTM.OLB` part library file for VSTIM (and select it).
3. Place the part. It looks like a regular voltage source with an implementation property displayed.
4. Click the implementation label and type `Vfirst`. This names the stimulus that you are going to create.
5. If you are working in a new schematic, choose *Save* from the *File* menu. This schematic save step is necessary since the schematic name is used to create the default stimulus file name.
6. Click the VSTIM part to select it.

@ 2023 All Rights Reserved.

7. From the *Edit* menu, choose *PSpice Stimulus*. This starts the *Stimulus Editor* and displays the *New Stimulus* dialog box. You can see that the stimulus already has the name of Vfirst.

8. Select PWL in the dialog box and click *OK*. The cursor looks like a pencil. The message in the status bar at the bottom of the screen lets you know that you are in the process of adding new data points to the stimulus. The left end of the bottom status bar displays the current coordinates of the cursor.

9. Move the cursor to (200ns, 1) and click the left mouse button. This adds the point. Notice that there is automatically a point at (0,0). Ignore it for now and continue to add a couple more points to the right of the current one.

10. Click-right to stop adding points.

11. From the *File* menu, choose *Save*.

Note: If you make a mistake or want to make any changes, reshape the trace by dragging any of the handles to a new location. The dragged handle cannot pass any other defined data point.

➡ To delete a point, click its handle and press *Delete*.

To add additional points, either choose Add Point from the *Edit* menu, press *Alt+A*, or click the Add Point toolbar button .

@ 2023 All Rights Reserved.

At this point you can return to the schematic capture application, edit the current stimulus, or go on to create another.

Let us take an example of a Sine wave sweep.

This example creates a 10k sine wave with the amplitude parameterized so that it can be swept during a simulation.

1. Open an existing schematic or start a new one.
2. Place a *VSTIM* part on your schematic.
3. To name the stimulus, double-click the implementation property and type `Vsin`.
4. Click the *VSTIM* part to select it.
5. From the *PSpice* menu, choose *Edit Stimulus* to start the *Stimulus Editor*.
6. Define the stimulus parameter for amplitude:
   a. From the *New Stimulus* dialog box, choose *Cancel*.
   b. From the *Tools* menu, choose *Parameters*.
   c. Enter `AMP=1` in the Definition text box, and click *OK*.
   d. From the *Stimulus* menu, choose New or click the *New Stimulus* button in the toolbar.
   e. Give the stimulus the name of Vsin.
   f. Select *SIN* as the type of stimulus to be created, and click *OK*.
7. Define the other stimulus properties:

@ 2023 All Rights Reserved.

   a. Enter `0` for Offset Value.

   b. Enter `{AMP}` for Amplitude. The curly braces are required. They indicate that the expression needs to be evaluated at simulation time.

   c. Enter `10k` for Frequency and click OK.

   d. From the File menu, choose Save.

   e. Click Yes to update the schematic.

8. In the schematic capture tool, place and define the PARAM symbol:

   a. From the Place menu, choose Part.

   b. Either browse `SPECIAL.OLB` for the PARAM part or type in the name.

   c. Place the part on your schematic and double-click it.

   d. Click New to add a new user property.

   e. Set the value property name to AMP (no curly braces).

   f. Set the value of the AMP property to 1.

9. Set up the parametric sweep and other analyses:

   a. From the *PSpice* menu, choose *Edit Simulation Profile*, and select the *Parametric Sweep* option.

   b. Select *Global Parameter* in the Swept Var. Type frame.

   c. Select *Linear* in the Sweep type frame.

10. Enter `AMP` in the Parameter name text box.

11. Specify values for the *Start Value*, *End Value*, and *Increment* text boxes.

Creating New Stimulus Symbols

1. Use the Capture part editor to edit or create a part with the following properties:

Property	Value
Implementation Type	PSpice Stimulus
Implementation	Name of the stimulus model

@ 2023 All Rights Reserved.

Property	Value
STIMTYPE	Type of stimulus; valid values are ANALOG or DIGITAL; if this property is nonexistent, the stimulus is assumed to be ANALOG

@ 2023 All Rights Reserved.

Editing a Stimulus

To edit an existing stimulus, do the following:

1. Start the *Stimulus Editor* and select *File*, Open to open the required Stimulus library.
2. Double-click the trace name (at the bottom of the X axis for analog and to the left of the Y axis for digital traces.) This opens the *Stimulus Attributes* dialog box where you can modify the attributes of the stimulus directly and immediately see the effect of the changes.

To edit a PWL stimulus, do the following:

PWL stimuli are a little different since they are a series of time/value pairs.

1. Double-click the trace name. This displays the handles for each defined data point.
2. Click any handle to select it. To reshape the trace, drag it to a new location. To delete the data point, press *Delete*.
3. To add additional data points, either select Add from the *Edit* menu or click the *Add Point* button.
4. Right-click to end adding new points.

To select a time and value scale factor for PWL stimuli, do the following:

1. Select the PWL trace by clicking on its name.
2. Select *Attributes* from the *Edit* menu or click the corresponding toolbar button .

Note that the procedure provides a fast way to scale a PWL stimulus.

Deleting and Removing Traces

To delete a trace from the displayed screen, select the trace name by clicking on its name, then press *Delete*. This will only erase the display of the trace, not delete it from your file. The trace is still available by selecting Get from the *Stimulus* menu.

➡ To remove a trace from a file, select *Remove* from the *Stimulus* menu.

Note that once a trace is removed, it is no longer retrievable. Remove traces with caution.

@ 2023 All Rights Reserved.

Configuring Stimulus Manually

Stimuli can be characterized by manually starting the Stimulus Editor and saving their specifications to a file. These stimulus specifications can then be associated to stimulus instances in your schematic or to stimulus symbols in the symbol library.

To manually configure a stimulus, do the following:

1. Start the *Stimulus Editor* by choosing the *Stimulus Editor* icon from the Windows Start menu, installed OrCAD X program group, PSpice Accessories option.
2. Open a stimulus file by choosing *Open* from the *File* menu. If the file is not found in your current library search path, you are prompted for a new file name.
3. Create one or more stimuli to be used in your schematic. For each stimulus:
   - **a.** Name it whatever you want. This name will be used to associate the stimulus specification to the stimulus instance in your schematic, or to the symbol in the symbol library.
   - **b.** Provide the transient specification.
   - **c.** From the *File* menu, choose *Save*.
4. In the schematic page editor, configure the Stimulus Editor's output file into your schematic:
   - **a.** From the *PSpice* menu, choose *Edit Simulation Profile* to display the *Simulation Settings* dialog box.
   - **b.** In the *Simulation Settings* dialog box, select the *Configuration Files* tab.
   - **c.** Click *Include* in the *Category* field to display the Include files list.
   - **d.** Enter the file name specified in step 2.
   - **e.** If the stimulus specifications are for only for the current profile, click the Add to Profile button. For local use in the current design, click the Add to Design button. For global use by any design, use Add as Global instead.
   - **f.** Click *OK*.
5. Modify either the stimulus instances in the schematic or symbols in the symbol library to reference the new stimulus specification.
6. Associate the transient stimulus specification to a stimulus instance:

@ 2023 All Rights Reserved.

a. Place a stimulus part in your schematic from the part set: VSTIM, ISTIM, and DIGSTIMn.

b. Click the VSTIM, ISTIM, or DIGSTIMn instance.

c. From the *Edit* menu, choose *Properties*.

d. Click the *Implementation* cell, type in the name of the stimulus, and click *Apply*.

e. Complete specification of any VSTIM or ISTIM instances by selecting Properties from the *Edit* menu and editing their DC and AC attributes.

Click the DC cell and type its value.

Click the AC cell, type its value, and then click *Apply*.

f. Close the property editor spreadsheet.

7. To change stimulus references globally for a part:

a. Select the part you want to edit.

b. From the *Edit* menu, choose *Part* to start the part editor.

c. Create or change the part definition, making sure to define the following property:

Implementation	stimulus name as defined in the Stimulus Editor

@ 2023 All Rights Reserved.

Transient (time) Response

The transient response analysis causes the response of the circuit to be calculated from TIME = 0 to a specified time. A transient analysis specification is shown for the circuit `EXAMPLE.OPJ`.

The analysis is to span the time interval from 0 to 1000 nanoseconds and values should be reported to the simulation output file every 20 nanoseconds.

Figure 10-1 Transient analysis setup for EXAMPLE.OPJ

During a transient analysis, any or all of the independent sources may have time-varying values. In `EXAMPLE.OPJ`, the only source which has a time-varying value is V1 (VSIN part) with attributes:

```
VOFF = 0v
VAMPL = 0.1v
FREQ = 5Meg
```

V1 varies as a 5 MHz sine wave with an offset voltage of 0 volts and a peak amplitude of 0.1 volts.

@ 2023 All Rights Reserved.

In general, more than one source has time-varying values; for instance, two or more clocks in a digital circuit.

Figure 10-2 Example schematic EXAMPLE.OPJ.

Note: The example circuit EXAMPLE.OPJ is provided with PSpice.

The transient analysis does its own calculation of a bias point to start with, using the same technique as described for DC sweep. This is necessary because the initial values of the sources can be different from their DC values. In the simulation output file `EXAMPLE.OUT`, the bias point report for the transient bias point is labeled INITIAL TRANSIENT SOLUTION.

To report the bias point information for nonlinear controlled sources and semiconductors, choose the OP option from the *Output File Options* dialog box. This bias point information is reported in the output file under the OPERATING POINT INFORMATION section.

@ 2023 All Rights Reserved.

Internal Time Steps

During analog analysis, PSpice maintains an internal time step which is continuously adjusted to maintain accuracy while not performing unnecessary steps. During periods of inactivity, the internal time step is increased. During active regions, it is decreased. The maximum internal step size can be controlled by specifying it in the Maximum Time Step text box in the Transient dialog box. PSpice will never exceed either the step ceiling value or two percent of the total transient run time, whichever is less.

The internal time steps used may not correspond to the time steps at which information is reported. The values at the print time steps are obtained by second-order polynomial interpolation from values at the internal steps.

When simulating mixed analog/digital circuits, there are actually two time steps: one analog and one digital. This is necessary for efficiency. Since the analog and digital circuitry usually have very different time constants, any attempt to lock them together would greatly slow down the simulation. The time step shown on the PSpice display during a transient analysis is that of the analog section.

Switching Circuits

Running transient analysis on switching circuits can lead to long run times. PSpice must keep the internal time step short compared to the switching period, but the circuit's response extends over many switching cycles.

One method of avoiding this problem is to transform the switching circuit into an equivalent circuit without switching. The equivalent circuit represents a sort of quasi steady-state of the actual circuit and can correctly model the actual circuit's response as long as the inputs do not change too fast.

@ 2023 All Rights Reserved.

Hysteresis Curves

Transient analysis can be used to look at a circuit's hysteresis. Consider, for instance, the circuit shown in the following figure:

@ 2023
All Rights Reserved.

```
* Capture Netlist

R_RIN    1 2 50
R_RC1    0 3 50
R_R1     3 5 185
R_R2     5 8 760
R_RC2    0 6 100
R_RE     4 8 260
R_RTH2   7 0 85
C_CLOAD  0 7 5PF
V_VEE    8 0 dc -5
V_VIN    1 0
+PWL 0 -8 1MS -1.0V 2MS -1.8V
R_RTH1   8 7 125
Q_Q1     3 2 4 QSTD
Q_Q2     6 5 4 QSTD
Q_Q3     0 6 7 QSTD
Q_Q4     0 6 7 QSTD
```

Figure 10-4 Netlist for Schmitt trigger circuit.

The QSTD model is defined as:

```
.MODEL QSTD NPN( is=1e-16 bf=50 br=0.1 rb=50 rc=10 tf=.12ns tr=5ns
+ cje=.4pF pe=.8 me=.4 cjc=.5pF pc=.8 mc=.333 ccs=1pF va=50)
```

Instead of using the DC sweep to look at the hysteresis, use the transient analysis, (Print Step = .01ms and Final Time = 2ms) sweeping VIN from -1.8 volts to -1.0 volts and back down to -1.8 volts, very slowly. This has the following advantages:

- it avoids convergence problems
- it covers both the upward and downward transitions in one analysis

After the simulation, in the Probe window in PSpice, the X axis variable is initially set to Time. By selecting X Axis Settings from the Plot menu and clicking on the Axis Variable button, you can set the X axis variable to V(1). Then use Add on the Trace menu to display V(7), and change the X axis to a user-defined data range from -1.8V to -1.0V (Axis Settings on the Plot

@ 2023 All Rights Reserved.

menu). This plots the output of the Schmitt trigger against its input, which is the desired outcome. The result looks similar to in the following figure:

Figure 10-5 Hysteresis curve example: Schmitt trigger.

@ 2023 All Rights Reserved.

Fourier Analysis

Fourier analysis is enabled through the *Output File Options* dialog box under the *Time Domain (Transient)* Analysis type. Fourier analysis calculates the DC and Fourier components of the result of a transient analysis. By default, the first through ninth components are computed; however, more can be specified.

Note: You must do a transient analysis in order to do a Fourier analysis. The sampling interval used during the Fourier transform is equal to the print step specified for the transient analysis.

When selecting Fourier to run a harmonic decomposition analysis on a transient waveform, only a portion of the waveform is used. Using the *Probe* window in PSpice, a Fast Fourier Transform (FFT) of the complete waveform can be calculated and its spectrum displayed.

In the example shown in Figure 10-1 on page 358, the voltage waveform at node OUT2 from the transient analysis is used and the fundamental frequency is one megahertz for the harmonic decomposition. The period of fundamental frequency is one microsecond (inverse of the fundamental frequency). Only the last one microsecond of the transient analysis is used, and that portion is assumed to repeat indefinitely. Since V1's sine wave does indeed repeat every one microsecond, this is sufficient. In general, however, you must make sure that the fundamental Fourier period fits the waveform in the transient analysis.

@ 2023 All Rights Reserved.

11

Parametric Analysis

Parametric analysis performs multiple iterations of a specified standard analysis while varying a global parameter, model parameter, component value, or operational temperature. The effect is the same as running the circuit several times, once for each value of the swept variable.

You can use performance analysis to derive values from a series of simulator runs and plot these values versus a parameter that varies between the simulator runs.

To run a parametric analysis, a circuit design should set up the following to meet the minimum requirements:

- Set up the circuit according to the swept variable type.
- Set up a DC sweep, AC sweep, or transient analysis.

Swept Variable Type	Requirement
voltage source	voltage source with a DC specification (VDC, for example)
temperature	none
current source	current source with a DC specification (IDC, for example)
model parameter	PSpice A/D model
global parameter	global parameter defined with a parameter block (PARAM)

To run a parametric sweep, do the following:

1. In the *Simulation Settings* dialog box, from the *Analysis Type* list box, select `Time Domain (Transient)`.
2. Under *Options*, select *Parametric Sweep* if it is not already enabled.

@ 2023 All Rights Reserved.

3. Specify the required parameters for the sweep.

4. Click *OK* to save the simulation profile.
5. From the *PSpice* menu, choose *Run* to start the simulation.

Note: Do not specify a DC sweep and a parametric analysis for the same variable.

Performance Analysis

Let us take an example of RLC filter to perform a parametric sweep and analyze the results with performance analysis.

For this example, the derived values are the overshoot and the rise time versus the damping resistance of the filter.

The schematic representation for the RLC filter (`RLCFILT.OPJ`) is shown in the figure.

This series of PSpice A/D runs varies the value of resistor R1 from 0.5 to 1.5 ohms in 0.1 ohm steps. Since the time-constant of the circuit is about one second, perform a transient analysis of approximately 20 seconds.

You can create the circuit in your schematic capture application by placing a piecewise linear independent current source (IPWL from `SOURCE.OLB`) and set the current source properties as follows:

```
AC = 1a
T1 = 0s
I1 = 0a
```


@ 2023 All Rights Reserved.

Figure 11-1 Passive filter schematic.

```
T2 = 10ms
I2 = 0a
T3 = 10.1ms
I3 = 1a
```

Place an instance of a resistor and set its VALUE property to the expression, {R}.

To define R as a global parameter, place a PARAM pseudo-component and use the *Property Editor* to create a new property R and set its value to 0.5. Place an inductor and set its value to 1H, place a capacitor and set its value to 1, and place an analog ground symbol (0 from the `SOURCE.OLB` library). Wire the schematic symbols together as shown in the figure.

Run PSpice A/D with the following analyses enabled:

transient	print step:	100ms
	final time:	20s
parametric	swept var. type:	global parameter
	sweep type:	linear
	name:	R
	start value:	0.5
	end value:	1.5
	increment:	0.1

After setting up the analyses, start the simulation by choosing *Run* from the *PSpice* menu.

@ 2023 All Rights Reserved.

Using Performance Analysis to Plot Overshoot and Rise Time

After performing the simulation that creates the data file `RLCFILT.DAT`, you can calculate the specified performance analysis measurements.

When the simulation is finished, a list appears containing all of the sections (runs) in the data file produced by PSpice A/D. To use the data from every run, select All and click *OK* in the *Available Selections* dialog box. In the case of the following figure, the trace I(L1) from the ninth section was added by specifying the following in the *Add Traces* dialog box:

```
-I(L1)@9
```

Note: To display the *Add Traces* dialog box, from the *Trace* menu, choose *Add Trace* or click the *Add Trace* toolbar button .

Figure 11-2 Current of L1 when R1 is 1.5 ohms.

To run performance analysis, do the following:

@ 2023 All Rights Reserved.

1. From the *Trace* menu, choose *Performance Analysis*.
2. Click *OK*.

   PSpice A/D resets the X-axis variable for the graph to be the parameter that changed between PSpice A/D runs. In the example, this is the R parameter.

To see the rise time for the current through the inductor L1, click the *Add Trace* toolbar button and then enter:

```
genrise(-I(L1) )
```

The genrise and overshoot measurements are contained in the PSPICE.PRB file in the `<target_installation_directory>\PSpice\Common` directory. For legacy users, there are now two files in the PSpice\Common directory that uses measurement names:

- PSPICE_OLB.PRB
- PSPICE.PRB

The figure shows how the rise time decreases as the damping resistance increases for the filter.

Another Y axis can be added to the plot for the overshoot of the current through L1 by selecting Add Y Axis from the Plot menu. The Y axis is immediately added. Now click the *Add Trace* toolbar button and enter:

```
overshoot(-I(L1) )
```


@ 2023 All Rights Reserved.

The following figure shows how the overshoot increases with increasing resistance.

Figure 11-3 Rise time and overshoot vs. damping resistance.

Troubleshooting Tip

More than one PSpice A/D run or data section is required for performance analysis. Because one data value is derived for each waveform in a related set of waveforms, at least two data points are required to produce a trace.

Use *Evaluate Measurement* (from the *Trace* menu) to evaluate a measurements on a single waveform and produce a single data point result.

@ 2023 All Rights Reserved.

Frequency Response vs. Arbitrary Parameter

You can view a plot of the linear response of a circuit at a specific frequency as one of the circuit parameters varies (such as the output of a band pass filter at its center frequency vs. an inductor value).

In this example, the value of a nonlinear capacitance is measured using a 10 kHz AC signal and plotted versus its bias voltage. The capacitance is in parallel with a resistor, so a trace expression is used to calculate the capacitance from the complex admittance of the R-C pair.

Note: This technique for measuring branch capacitances works well in both simple and complex circuits.

Create the circuit in the schematic capture application as shown in the following figure.

Figure 11-4 RLC filter example circuit.

To create the capacitor model in the schematic editor:

1. Place a CBREAK part.

@ 2023 All Rights Reserved.

2. Select it so that it is highlighted.
3. From the *Edit* menu, choose *PSpice* Model.
4. In the Model Text frame, enter the following:

   ```
   .model Cnln CAP(C=1 VC1=-0.01 VC2=0.05)
   ```

5. From the *File* menu, choose *Save*.

Set up the circuit for a parametric AC analysis (sweep Vbias), and run PSpice A/D. Include only the frequency of interest in the AC sweep.

To display the results, do the following:

Use PSpice A/D to display the capacitance calculated at the frequency of interest versus the stepped parameter.

1. Simulate the circuit.
2. Load all AC analysis sections.
3. From the *Trace* menu, choose *Add Trace* or click the *Add Trace* toolbar button .
4. Add the following trace expression:

   ```
   IMG(-I(Vin)/V(1,0))/(2*3.1416*Frequency)
   ```

   Or add the expression:

   ```
   CvF(-I(Vin)/V(1,0))
   ```

Where CvF is a macro which measures the effective capacitance in a complex conductance. Macros are defined using the Macros command on the Trace menu. The CvF macro should be defined as:

```
CvF(G)= IMG(G)/(2*3.1416*Frequency)
```

Note: -I(Vin)/V(1) is the complex admittance of the R-C branch; the minus sign is required for correct polarity.

Using Performance Analysis to Plot Capacitance vs. Bias Voltage

To plot capacitance vs. bias voltage, do the following:

1. From the *Trace* menu, choose *Performance Analysis*.
2. Click *Wizard*.
3. Click *Next*>.
4. Click *YatX* in the *Choose a Measurement* list, and click *Next*>.

@ 2023 All Rights Reserved.

5. In the Name of Trace text box, type the following:

   `CvF(-I(Vin)/V(1))`

6. In the X value to get Y value at text box, type `10K`.

7. Click *Next>*.

   The wizard displays the gain trace for the first run to text the measurement (YatX).

8. Click *Finish*.

   The resultant plot is shown in the following figure:

Figure 11-5 Plot of capacitance versus bias voltage.

@ 2023
All Rights Reserved.

@ 2023 All Rights Reserved.

12

Temperature Analysis

For a temperature analysis, PSpice A/D reruns standard analyses set in the *Simulation Settings* dialog box at different temperatures.

Note: Running multiple analyses for different temperatures can also be achieved using parametric analysis. With parametric analysis, the temperatures can be specified either by list, or by range and increments within the range.

You can specify zero or more temperatures. If no temperature is specified, the circuit is run at 27°C. If more than one temperature is listed, the simulation runs once for each temperature in the list.

Setting the temperature to a value other than the default results in recalculating the values of temperature-dependent devices. In `EXAMPLE.OPJ`, the temperature for all of the analyses is set to 35°C. The values for resistors RC1 and RC2 are recomputed based upon the CRES model which has parameters TC1 and TC2 reflecting linear and quadratic temperature dependencies.

@ 2023 All Rights Reserved.

Likewise, the Q3 and Q4 device values are recomputed using the Q2N2222 model which also has temperature-dependent parameters. In the simulation output file, these recomputed device values are reported in the section labeled TEMPERATURE ADJUSTED VALUES.

Figure 12-1 Example schematic EXAMPLE.OPJ.

Note: The example circuit EXAMPLE.OPJ is provided with the installed programs.

Setting up a Temperature Analysis

There is no minimum circuit design requirements to run temperature analysis.

The minimum program setup requirements are as follows:

1. In the *Simulation Settings* dialog box, from the *Analysis Type* list box, select `Time Domain (Transient)`.
2. Under *Options*, select *Temperature Sweep* if it is not already enabled.

   Note: Temperature Sweep is not supported in the Advanced Analysis flow.

@ 2023 All Rights Reserved.

3. Specify the required parameters for the sweep.

4. Click *OK* to save the simulation profile.
5. From the *PSpice* menu, choose *Run* to start the simulation.

@ 2023
All Rights Reserved.

@ 2023 All Rights Reserved.

13

Statistical Analyses

PSpice supports the following type of statistical analyses:

- Monte Carlo Analysis
- Sensitivity Analysis

The Monte Carlo and worst-case are statistical analyses that vary the lot or device tolerances of devices between multiple runs of an analysis (DC, AC, or transient). Before running the analysis, you must set up the model and/or lot tolerances of the model parameter to be investigated.

A Monte Carlo analysis performs a Monte Carlo (statistical) analysis of the circuit. A worst-case analysis performs a sensitivity and worst-case analysis of the circuit.

Sensitivity (worst-case) analyses are different from Monte Carlo analyses in that they compute the parameters using the sensitivity data rather than using random numbers.

You can run either a Monte Carlo or a worst-case analysis, but you *cannot* run both at the same time. Multiple runs of the selected analysis are done while parameters are varied. You can select only one analysis type (AC, DC, or transient) per run. The selected analysis is repeated in subsequent passes of the analysis.

As the number of Monte Carlo or worst-case runs increases, simulation takes longer and the data file gets larger. Large data files may be slow to open and slow to draw traces.

One way to work around this is to set up an overnight batch job to run the simulation and execute commands. You can even set up the batch job to produce a series of plots on paper to be ready for you in the morning.

Related Topics

- Batch Simulations

@ 2023 All Rights Reserved.

Monte Carlo History Support

Using the history support feature of Monte Carlo, you can store the model parameter values used for each Monte Carlo run, in a separate file, and later reuse these values.

For every Monte Carlo run, the model parameter values are generated randomly with in the tolerance range specified by you. With the history support feature, you can save these randomly generated values and reuse exactly the same values in the next analysis.

The Monte Carlo history support feature allows you to compare the results of two Monte Carlo analyses by manually changing only one or more parameter values. For comparison between simulations, the random numbers have to remain the same for the toleranced model parameters and this can be achieved using the Monte Carlo history support feature.

Monte Carlo History Support enables the following functionalities:

- Saving Parameter Values in a File
- Reusing Parameter Values

Saving Parameter Values in a File

To enable saving of the randomly generated model parameter values, do the following:

1. Select *PSpice – Edit Simulation Profile*.

   The *Simulation Settings* dialog box appears.

2. In the *Analysis* tab, select the *Analysis Type* as `Time Domain(Transient), DC Sweep or AC Sweep/Noise,` or `Bias Point`.

3. In the *Options* list, select *Monte Carlo/Worst Case*.

@ 2023 All Rights Reserved.

4. Select the *MC Load/Save* button.

5. In the Load/Save Monte Carlo Parameter File dialog box, select the *Save Parameter values in filename:* check box.

6. In the text box that is enabled, specify the name and the location of the file in which the parameter data is to be saved.

7. Click *OK* to save your settings.

If you now run the Monte Carlo analysis, the model parameter values used for various simulation runs will be stored in the `.mcp` file specified by you.

@ 2023
All Rights Reserved.

Reusing Parameter Values

To reuse model parameter values from a previous Monte Carlo analysis, do the following:

1. Select *PSpice – Edit Simulation Profile*.

   The Simulation Settings dialog box appears.

2. In the *Analysis* tab, select the *Analysis Type* as `Time Domain(Transient), DC Sweep` or `AC Sweep/Noise,` or `Bias Point`.

3. In the *Options* list, select *Monte Carlo/Worst Case*

4. Select the *MC Load/Save* button.

5. In the Load/Save Monte Carlo Parameter File dialog box, select the *Load Parameter values in filename:* check box.

6. In the text box that is enabled, specify the name and the location of the `.mcp` file from which the parameter data is to be read.

7. Click *OK* to save your settings.

Now when you run the Monte Carlo analysis, the simulator will reuse all the model parameter values stored in the `.mcp` files. Any new or additional parameter values will be varied separately with in the tolerance range.

While reusing values from a `.mcp` file, you must take care of the following points:

- When you use the Monte Carlo history support feature, the values stored in a `.mcp` file for model parameters are different from the values stored for components, such as resistor, capacitor, and inductor. In case of model parameters, actual parameter values used for simulation are stored. Whereas in case of the components, instead of the actual value, the multiplication factor used for generating the random values are stored.

Model Parameter with Base Value 100 and Tolerance 10%	Resistor with Base Value 100 and Tolerance 10%
`1.00000e+002`	1
`1.02747e+002`	1.002
`1.07986e+002`	.986
`1.04803e+002`	1.003
`9.17754e+001`	.9912

@ 2023 All Rights Reserved.

The difference in the method used for storing values from previous Monte Carlo runs, is highlighted when you change the original value of a parameter and also choose to reuse the values from a previous Monte Carlo run.

For example, if you change the original base value of a model parameter from 100 to 50 and reuse the values from an `.mcp` file, the simulation results will be based on the original parameter value, which is 100. Whereas, in case you change the value of a resistor from 100Ω to 10Ω, and reuse the values from a `.mcp` file, simulation results will be based on the changed value of 10Ω.

- If the number of Monte Carlo runs is greater than the number of values in the `.mcp` file, PSpice simulator with first reuse all the values from the .mcp file and then for rest of the runs, the random values will be generated using the base value and the tolerance defined for the parameter in the schematic.

  For example, consider that for a particular model parameter, the `.mcp` file from which the parameter data is to be read has 10 entries, but for the current simulation, 20 Monte Carlo runs are required. In such cases, for the first 10 Monte Carlo runs the values will be read from the `.mcp` file. For the last 10 runs, the simulator will calculate the values using the base value of the model parameter and the tolerance specified on it.

The Monte Carlo Parameter (.mcp) file

When you use the Monte Carlo history support feature, you either generate a Monte Carlo Parameter (`.mcp`) file or use parameters from a `.mcp` file. A `.mcp` file is a text file that stores the parameter information generated during Monte Carlo analysis. The data is stored in a tabular format with the data values separated by *white space*s, *blank*s, or *tabs*. Each columns in a `.mcp` file indicates a parameter whose values was varied during the Monte Carlo analysis.

@ 2023 All Rights Reserved.

The format for a `.mcp` file:

In a `.mcp` file, the name of the variable parameter is defined using the following format:

```
<instance_name>::ParameterName.
```

where...	indicates...
`instance_name`	reference designator or the name that appears in the PSpice netlist
`ParameterName`	indicates whether the component is a resistor, capacitor, transistor, and so on.

For example, if the name of the variable parameter is `R_U2_R2::R`, `R_U2_R2` is the reference designator and `R` indicates the parameter name, which in this case is a resistor.

A sample `.mcp` file that has three parameters values varied as shown.

```
C_U2_C2::C          R_U1_R1::R          R_U1_R2::R
1.00000e+000        1.00000e+000        1.00000e+000
1.03859e+000        9.94929e-001        9.94938e-001
1.01677e+000        9.96618e-001        1.00573e+000
9.94697e-001        1.00455e+000        9.99755e-001
9.93533e-001        9.98663e-001        1.00048e+000
```


@ 2023 All Rights Reserved.

Waveform Reports

For Monte Carlo analyses, there are five variations of the output that you can specify in the Save data from text box on the *Monte Carlo* dialog box. Following are the options:

<none>	No output is generated
All	Forces all output to be generated (including nominal run)
First*	Generates output only during the first *n* runs
Every*	Generates output for every *n*th run
Runs(list)*	Does specified analysis and generates outputs only for the listed runs (up to 25 values can be specified in the list)

The * indicates that you can set the number of runs in the runs text box.

Values for the output variables specified in the selected analyses are saved to the simulation output file and data file.

Note: In excess of about 10 runs, the waveform display can look more like a band than a set of individual waveforms. This can be useful for seeing the typical spread for a particular output variable. As the number of runs increases, the spread more closely approximates the actual worst-case limits for the circuit.

Note: Even a modest number of runs can produce large output files.

Collating Functions

You can further compress the results of Monte Carlo and worst-case analyses. If you use the collating function, a single number represents each run. You can click the *More Settings Options* button and select a function from the *Find* list. A table of deviations per run is reported in the simulation output file.

Collating functions are listed in Table 13-1.

Table 13-1 Collating functions used in statistical analyses

Function	Description
YMAX	Find the greatest difference in each waveform from the nominal

@ 2023 All Rights Reserved.

Function	Description
MAX	Find the maximum value of each waveform
MIN	Find the minimum value of each waveform
RISE_EDGE	Find the first occurrence of the waveform crossing above a specified threshold value
FALL_EDGE	Find the first occurrence of the waveform crossing below a specified threshold value

Output Control for Statistical Analyses

Monte Carlo and Sensitivity(worst-case) analyses generate the following types of reports:

- Model parameter values used for each run (that is, the values with tolerances applied)
- Waveforms from each run, as a function of specifying data collection, or by specifying output variables in the analysis set up
- Summary of all the runs using a collating function

Output is saved to the data file for use by the waveform analyzer. For Monte Carlo analyses, you can use the performance analysis feature to produce histograms of derived data.

Generating Model Parameter Values Report

To produce a list of the model parameters actually used for each run,

1. In the *Simulation Settings* dialog box, click the *Analysis* tab.
2. From the *Analysis Type* list, select an analysis type.
3. Under *Options*, select *Monte Carlo/Worst Case*.
4. Click the *More Settings* button.
5. Select *List model parameter values in the output file of each run*.
6. Click *OK*.

This list is written to the simulation output file at the beginning of the run and contains the parameters for each device, as opposed to the parameters for each .MODEL statement. This is because devices can have different parameter values when using a model statement containing a DEV tolerance.

@ 2023 All Rights Reserved.

Note that for midsize and large circuits, the List option can produce a large output file.

@ 2023 All Rights Reserved.

Temperature Considerations in Statistical Analyses

The statistical analyses perform multiple runs, as does the temperature analysis. Conceptually, the Monte Carlo and worst-case loops are inside the temperature loop.

However, since both temperature and tolerances affect the model parameters, OrCAD X recommends not using temperature analysis when using Monte Carlo or worst-case analysis.

Also, you cannot sweep the temperature in a DC sweep analysis or put tolerances on temperature coefficients while performing one of these statistical analyses. In `EXAMPLE.OPJ`, the temperature value is fixed at 35◦C.

Figure 13-1 Example schematic EXAMPLE.OPJ.

Note: The example schematic `EXAMPLE.OPJ` is provided with the installed programs.

@ 2023
All Rights Reserved.

Monte Carlo Analysis

The Monte Carlo analysis calculates the circuit response to changes in part values by randomly varying all of the model parameters for which a tolerance is specified. This provides statistical data on the impact of a device parameter's variance.

Monte Carlo analysis is frequently used to predict yields on production runs of a circuit. With Monte Carlo analysis, model parameters are given tolerances, and multiple analyses (DC, AC, or transient) are run using these tolerances.

For `EXAMPLE.OPJ` in Figure 13-1 on page 388, you can analyze the effects of variances in the values of resistors RC1 and RC2 by assigning a model description to these resistors that includes a 5% device tolerance on the multiplier parameter R.

Then you can perform a Monte Carlo analysis. First, the simulator performs a DC analysis with the nominal R multiplier value for RC1 and RC2. Then it performs a set number of additional runs with the R multiplier varied independently for RC1 and RC2 within a 5% tolerance.

Setting up a Simulation for Monte-Carlo Analysis

To modify example.opj and set up simulation, do the following:

1. Replace RC1 and RC2 with RBREAK parts from the `BREAKOUT.OLB` part library, setting property values to match the resistors that are being replaced (VALUE=10k) and reference designators to match previous names.
2. Select an RBREAK part and choose *PSpice Model* from the *Edit* menu.

   The *Model Editor* window appears.
3. Create the model CRES by replacing the model text:

   ```
   .model Rbreak RES R=1
   ```

   with the text:

   ```
   .MODEL CRES RES( R=1 DEV=5% TC1=0.02
   + TC2=0.0045 )
   ```

   Where TC1 is the linear temperature coefficient. TC2 is the quadratic temperature coefficient.
4. From the *File* menu in *Model Editor*, choose *Save*.

   The schematic editor automatically attaches the CRES model to the selected RBREAK part by updating the IMPLEMENTATION property on the part.

@ 2023 All Rights Reserved.

5. Double-click the second RBREAK part to display the Parts spreadsheet.
6. In the IMPLEMENTATION text box, change the value to CRES, then click *Apply*.
7. Close the Parts spreadsheet.
8. From the *File* menu, choose *Save*. By default, your schematic application saves the definition to the model library `EXAMPLE.LIB` and automatically configures the file for local use with the current schematic.
9. In your schematic application, set up a new Monte Carlo analysis as shown in Figure 13-1 on page 388. The analysis specification tells PSpice A/D to do one nominal run and four Monte Carlo runs, saving the DC analysis output from those five runs.

Figure 13-2 Monte Carlo analysis setup for EXAMPLE.OPJ

For DC sweep, specify the following *Primary Sweep* settings:

1. Select *Voltage* source in the Sweep variable frame.
2. Enter *V2* in the *Name* text box.
3. Select *Linear* in the *Sweep* type frame.
4. Enter a 0 in the *Start* value text box.
5. Enter a 12 in the *End* value text box.
6. Enter a 1 in the *Increment* text box.

@ 2023 All Rights Reserved.

PSpice A/D starts by running *all* of the analyses enabled in the *Simulation Settings* dialog box with all parameters set to their nominal values.

However, with Monte Carlo enabled, PSpice A/D saves the DC sweep analysis results for later reference and comparison. After the nominal analyses are finished, PSpice A/D performs the additional specified analysis runs (in this example, DC sweep).

Subsequent runs use the same analysis specification as the nominal run with one major exception: instead of using the nominal parameter values, the tolerances are applied to set new parameter values and thus, new part values.

There is a trade-off in choosing the number of Monte Carlo runs. More runs provide better statistics, but they require more time. The amount of time scales directly with the number of runs: 20 transient analyses take 20 times as long as one transient analysis. During Monte Carlo runs, the PSpice A/D status display includes the current run number and the total number of runs left.

Note: PSpice A/D offers a facility to generate histograms of data derived from Monte Carlo waveform families through the performance analysis feature.

History Support

The Monte Carlo analysis calculates the circuit response to changes in part values by randomly varying all of the model parameters for which a tolerance is specified. However, at times users might want to keep some or all of the parameters similar for multiple analysis so that they can compare the results of multiple simulations.

Consider an example of IC designs, which usually require numerous runs. In such cases, rather than doing large number of runs (with too many model parameters), users would prefer doing less number of runs (with less number of parameters), manually change some components/parameters values or add a small amount of circuitry, and then continue the analysis. Here user expects that simulator should use similar values for all model parameters, so that a comparison can be done between multiple simulations.

@ 2023 All Rights Reserved.

PSpice A/D allows you to save randomly generated values for a run in a .mcp file. You can then load this file to reuse the values for subsequent runs.

Reading the Summary Report

The summary report generated in this example (see Figure 13-3 on page 393) specifies that the waveform generated from V(OUT1) should be the subject of the collating function YMAX. In each of the last four runs, the new V(OUT1) waveform is compared to the nominal V(OUT1)

@ 2023
All Rights Reserved.

waveform for the first run, calculating the maximum deviation in the Y direction (YMAX collating function). The deviations are printed in order of size along with their run number.

```
 ****     SORTED DEVIATIONS OF V(OUT1,OUT2) TEMPERATURE =   35.000 DEG C

                        MONTE CARLO SUMMARY

*****************************************************************************



Mean Deviation =    -.2477
Sigma          =     .3035

 RUN                     MAX DEVIATION FROM NOMINAL

Pass    3                 .5729  (1.89 sigma)  lower   at V_V1 =    -.02
                         (  94.885% of Nominal)

Pass    4                 .3549  (1.17 sigma)  lower   at V_V1 =    -.02
                         (  96.832% of Nominal)

Pass    2                 .3122  (1.03 sigma)  lower   at V_V1 =    -.02
                         (  97.212% of Nominal)

Pass    5                 .2493  ( .82 sigma)  higher  at V_V1 =    -.02
                         ( 102.23% of Nominal)
```

Figure 13-3 Summary of Monte Carlo runs for EXAMPLE.OPJ

@ 2023 All Rights Reserved.

With the List option enabled, a report is also generated showing the parameter value used for each device in each run. In this case (see Figure 13-4 on page 394), run three shows the highest deviation.

```
 * C:\ORCAD\EX\EXAMPLE.SCH

 ****     UPDATED MODEL PARAMETERS          TEMPERATURE =   35.000 DEG C

                      MONTE CARLO PASS    3

*****************************************************************************



 **** CURRENT MODEL PARAMETERS FOR DEVICES REFERENCING cres
                       R_RC1           R_RC2
             R      1.0304E+00      9.5053E-01
```

Figure 13-4 Parameter values for Monte Carlo pass three.

@ 2023 All Rights Reserved.

Monte Carlo Analysis of a Pressure Sensor

This example shows how the performance of a pressure sensor circuit with a pressure-dependent resistor bridge is affected by manufacturing tolerances, using Monte Carlo analysis to explore these effects.

Drawing the schematic

To begin, construct the bridge as shown in the following figure:

Figure 13-5 Pressure sensor circuit.

Here are a few things to know when placing and connecting the part:

- To get the part you want to place, from the Place menu, choose Part.
- To rotate a part before placing it, press *R*.
- For V1 and Meter, place a generic voltage source using the VSRC part. When you place the source for the meter, change its name by double-clicking the part and typing `Meter` in the Reference cell in the Parts Spreadsheet.
- For R1-R7, place a resistor using the R part.

@ 2023 All Rights Reserved.

- Place the analog ground using the 0 ground symbol from the SOURCE.OLB part library.
- To connect the parts, from the Place menu, choose Wire.
- To move values or reference designators, click the value or reference designator to select it, then drag it to the new location.

Defining part values

For the pressure sensor, you need to do the following:

- Change the resistor values for R3, R5, R6, and R7 from their default value of 1 k.
- Set the DC value for the V1 voltage source.

Note: Because the Meter source is used to measure current, it has no DC value and can be left unchanged.

To change resistor values, do the following:

1. Double-click the value for a resistor.
2. Type the new value. Depending on the resistor you are changing, set its value to one of the following:

If you are changing this resistor...	Type this...
R3	`{1k*(1+P*Pcoeff/Pnom)}` **Note:** The value for R3—{1k*(1+P*Pcoeff/Pnom)}—is an expression that represents linear dependence of resistance on pressure. To complete the definition for R3, you will create and define global parameters for Pcoeff, P, and Pnom later on in this example.
R5	2k
R6	470
R7	25

@ 2023 All Rights Reserved.

3. Repeat steps 1 and 2 for each resistor on your schematic page.

To set the DC value for the V1 source and make it visible

1. Double-click the V1 source part.
2. In the Parts Spreadsheet, click in the cell under the DC column.
3. Type `1.35v`.
4. Click the Display button.
5. In the Display Format frame, choose the Value Only option to make the DC value (`1.35v`) visible on the schematic.
6. Click *OK*, then click Apply to apply the changes you have made to the part.
7. Close the Parts Spreadsheet.

Setting up the Parameters

To complete the value specification for R3, define the global parameters Pcoeff, P, and Pnom.

To define and initialize Pcoeff, P, and Pnom

1. Place a PARAM part on the schematic page.
2. Double-click the PARAM part to display the Parts Spreadsheet.
3. For each parameter, create a new property by clicking New and typing its name. Enter its corresponding value by clicking in the cell under the new property name and typing its value. Specify the parameter name and corresponding value as follows.

Table 13-2

Property	Value
Pcoeff	-0.06
P	0
Pnom	1.0

4. Click Apply to save the changes you have made then close the Parts Spreadsheet.

@ 2023 All Rights Reserved.

Using resistors with models

To explore the effects of manufacturing tolerances on the behavior of this circuit, you set device (DEV) and (LOT) tolerances on the model parameters for resistors R1, R2, R3, and R4 in a later step (see Defining tolerances for the resistor models). This means you need to use resistor parts that have model associations.

Because R parts do not have associated models (and therefore no model parameters), change the resistor parts to Rbreak parts that do have models.

Note: When PSpice A/D runs a Monte Carlo analysis, it uses tolerance values to determine how to vary model parameters during the simulation.

To replace R1, R2, R3, and R4 with the RBREAK part

1. Click R1 to select it.
2. Hold down the *Ctrl* key and click R2, R3 and R4 to add them to the selection set.
3. Press *Delete* to delete the selection set.
4. From the Place menu, choose Part.
5. Type `RBREAK` in the Part text box. (If RBREAK is not available, click the Add Library button and select `BREAKOUT.OLB` to configure it for use in Capture.)
6. Click OK.
7. Manually place the RBREAK part in the circuit diagram where R1, R2, R3 and R4 were located.
8. Double-click on each RBREAK part and change the reference designators as desired.

Saving the design

Before editing the models for the Rbreak resistors, save the schematic.

To save the design

1. From Capture's *File* menu, choose *Save*.

@ 2023 All Rights Reserved.

Defining tolerances for the resistor models

This section shows how to assign device (DEV) and lot (LOT) tolerances to the model parameters for resistors R1, R2, R3, and R4 using the model editor.

You can use the model editor to change the .MODEL or .SUBCKT syntax for a model definition. To find out more about the model editor, see Editing model text, or refer to the *PSpice A/D Reference Guide*.

To assign 2% device and 10% lot tolerances to the resistance multiplier for R1

1. Select R1.
2. From the *Edit* menu, choose *PSpice* Model.

   Capture searches the libraries for the Rbreak model definition and makes a copy to create an instance model.
3. To change the instance model name from Rbreak to Rmonte1, do the following:
   a. In the Model Text frame, double-click `Rbreak`.
   b. Type `RMonte1`.
4. To add a 2% device tolerance and a 10% lot tolerance to the resistance multiplier, do the following:
   a. Add the following to the .MODEL statement (after `R=1`):

      ```
      DEV=2% LOT=10%
      ```

      The model editing window is displayed.

1. From the File menu, choose Save.

By default, Capture saves the RMonte1 .MODEL definition to the `DESIGN_NAME.LIB` library, which is `PSENSOR.LIB`. Capture also automatically configures the library for local use. To find out more about adding model libraries to the configuration, see Configuring Model Libraries.

To have resistors R2 and R4 use the same tolerances as R1

1. In Capture's schematic page editor, select R2 and R4.
2. From the Edit menu, select Properties.
3. In the R2 row, click in the cell under the Implementation column and type `RMonte1`.

@ 2023 All Rights Reserved.

4. In the R4 row, click in the cell under the Implementation column and type `RMonte1`.

To assign 5% device tolerance to the resistance multiplier for R3

Figure 13-6 Model definition for RMonte1.

1. Select R3.
2. From the Edit menu, select PSpice Model.
3. In the Model Text frame, change the .MODEL statement to:

   ```
   .model RTherm RES R=1 DEV=5%
   ```

4. From the File menu, choose Save.

   Your schematic page is displayed.

@ 2023 All Rights Reserved.

Setting up the Analyses

This section shows how to define and enable a DC analysis that sweeps the pressure value and a Monte Carlo analysis that runs the DC sweep with each change to the resistance multipliers.

To set up the DC sweep

1. In the PSpice menu, choose *New Simulation Profile* or *Edit Simulation Profile*. (If this is a new simulation, enter the name of the profile and click *OK*.)

   The *Simulation Settings* dialog box appears.

Figure 13-7 Pressure sensor circuit with RMonte1 and RTherm model definitions.

1. Select DC Sweep in the Analysis type list box.
2. In the Sweep Variable frame, select Global Parameter.

@ 2023 All Rights Reserved.

3. Enter the following values:

Table 13-3

In this text box...	Type this...
Parameter name	P
Start value	0
End value	5.0
Increment	0.1

To set up the Monte Carlo analysis

1. Select the Monte Carlo/Worst Case option.
2. Check Monte Carlo if it is not already selected.
3. In the Number of runs text box, type 10.
4. In the Save data from list box, select All.
5. Type `I(Meter)` in the Output variable text box.
6. Click OK to save the simulation profile.

Running the analysis and viewing the results

To complete setup, simulate, and view results

1. From Capture's PSpice menu, choose Run to start the simulation

   When the simulation is complete, PSpice A/D automatically displays the selected waveform. Because PSpice A/D ran a Monte Carlo analysis, it saved multiple runs or sections of data. These are listed in the Available Sections dialog box.

2. From PSpice A/D's Trace menu, choose Performance Analysis.
3. Click the Select sections button.
4. In the Available Sections dialog box, click the All button.

@ 2023 All Rights Reserved.

5. Click OK.
6. To display current through the Meter voltage source, do the following:
   a. From Capture's PSpice menu, point to markers and choose Current into Pin.
   b. Place a current probe on the left-hand pin of the Meter source.
7. Switch to the Probe window to see the family of curves for I(Meter) as a function of P.

   Another way to view the family of curves without using schematic markers is as follows:
   a. From PSpice A/D's Trace menu, choose Add Trace.
   b. In the Simulation Output Variables list, double-click I(Meter).

Note: For more on analyzing Monte Carlo results in PSpice A/D, see the next section on Monte Carlo histograms.

Monte Carlo Histograms

You can display data derived from Monte Carlo waveform families as histograms. This is part of the performance analysis feature.

In this example, you simulate a fourth-order Chebyshev active filter, running a series of 100 AC analyses while randomly varying resistor and capacitor values for each run. Then, having defined performance analysis measurements for bandwidth and center frequency, you observe the statistical distribution of these quantities for the 100 runs.

Chebyshev Filter Example

The Chebyshev filter is designed to have a 10 kHz center frequency and a 1.5 kHz bandwidth. The stimulus specifications for V1, V2, and V3 are:

```
V1:   DC=-15
V2:   DC=+15
V3:   AC=1
```


@ 2023 All Rights Reserved.

The parts are rounded to the nearest available 1% resistor and 5% capacitor value. In this example, note how the bandwidth and the center frequency vary when 1% resistors and 5% capacitors are used in the circuit.

Figure 13-8 Chebyshev filter.

Creating models for Monte Carlo analysis

To vary the resistors and capacitors in the filter circuit, create models for these parts on which you can set device tolerances for Monte Carlo analysis. The `BREAKOUT.OLB` library contains generic devices for this purpose. The resistors and capacitors in this schematic are the Rbreak and Cbreak parts from `BREAKOUT.OLB`.

Using the Model Editor, modify the models for these parts as follows:

```
.model RMOD RES(R=1 DEV=1%)
.model CMOD CAP(C=1 DEV=5%)
```


@ 2023 All Rights Reserved.

Setting up the analysis

To analyze the filter, set up both an AC analysis and a Monte Carlo analysis. The AC analysis sweeps 50 points per decade from 100 Hz to 1 MHz. The Monte Carlo analysis is set to take 100 runs. Save data from all runs and set the output variable to V(OUT).

Creating histograms

Because the data file can become quite large when running a Monte Carlo analysis, to view just the output of the filter, you place a voltage probe at the output of the filter.

To collect data for the marked node only

1. From the PSpice menu, choose New Simulation Profile or Edit Simulation Profile. (If this is a new simulation, enter the name of the profile and click OK.)

   The Simulation Settings dialog box appears.

2. On the Data Collection tab, choose the At Markers Only option for each type of marker (Voltages, Currents, Power, Digital, Noise).

3. Click OK.

To run the simulation and load Probe with data

1. From Capture's PSpice menu, choose Run to start the simulation.

   When the simulation is complete, PSpice A/D automatically displays the selected waveform. Because PSpice A/D ran a Monte Carlo analysis, it saved multiple runs or sections of data. These are listed in the Available Sections dialog box.

2. In the Available Sections dialog box, click All.

3. Click OK.

To display a histogram for the 1 dB bandwidth

1. From PSpice A/D's Plot menu, choose Axis Settings.

2. Select the X Axis tab.

3. In the Processing Options frame, select the Performance Analysis check box.

4. Click OK.

@ 2023 All Rights Reserved.

The histogram display appears. The Y axis is the percent of samples.

5. From the Trace menu, choose Add Trace.
6. Choose Bandwidth.
7. In the Trace Expression box, specify Bandwidth(VDB(OUT),1).

Note: You can also display this histogram by using the performance analysis wizard to display Bandwidth (VDB(OUT),1).

To change the number of histogram divisions

1. From the Tools menu, choose Options.
2. In the Number of Histogram Divisions text box, replace `10` with `20`.
3. Click OK.

The histogram for 1 dB bandwidth is shown in the following figure:

@ 2023 All Rights Reserved.

Figure 13-9 1 dB bandwidth histogram.

The statistics for the histogram are shown along the bottom of the display. The statistics show the number of Monte Carlo runs, the number of divisions or vertical bars that make up the histogram, mean, sigma, minimum, maximum, 10th percentile, median, and 90th percentile.

- Ten percent of the measurement values is less than or equal to the 10th percentile number, and 90% of the measured values is greater than or equal to that number.
- If there is more than one measurement expression value that satisfies this criteria, then the 10th percentile is the midpoint of the interval between the measured values that satisfy the criteria. Similarly, the median and 90th percentile numbers represent measured values such that 50% and 90% (respectively) of the measured values are less than or equal to those numbers.
- Sigma is the standard deviation of the measurement expression values.

If needed, you can turn off the statistical data display as follows:

1. From the Tools menu, choose Options.
2. Clear the Display Statistics check box.
3. Click Save, and then OK.

You can also show the distribution of the center frequency of the filter.

To display the center frequency

1. From the Trace menu, choose Add Trace.
2. Choose CenterFreq.
3. In the Trace Expression box, specify CenterFreq(VDB(OUT),1).

@ 2023 All Rights Reserved.

The new histogram replaces the previous histogram. To display both histograms at once, choose Add Plot to Window on the Plot menu before choosing Add from the Trace menu. The histogram of the center frequency is as shown in the following figure:

Figure 13-10 Center frequency histogram.

Copying histogram data

You can use the copy function to transfer the raw histogram data points for a particular trace to the Windows clipboard. This allows you to save the data as a standard ASCII text file, or paste it directly into a report or other document for later reference.

To copy histogram data to the clipboard

1. Select the trace symbol, or the trace name, in the histogram.
2. From the Edit menu, choose Copy (or press *Ctrl*+*C*).

@ 2023 All Rights Reserved.

The histogram data points for the trace will be transferred to the Windows clipboard.

To copy the histogram display to the clipboard

1. From the Window menu, choose Copy to Clipboard.

   The histogram graph will be transferred to the Windows clipboard.

@ 2023 All Rights Reserved.

Sensitivity Analysis

Worst-case analysis or Sensitivity analysis is used to find the worst probable output of a circuit or system given the restricted variance of its parameters. For instance, if the values of R1, R2, and R3 can vary by ±10%, then the worst-case analysis attempts to find the combination of possible resistor values which result in the worst simulated output. As with any other analysis, there are three important parts: inputs, procedure, and outputs.

Inputs

In addition to the circuit description, you need to provide two pieces of information:

- the parameter tolerances
- a definition of what *worst* means

You can set tolerances on any number of the parameters that characterize a model.

Note: You can define models for nearly all primitive analog circuit parts, such as resistors, capacitors, inductors, and semiconductor devices. PSpice A/D reads the standard model parameter tolerance syntax specified in the .MODEL statement. For each model parameter, PSpice A/D uses the nominal, minimum, and maximum probable values, and the DEV and/or LOT specifiers; the probability distribution type (such as UNIFORM or GAUSS) is ignored.

The criterion for determining the *worst* values for the relevant model parameters is defined in the .WC statement as a function of any standard output variable in a specified range of the sweep.

In a given range, reduce the measurement to a single value by one of these five collating functions:

MAX	Maximum output variable value
MIN	Minimum output variable value
YMAX	Output variable value at the point where it differs the most with the nominal run
RISE_EDGE (value)	Sweep value where the output variable value crosses *above* a given threshold value
FALL_EDGE (value)	Sweep value where the output variable value crosses *below* a given threshold value

@ 2023 All Rights Reserved.

You can define *worst* as the highest (HI) or lowest (LO) possible collating function relative to the nominal run.

You can use analog behavioral models to measure waveform characteristics other than those detected by the available collating functions, such as rise time or slope. You can also use analog behavioral models to incorporate several voltages and currents into one output variable to which a collating function may be applied.

Procedure

To establish the initial value of the collating function, worst-case analysis begins with a nominal run using all model parameters at their nominal values.

Next, multiple sensitivity analyses determine the individual effect of each model parameter on the collating function. This is accomplished by varying model parameters, one at a time, in consecutive simulations. The direction (*better* or *worse*) in which the collating function changes with a small increase in each model parameter is recorded.

Finally, for the worst-case run, each parameter value is taken as far from its nominal as allowed by its tolerance, in the direction which should cause the collating function to be its worst (given by the HI or LO specification).

Note: This procedure saves time by performing the minimum number of simulations required to make an educated guess at the parameter values that produce the worst results. It also has some limitations, which are described in the following sections.

Outputs

A summary of the sensitivity analysis is printed in the PSpice A/D output file (`.OUT`). This summary shows the percent change in the collating function corresponding to a small change in each model parameter. If a .PROBE statement is included in the circuit file, then the results of the nominal and worst-case runs are saved for viewing in the Probe window.

Caution: An important condition for correct worst-case analysis

Worst-case analysis is not an optimization process; it does not *search* for the set of parameter values that result in the worst result.

It assumes that the worst case occurs when each parameter has been either pushed to one of its limits or left at its nominal value as indicated by the sensitivity analysis. It shows the true worst-case results when the collating function is ***monotonic*** within all tolerance combinations.

@ 2023 All Rights Reserved.

Otherwise, there is no guarantee. Usually you cannot be certain whether this condition is true, but insight into the operation of the circuit may alert you to possible anomalies.

@ 2023 All Rights Reserved.

Worst-case Analysis Example

The schematic shown in the following figure is for an amplifier circuit that is a biased BJT. This circuit is used to demonstrate how a simple worst-case analysis works. It also shows how *non-monotonic* dependence of the output on a single parameter can adversely affect the worst-case analysis.

Because an AC (small-signal) analysis is being performed, setting the input to unity means that the output, Vm([OUT]), is the magnitude of the gain of the amplifier. The only variable declared in this circuit is the resistance of *Rb2*. Because the value of *Rb2* determines the bias on the BJT, it also affects the amplifier's gain.

Figure 13-11 Simple biased BJT amplifier.

The following figure is the circuit file used to run one of the following:

- a parametric analysis (.STEP, shown enabled in the circuit file) that sets the value of resistor *Rb2* by stepping model parameter R through values spanning the specified DEV tolerance range, or
- a worst-case analysis (shown disabled in the circuit file) that allows PSpice A/D to determine the worst-case value for parameter R based upon a sensitivity analysis.

@ 2023 All Rights Reserved.

Only one of these analyses can run in any given simulation.

Note: The AC and worst-case analysis specifications (.AC and .WC statements) are written so that the worst-case analysis tries to minimize Vm([OUT]) at 100 kHz.

The netlist and circuit file are set up to run either a parametric (.STEP) or worst-case (.WC) analysis of the specified AC analysis. These simulations demonstrate the conditions under which worst-case analysis works well and those that can produce misleading results when output is not monotonic with a variable parameter.

```
* Worst-case analysis comparing monotonic and non-monotonic
* output with a variable parameter

.lib

***** Input signal and blocking capacitor *****
Vin   In      0     ac    1
Cin   In      B     1u

***** "Amplifier" *****
*   gain increases with small increase in Rb2, but
*   device saturates if Rb2 is maximized.
Vcc   Vcc     0     10
Rc    Vcc     C     1k
Q1    C       B     0     Q2N2222
Rb1   Vcc     B     10k
Rb2   B       0     Rbmod 720
.model Rbmod res(R=1 dev 5%)              ; WC analysis results
                                          ; are correct
* .model Rbmod res(R=1.1 dev 5%)          ; WC analysis misled
                                          ; by sensitivity

***** Load and blocking capacitor *****
Cout  C       Out   1u
Rl    Out     0     1k

* Run with either the .STEP or the .WC, but not both.
* This circuit file is currently set up to run the .STEP
* (.WC is commented out)

**** Parametric Sweep—providing plot of Vm([OUT]) vs. Rb2 ****
.STEP  Res  Rbmod(R)  0.8  1.2  10m

***** Worst-case analysis *****
* run once for each of the .model definitions stated above)
* WC  AC  Vm([Out])  min  range  99k  101k  list  output  all

.AC  Lin  3  90k  110k
.probe
.end
```


@ 2023 All Rights Reserved.

For demonstration, the parametric analysis is run first, generating the curve. This curve, derived using the YatX measurement illustrates the non-monotonic dependence of gain on *Rb2*.

```
YatX(1, X_value)=y1{1|sfxv(X_value)!1;}
```

Figure 13-13 YatX Measurement Expression

To do this yourself, place the measurement definition in a `PROBE.PRB` file in the circuit directory. Then start PSpice A/D, load all of the AC sweeps, set up the X axis for performance analysis, and add the following trace:

```
YatX(Vm([OUT]),100k)
```

Note: The YatX measurement is used on the simulation results for the parametric sweep (.STEP).

Next, the parametric analysis is commented out and the worst-case analysis is enabled. Two runs are made using the two versions of the *Rbmod* .MODEL statement shown in the circuit file. The model parameter, R, is a multiplier which is used to scale the nominal value of any resistor referencing the *Rbmod* model (*Rb2* in this case).

Figure 13-14 Amplifier netlist and circuit file.

The first .MODEL statement leaves the nominal value of *Rb2* at 720 ohms. The sensitivity analysis increments R by a small amount and checks its effect on Vm([OUT]). This slight increase in R causes an increase in the base bias voltage of the BJT, and increases the amplifier's gain, Vm([OUT]). The worst-case analysis correctly sets R to its minimum value for the lowest possible Vm([OUT]).

The second .MODEL statement scales the nominal value of *Rb2* by 1.1 to approximately 800 ohms. The gain still increases with a small increase in R, but a larger increase in R increases the base voltage so much that it drives the BJT into saturation and nearly eliminates the gain. The worst-case analysis is fooled by the sensitivity analysis into assuming that *Rb2* must be minimized to degrade the gain, but maximizing *Rb2* is much worse. Note that even an optimizer, which checks the local gradients to determine how the parameters should be varied, is fooled by this circuit.

@ 2023 All Rights Reserved.

The resulting curves are shown in the following figure:

Figure 13-15 Correct worst-case results.

@ 2023 All Rights Reserved.

In the above figure, output is monotonic within the tolerance range. Sensitivity analysis correctly points to the minimum value.

Figure 13-16 Incorrect worst-case results.

In the above figure, output is non-monotonic within the tolerance range, thus producing incorrect worst-case results.

Consider a slightly different scenario: Rb2 is set to 720 ohms so that maximizing it is not enough to saturate the BJT, but Rb1 is variable also. The true worst case occurs when Rb2 is maximized and Rb1 is minimized. Checking their individual effects is not sufficient, even if the circuit were simulated four times with each resistor in turn set to its extreme values.

@ 2023 All Rights Reserved.

Tips and Other Useful Information

VARY BOTH, VARY DEV, and VARY LOT

When VARY BOTH is specified in the .WC statement and a model parameter is specified with both DEV and LOT tolerances defined, the worst-case analysis may produce unexpected results. The sensitivity of the collating function is only tested with respect to LOT variations of such a parameter.

Figure 13-17 Schematic using VARY BOTH.

For example, during the sensitivity analysis, the parameter is varied once affecting all devices referring to it and its effect on the collating function is recorded. For the worst-case analysis,

@ 2023 All Rights Reserved.

the parameter is changed for all devices by LOT + DEV in the determined direction. See the example schematic and circuit file.

```
WCASE  VARY  BOTH      Test
Vin            1       0       10V
Rs             1       2       1K
Rwc1           2       3       Rmod     100
Rwc2           3       0       Rmod     100
.MODEL  Rmod  RES(R=1  LOT 10%  DEV 5%)
.DC   Vin      LIST    10
.WC  DC        V(3)    MAX     VARY BOTH        LIST     OUTPUT ALL
.ENDS
```

Figure 13-18 Circuit file using VARY BOTH.

In this case, V(3) is maximized if:

- *Rwc1* and *Rwc2* are both increased by 10% per the LOT tolerance specification, and
- *Rwc1* is decreased by 5% and *Rwc2* is increased by 5% per the DEV tolerance specification.

The final values for *Rwc1* and *Rwc2* should be 105 and 115, respectively. However, because *Rwc1* and *Rwc2* are varied together during the sensitivity analysis, it is assumed that both must be increased to their maximum for a maximum V(3). Therefore, both are increased by 15%.

The purpose of the technique is to reduce the number of simulations. For a more accurate worst-case analysis, you should first perform a worst-case analysis with VARY LOT, manually adjust the nominal model parameter values according to the results, then perform another analysis with VARY DEV specified.

Gaussian Distributions

Parameters using Gaussian distributions are changed by 1σ (one sigma) for the worst-case analysis.

Note: The default Gaussian distribution shall have the sigma value of 3.

@ 2023 All Rights Reserved.

A new Gaussian distribution *GaussUser* will have a default sigma value of 1.When this is selected, user can see the value of sigma displayed alongside. The sigma value can be changed from 1 to 9.

YMAX collating function

The purpose of the YMAX collating function is often misunderstood. This function does not try to maximize the deviation of the output variable value from nominal. Depending on whether HI or LO is specified, it tries to maximize or minimize the output variable value itself at the point where maximum deviation occurred during sensitivity analysis.

Note: This may result in maximizing or minimizing the output variable value over the entire range of the sweep. This collating function is useful when you know the direction in which the maximum deviation occurs.

RELTOL

During the sensitivity analysis, each parameter is varied (multiplied) by 1+RELTOL where RELTOL is specified in a .OPTIONS statement, or defaults to 0.001.

Sensitivity Analysis Results

The sensitivity analysis results are printed in the output file (.OUT). For each varied parameter, the percent change in the collating function and the sweep variable value at which the collating function was measured are given. The parameters are listed in *worst output*

@ 2023 All Rights Reserved.

order; for example, the collating function was its worst when the first parameter printed in the list was varied.

When you use the YMAX collating function, the output file also lists mean deviation and sigma values. These are based on the changes in the output variable from nominal at every sweep point in every sensitivity run.

Manual optimization

You can use worst-case analysis to perform *manual optimization* with PSpice A/D. The monotonicity condition is usually met if the parameters have a very limited range.

Performing worst-case analysis with tight tolerances on the parameters produces sensitivity and worst-case results (in the output file). You can use these to decide how the parameters should be varied to achieve the desired response. You can then make adjustments to the nominal values in the circuit file, and perform the worst-case analysis again for a new set of gradients.

Note: Parametric sweeps (.STEP), like the one performed in the circuit file, can be used to augment this procedure.

When to use Monte Carlo Analysis

Monte Carlo (.MC) analysis may be helpful when worst-case analysis cannot be used. Monte Carlo analysis can often be used to verify or improve on worst-case analysis results. Monte Carlo analysis randomly selects possible parameter values, which can be thought of as randomly selecting points in the *parameter space*. The worst-case analysis assumes that the worst results occur somewhere on the surface of this space, where parameters (to which the output is sensitive) are at one of their extreme values.

If this is not true, the Monte Carlo analysis may find a point at which the results are worse. To try this, replace .WC in the circuit file with .MC *<#runs>*, where *<#runs>* is the number of simulations you want to perform. More runs provide higher confidence results. The Monte Carlo summary in the output file lists the runs in decreasing order of collating function value.

Tip

To save disk space, do not specify any OUTPUT options.

Next, add the following option to the .MC statement, and simulate again.

```
OUTPUT LIST RUNS <worst_run#>
```


@ 2023 All Rights Reserved.

This performs only two simulations: the nominal and the worst Monte Carlo run. The parameter values used during the worst run are written to the output file, and the results of both simulations are saved.

Using Monte Carlo analysis with YMAX is a good way to obtain a conservative guess at the maximum possible deviation from nominal, since worst-case analysis usually cannot provide this information.

@ 2023 All Rights Reserved.

14

Digital Simulation

Digital simulation is the analysis of logic and timing behavior of digital devices over time. PSpice A/D simulates this behavior during transient analysis. When computing the bias point, PSpice A/D considers the digital devices in addition to any analog devices in the circuit.

PSpice A/D performs detailed timing analysis subject to the constraints specified for the devices. For example, flip-flops perform setup checks on the incoming clock and data signals. PSpice A/D reports any timing violations or hazards as messages written to the simulation output file and the waveform data file.

Related Topics

- Tracking timing violations and hazards

Simulating Digital Circuits

Do the following steps to develop and simulate digital circuits:

1. Drawing the design.
2. Defining the stimuli.
3. Setting the simulation time.
4. Adjusting the simulation parameters.
5. Starting the simulation.
6. Analyzing the results.

@ 2023 All Rights Reserved.

Concepts of Digital Simulation

Following are the concepts of digital simulation:

States

When the circuit is in operation, digital nodes take on values or output states shown in Table 14-1. Each digital state has a *strength* component as well. Strengths are described in the next section.

Table 14-1 Digital states

State	Description
0	Low, false, no, off
1	High, true, yes, on
R	Rising (changes from 0 to 1 sometime during the R interval)
F	Falling (changes from 1 to 0 sometime during the F interval)
X	Unknown: may be high, low, intermediate, or unstable
Z	High impedance: may be high, low, intermediate, or unstable

Note: States do not necessarily correspond to a specific, or even stable, voltage. A logical 1 level means only that the voltage is somewhere within the *high* range for the particular device family. The rising and falling levels only indicate that the voltage crosses the 0–1 threshold at some time during the R or F interval, not that the voltage change follows a particular slope.

@ 2023 All Rights Reserved.

Strengths

When a digital node is driven by more than one device, PSpice A/D determines the correct level of the node. Each output has a *strength* value, and PSpice A/D compares the strengths of the outputs driving the node. The *strongest* driver determines the resulting level of the node. If outputs of the same strength but different levels drive a node, the node's level becomes X.

PSpice A/D supports 64 strengths. The lowest (weakest) strength is called *Z*. The highest (strongest) strength is called the *forcing* strength. The Z strength (called high impedance) is typically output by disabled tristate gates or open-collector output devices. PSpice A/D reports any nodes of Z strength (at any level) as Z, and reports all other nodes by the designations.

@ 2023 All Rights Reserved.

Defining a Digital Stimulus

A digital stimulus defines input to the digital portions of your circuit, playing a role similar to that played by the independent voltage and current sources for the analog portion of your circuit.

The following table summarizes the digital stimuli provided in the part libraries.

Table 14-2

If you want to specify the input signal by...	Then use this part...	For this type of digital input...
Using the Stimulus Editor	DIGSTIMn	signal or bus stimulus
Defining part properties	DIGCLOCK	clock signal
	STIM1	one-bit stimulus
	STIM4	four-bit stimulus
	STIM8	eight-bit stimulus
	STIM16	sixteen-bit stimulus
	FILESTIM1	one-bit file-based stimulus
	FILESTIM2	two-bit file-based stimulus
	FILESTIM4	four-bit file-based stimulus
	FILESTIM8	eight-bit file-based stimulus
	FILESTIM16	sixteen-bit file-based stimulus
	FILESTIM32	thirty-two-bit file-based stimulus

@ 2023 All Rights Reserved.

Using the DIGSTIMn part

Use the DIGSTIMn stimulus parts to define a stimulus for a net or bus using the Stimulus Editor.

To use the DIGSTIMn part

1. From Capture's Place menu, choose Part.
2. Place and connect the DIGSTIM1 stimulus part from SOURCSTM.OLB to a wire or bus in your design.
3. Click the stimulus instance to select it.
4. From the Edit menu, choose PSpice Stimulus.

   This starts the Stimulus Editor. The New Stimulus dialog box appears prompting you to define a new stimulus.
5. Enter DIGSTIM1 in the Name text box.
6. In the Digital frame, select Signal.
7. Click OK.
8. Define stimulus transitions.

Defining Input Signals using Stimulus Editor

Defining Signal Transitions

You can do any of the following when defining digital signal transitions:

- Add a transition
- Move a transition
- Edit a transition
- Delete a transition

Note: These operations cannot be applied to a stimulus defined as a clock signal.

@ 2023 All Rights Reserved.

To add a transition

1. From the Plot menu, choose Axis Settings.
2. Enter values in the Displayed Range for Time text boxes and the Timing Resolution text box as required for adding transitions.

   For example, if you wish to add transitions every 1ms, set the Timing Resolution to 1ms.
3. Select the digital stimulus you want to edit.

   When you select a transition to edit, a red handle appears.
4. From the Stimulus Editor's Edit menu, choose Add.
5. Click the waveform at the time where the transition is required.
6. Repeat step 4 to add additional transitions.
7. When you finish, right-click to exit the edit mode.

To move a transition

1. Click the transition you want to move.
2. If needed, use *Shift*+click to select additional transitions on the same signal or different signals.
3. Reposition the transition (or transitions) by dragging.

Note: If you press *Shift* while dragging, then all selected transitions move by the same amount.

To edit a transition

1. Do one of the following:
   - ❑ Select the transition you want to edit and from the Edit menu, choose Attributes.
   - ❑ Double-click the transition you want to edit.
2. In the Edit Digital Transition dialog box, edit the timing and value of the transition.
3. Click OK.

To delete a transition

1. Click the transition you want to delete.

@ 2023 All Rights Reserved.

2. If needed, press *Shift*+click to select additional transitions on the same signal or different signals.
3. From the Edit menu, choose Delete.

Defining Clock Transitions

To create a clock stimulus

1. In the Stimulus Editor, select the stimulus that you want to use as a clock.
2. From the Stimulus menu, choose Change Type.
3. Under Type, choose Clock.
4. Click OK.
5. Enter values for the clock signal attributes.

Table 14-3

For this attribute...	Enter this...
Frequency	clock rate
Duty Cycle	percent of high versus low in decimal or integer units
Initial Value	starting value: 0 or 1
Time Delay	time after simulation begins when the clock stimulus takes effect

Example: To create a clock signal with a clock rate of 20 MHz, 50% duty cycle, a starting value of 1, and time delay of 5 nsec, set the signal properties as follows:

```
Frequency       = 20Meg
Duty Cycle      = 0.50 (or 50)
Initial Value   = 1
Time Delay      = 5ns
```

6. From the File menu, choose Save.

@ 2023 All Rights Reserved.

To change clock attributes

1. In the Stimulus Editor, do one of the following:
   - ❑ Double-click the clock name to the left of the axis.
   - ❑ Click the clock name and from the Edit menu, choose attributes.
2. Modify the clock attributes as needed.
3. Click OK.

Defining bus transitions

There are three steps for creating a bus:

1. Creating the digital bus stimulus.
2. Introducing transitions.
3. Optionally defining the radix for bus values.

These steps are described in detail in the following procedures.

To create a digital bus stimulus

1. From the Stimulus menu, choose New.
2. In the Name text box, enter Bus.
3. In the Digital frame, select Bus.
4. If needed, change the bus width from its default value of 8 bits. To do this, in the Width text box, type a different integer.
5. Click OK.

During any interval, the bits on the bus lines represent a value from zero through (2^n - 1), where *n* is the number of bus lines. To set bus values, introduce transitions using either of the two methods described.

To introduce transitions (method one)

1. From the Plot menu, choose Axis Settings.
2. Enter values in the Displayed Range for Time text boxes and the Timing Resolution text box as required for adding transitions.

@ 2023 All Rights Reserved.

For example, if you wish to add transitions every 1 ms, set the Timing Resolution to 1ms.

3. From the Stimulus Editor's Edit menu, choose Add.
4. In the digital value field on the toolbar (just right of the Add button), type a bus value in any of the following ways:

Table 14-4

To get this effect...	Type this...
A literal value	<*unsigned_number*>[;*radix*] Example: `12`
An increment	+<*unsigned_number*>[;*radix*] Example: `+12;H`
A decrement	-<*unsigned_number*>[;*radix*] Example: `-12;O`

If you do not enter a radix value, the Stimulus Editor appends the default bus radix. To find out about valid radix values, see page 14-11.

5. Click the waveform where you want the transition added.
6. Repeat steps 4 and 5 as needed.
7. When you finish, right-click to exit the editing mode.

To introduce transitions (method two)

1. From the Plot menu, choose Axis Settings.
2. Enter values in the Displayed Range for Time text boxes and the Timing Resolution text box as required for adding transitions.

   For example, if you wish to add transitions every 1 ms, set the Timing Resolution to 1ms.
3. From the Stimulus Editor's Edit menu, choose Add.

@ 2023 All Rights Reserved.

4. Place the tip of the pencil-shaped pointer on the waveform, and click to create transitions as shown here:

Here are some other things that you can do:

- ❑ Move a transition left or right by clicking and dragging.
- ❑ Delete a transition by selecting it and then, from the Edit menu, choosing Delete (or by pressing *Del*).
- ❑ Select more than one transition by holding down *Shift* while clicking.

5. When you finish creating transitions, right-click.
6. Click the transition at the start (far left) of the interval. A small diamond appears over the transition.
7. From the Edit menu, choose Attributes to display the Edit Digital Transition dialog box.
8. In the Transition Type frame, choose Set Value, Increment, or Decrement.
9. Do one of the following to specify the bus value:
   - ❑ In the Value text box, type a value.
   - ❑ Select one of these defaults from the list: 0, All bits 1, X (Unknown), or Z (High impedance).
10. Click OK.
11. Repeat steps 6 through 10 for each transition.

To set the default bus radix

1. From the Tools menu, choose Options.

@ 2023 All Rights Reserved.

2. In the Bus Display Defaults frame, from the Radix list, select the radix you want as default.

Table 14-5

Select this radix...	To show values in this notation...
Binary	base 2
Octal	base 8
Decimal	base 10
Hexadecimal	base 16

3. Click OK.

Adding loops

Suppose you have a stimulus that looks like this:

and you want to create a stimulus that consists of three consecutive occurrences of the sequence that starts at A and ends at B:

You can do this by using a standard text editor to edit a stimulus library file. Within this file is a sequence of transitions that produces the original waveform. With a text editor you can modify the stimulus definition so it repeats itself.

To add a loop

1. In the Stimulus Editor, save and close the stimulus file.
2. In a standard text editor (such as Notepad), open the stimulus file.

@ 2023 All Rights Reserved.

3. Find the set of consecutive lines comprising the sequence that you want to repeat.

   Each relevant line begins with the time of the transition and ends with a value or change in value.

4. Before these lines, insert a line that uses this syntax:

   `+ Repeat for` *`n_times`*

   where *n_times* is one of the following:

   - A positive integer representing the number of repetitions.
   - The keyword FOREVER, which means repeat this sequence for an unlimited number of times (like a clock signal).

5. Insert a line that uses this syntax:

   ```
   + Endrepeat
   ```

6. From the File menu, choose Save.

Given the example shown on page 3, if you wanted to repeat the sequence shown from point A to point B three times, then you would modify the stimulus file as shown here (added lines are in bold):

```
+ Repeat for 3
+ +0s 000000000
+ 250us INCR BY 000000001
+ 500us 000000010
+ 750us INCR BY 000000001
+ 1ms 000000000
+ Endrepeat
```

Using the DIGCLOCK Part

The DIGCLOCK part allows you to define a clock signal by using the part's properties.

For information on how to define a clock signal using the Stimulus Editor with the DIGSTIMn part, see Defining Signal Transitions on page 427.

To define a clock signal using DIGCLOCK

1. From Capture's Place menu, choose Part.
2. Place and connect a DIGCLOCK part.
3. Double-click the part instance.

@ 2023 All Rights Reserved.

4. Define the properties as described.

Table 14-6

For this property...	Specify this...
DELAY	Time before the first transition of the clock
ONTIME	Time in high state for each period
OFFTIME	Time in low state for each period
STARTVAL	Low state of clock (default:0)
OPPVAL	High state of clock (default: 1)

The support for expressions in PSpice Modeling Application has now been extended to digital clock source, *DigClock*.

Using STIM1, STIM4, STIM8 and STIM16 parts

The STIM*n* parts have a single pin for connection. STIM1 is used for driving a single net. STIM4, STIM8 and STIM16 drive buses that are 4, 8 and 16 bits wide, respectively. The properties for all of these parts are the same as those shown in Table 14-7.

Table 14-7

Property	Description
WIDTH	Number of output signals (nodes).

@ 2023 All Rights Reserved.

Table 14-7

Property	Description
FORMAT	Sequence of digits defining the number of signals corresponding to a digit in any <*value*> term appearing in a COMMAND*n* property definition. Each digit must be either 1, 3, or 4 (binary, octal, hexadecimal, respectively); the sum of all digits in FORMAT must equal WIDTH.
IO_MODEL	I/O model describing the stimulus' driving characteristics.
IO_LEVEL	Interface subcircuit selection from one of the four analog/digital subcircuits provided with the part's I/O model.
DIG_PWR	Digital power pin used by the interface subcircuit.
DIG_GND	Digital ground pin used by the interface subcircuit.
TIMESTEP	Number of seconds per clock cycle or step.
COMMAND1-COMMAND16	Stimulus transition specification statements including time/value pairs, labels, and conditional constructs.

When placed, you must connect each part to the wire or bus of the corresponding radix. Generally, you only need to modify the FORMAT, TIMESTEP, and COMMAND*n* properties.

Typically, each COMMAND*n* property contains only one command line. It is possible to enter more than one command line per property by placing \n+ between command lines in a given definition. (The *n* must be lower case and no spaces between characters; spaces may precede or follow the entire key sequence).

Using the FILESTIM*n* parts

The FILESTIM*n* parts have a single pin for connection to the rest of the circuit. FILESTIM1 is used for driving a single net. FILESTIM2, FILESTIM4, FILESTIM8, FILESTIM16 and FILESTIM32 drive buses that are 2, 4, 8, 16 and 32 bits wide, respectively. You must define the digital stimulus specification in an external file. Using this technique, stimulus definitions

@ 2023 All Rights Reserved.

can be created from scratch or extracted with little modification from another simulation's output file.

Table 14-8 lists the properties of the FILESTIM*n* parts. The IO_MODEL, IO_LEVEL, and PSPICEDEFAULTNET properties describing this part's I/O characteristics are provided with default values that rarely need modification. However, you must define the FILENAME property with the name of the external file containing the digital stimulus specification.

The SIGNAME property specifies the name of the signal inside the stimulus file which becomes the output from the FILESTIM*n* part. If left undefined, the name of the connected net (generally a labeled wire) determines which signal is used.

Table 14-8 FILESTIM*n* part properties

Property	Description
FILENAME	Name of file containing the stimulus specification. **Note:** If you do not specify the path to the stimulus file, you must place the file in the folder for the simulation profile for which you are configuring the stimulus.
SIGNAME	Name of output signal
IO_MODEL	I/O model describing the stimulus' driving characteristics
IO_LEVEL	Interface subcircuit selection from one of the four AtoD or DtoA subcircuits provided with the part's I/O model
PSPICEDEFAULTNET	Hidden digital power and ground pins used by the interface subcircuit. Name of the default net to use.

@ 2023 All Rights Reserved.

For example, a FILESTIM*n* part can be used to reset a counter, which could appear as shown in the following figure:

Figure 14-1 FILESTIM1 used on a schematic page.

In this case, the FILESTIM1 part instance, U2, generates a reset signal to the CLR pin of the 74393 counter.

To set up the U2 stimulus

The following steps set up the U2 stimulus so that the 74393 counter is cleared after 40 nsec have elapsed in a transient analysis.

1. Create a stimulus file named `RESET.STM` that contains the following lines:

   ```
   Reset

   0ns  1
   40ns 0
   ```

   The header line contains the names of all signals described in the file. In this case, there is only one: Reset.

   The remaining lines are the state transitions output for the signals named in the header. In this case, the Reset signal remains at state 1 until 40nsec have elapsed, at which time it drops to state 0.

   Note: A blank line is required between the signal name list and the first transition.

2. Place the `RESET.STM` file in the folder for the simulation profile for which you are configuring the stimulus.

3. Associate this file with the digital stimulus instance, U2, by setting U2's FILENAME property to `RESET.STM`.

@ 2023 All Rights Reserved.

4. Define the signal named Reset in `RESET.STM` as the output of U2 by setting U2's SIGNAME property to Reset. Since the labeled wire connecting U2 with the 74393 counter is also named Reset, it is also acceptable to leave SIGNAME undefined.

@ 2023 All Rights Reserved.

Defining Simulation Time

To set up the transient analysis

1. From Capture's PSpice menu, choose New Simulation Profile.
2. Enter a name for the new simulation profile.
3. Click OK.
4. In the Analysis Type list box on the Analysis tab, select Time Domain (Transient).
5. In the Run to Time text box, type the duration of the transient analysis.
6. Click OK.

@ 2023 All Rights Reserved.

Adjusting Simulation Parameters

Use the *Options* tab of the Simulation Settings dialog box to adjust the simulation behavior of your circuit's digital devices.

To access the digital settings in the Options tab

1. From Capture's PSpice menu, choose Edit Simulation Profile.
2. Click the Options tab.
3. In the tree structure, select Gate-level simulation.

   Each of the dialog box settings is described in the following sections. For additional options, see Output control options on page 453.

@ 2023 All Rights Reserved.

Selecting Propagation Delays

All digital devices—including primitives and library models—perform simulations using either minimum, typical, maximum or worst-case (min/max) timing characteristics. You can set the delay circuit-wide or on individual device instances.

Circuit-wide propagation delays

You can set these to minimum, typical, maximum or variable within the min/max range for digital worst-case timing simulation on the Options tab of the Simulation Settings dialog box.

To specify the delay level circuit-wide

1. From Capture's PSpice menu, choose Edit Simulation Profile.
2. Click the Options tab.
3. In the Category list box, select Gate-level simulation.

Part instance propagation delays

You can set the propagation delay mode on an individual device, thereby overriding the circuit-wide delay mode.

To override the circuit-wide default on an individual part

1. Set the part's MNTYMXDLY property from 1 to 4 where

   1 = minimum

   2 = typical

   3 = maximum

   4 = worst-case (min/max)

   By default, MNTYMXDLY is set to 0, which tells PSpice A/D to use the circuit-wide value defined in the Options tab.

@ 2023 All Rights Reserved.

Initializing Flip-flops

To initialize all flip-flops and latches, do the following:

Select one of the three Flip-flop Initialization choices on the *Options* tab:

1. If set to X, all flip-flops and latches produce an X (unknown state) until explicitly set or cleared, or until a known state is clocked in.
2. The X initialization is the *safest* setting, since many devices do not power up to a known state. However, the 0 and 1 settings are useful in situations where the initial state of the flip-flop is unimportant to the function of the circuit, such as a toggle flip-flop in a frequency divider.
3. If set to 0, all such devices are cleared.
4. If set to 1, all such devices are preset.

Starting a Simulation

To start the simulation, do the following:

➡ From the *PSpice* menu, choose *Run*.

After PSpice A/D completes the simulation, the graphical waveform analyzer starts automatically.

@ 2023 All Rights Reserved.

Analyzing Results

PSpice A/D includes a graphical waveform analyzer for simulation results. In effect, the waveform viewer in PSpice A/D is a software oscilloscope. Running PSpice A/D corresponds to building or changing a breadboard, and the waveform viewer corresponds to looking at the breadboard with an oscilloscope. You can observe and interactively manipulate the waveform data produced by circuit simulation. For a full discussion of how the waveform viewer is used to analyze results, see Analyzing Waveforms with PSpice

For mixed analog/digital simulations, the waveform analyzer can display analog and digital waveforms simultaneously with a common time base.

PSpice A/D generates two forms of output: the simulation output file and the waveform data file. The calculations and results reported in the simulation output file are like an audit trail of the simulation. However, the graphical analysis of information stored in the data file is a more informative and flexible method for evaluating simulation results.

To display waveforms

1. From the Trace menu, choose Add Trace.

@ 2023 All Rights Reserved.

2. Select traces for display:
   - In the Simulation Output Variables list, click any waveforms you want to display. Each appears in the Trace Expressions box at the bottom.
   - Construct expressions by selecting operators, functions and/or macros from the Functions or Macros list, and output variables in the Simulation Output Variables list.
   - You can also type trace expressions directly into the Trace Expression text box. A typical set of entries might be:

     ```
     IN1 IN2 Q1 Q2
     ```

     Note: Use spaces or commas to separate the output variables you place in the Trace Expressions list.
3. Click OK.

Waveforms for the selected output variables appear.

For detailed information on digital traces, see Digital Trace Expressions.

Adding digital signals to a plot

When defining digital trace expressions, you can include any combination of digital signals, buses, signal constants, bus constants, digital operators, macros and the Time sweep variable.

The following rules apply:

- An arithmetic or logical operation between two bus operands results in a bus value that is wide enough to contain the result.
- An arithmetic or logical operation between a bus operand and a signal operand results in a bus value.

The syntax for expressing a digital output variable or expression is:

digital_output_variable[;*display_name*]

or

@ 2023 All Rights Reserved.

digital_expression[;*display_name*]

Table 14-9

This placeholder...	Means this...
digital_output_variable	output variable from the Simulation Output Variable list (Digital check box selected)
digital_expression	expression using digital output variables and operators
display_name (optional)	text string (name) to label the signal on the plot, instead of using the default output variable notation

To add a digital trace expression

1. In the Add Traces dialog box, make sure you select the Digital check box.
2. Do one of the following:
   - In the Simulation Output Variables list, click the signal you want to display.
   - In the Trace Expression text box, create a digital expression by either typing the expression, or by selecting digital output variables from the Simulation Output Variables list and digital operators from the Digital Operators and Functions list.
3. If you want to label a signal with a name that is different from the output variable:
   a. Click in the Trace Expression text box after the last character in the signal name.
   b. Type ;*display_name* where *display_name* is the name of the label.

   Example: `U2:Y;OUT1`
   where U2:Y is the output variable. On the plot, the signal is labeled OUT1.

@ 2023 All Rights Reserved.

Adding buses to a waveform plot

You can evaluate and display a set of up to 32 signals as a bus *even if the selected signals were not originally a bus*. This is done by following the same procedure already given for adding digital signals to the plot. However, when adding a bus, be sure to enclose the list of signals in braces: { }.

```
{ Q3 Q2 Q1 Q2 }
```

The complete syntax is as follows:

{*signal_list*}[;[*display_name*][;*radix*]]

or

{*bus_prefix*[*msb*:*lsb*]}[;[*display_name*][;*radix*]]

Table 14-10

This placeholder...	Means this...
signal_list	comma- or space-separated list of up to 32 digital node names, in sequence from high order to low order
bus_prefix[msb:lsb]	alternate way to express up to 32 signals in the bus
display_name (optional)	text string (name) to label the bus on the plot, instead of using the default output variable notation **Note:** To change the radix without changing the display name, be sure to include two consecutive semicolons. Example: `{A3,A2,A1,A0};;radix`
radix (optional)	numbering system in which to display bus values

@ 2023 All Rights Reserved.

Valid entries for *radix* are shown in the following table.

Table 14-11

For this numbering system...	Use this notation...
Binary (base 2)	B
Decimal (base 10)	D
Hexadecimal (base 16)	H or X
Octal (base 8)	O (the letter)

To add a bus expression

1. In the Add Traces dialog box, in the Functions and Macros list, choose Digital Operators and Constants.
2. Click the `{ }` entry.
3. In the Simulation Output Variables list, select the signals in high-order to low-order sequence.
4. If you want to label the bus with a name that is different from the default:
   - **a.** Click in the Trace Expression text box after the last character in the bus name.
   - **b.** Type ;*display_name* where *display_name* is the name of the label.
5. If you want to set the radix to something different from the default:
   - **a.** Click in the Trace Expression text box after the last character in the expression.
   - **b.** Type one of the following where *radix* is a value from Table 14-11:
     - If you specified a *display_name*, then type *;radix*.
     - If you did not specify a *display_name*, then type *;;radix* (two semicolons preceding the radix value).

Examples:

- `{Q2,Q1,Q0};A;O` specifies a 3-bit bus whose most significant bit is Q2. PSpice A/D labels the plot A, and values appear in octal notation.
- `{a3,a2,a1,a0};;d` specifies a 4-bit bus. On the plot, values appear in decimal notation. Since no display name is specified, PSpice A/D uses the signal list as a label.

@ 2023 All Rights Reserved.

- `{a[3:0]}` is equivalent to `{a3,a2,a1,a0}`

@ 2023 All Rights Reserved.

Tracking timing violations and hazards

When there are problems with your design, such as setup/hold violations, pulse-width violations, or worst-case timing hazards, PSpice A/D saves messages to the simulation output file or data file. You can select messages and have the associated waveforms and detailed message text automatically appear.

PSpice A/D can also detect persistent hazards that may have a potential effect on a primary circuit output or on the internal state of the design.

Persistent hazards

Digital problems are usually either timing violations or timing hazards. Timing violations include SETUP, HOLD and minimum pulse WIDTH violations of component specifications. This type of violation may produce a change in the state behavior of the design, and potentially in the answer. However, the effects of many of these errors are short-lived and do not influence the final circuit results.

For example, consider an asynchronous data change on the input to flip-flop FF1. The data change is too close to the clock edge e1, resulting in a SETUP violation. In a hardware implementation, the output of FF1 may or may not change. However, some designs are not sensitive to this individual missed data because the next clock edge (e2 in this example) latches the data. The designer must judge the significance of timing errors, accounting for the overall behavior of the design.

Figure 14-2 Circuit with a timing error.

Timing hazards are most easily identified by simulating a design in worst-case timing mode, usually close to its critical timing limits. Under such conditions, PSpice A/D reports conditions such as AMBIGUITY CONVERGENCE hazards. Again, these may or may not pose a problem to the operation of the design.

@ 2023 All Rights Reserved.

However, there are identifiable cases that cause major problems. An example of a major problem is shown in the following figure. Due to the simultaneous arrival of two timing ambiguities (having unrelated origins, therefore nothing in common) at the inputs to gate G1, PSpice A/D reports the occurrence as an AMBIGUITY CONVERGENCE hazard. This means that the output of G1 may glitch.

Figure 14-3 Circuit with a timing ambiguity

Note that the output fans out to two devices, G2 and L1. The effects of a glitch on G1 in this case do not reach the circuit output P1, because that path is not sensitized (since the other input to G2 is held LO and thus blocks the symptom). However, because G1's output is also used to clock latch L1, the effects of a glitch could result in visibly incorrect behavior on output P2. This is an example of a persistent hazard.

A persistent hazard is a timing violation or hazard that has a potential effect on a primary (external) circuit output or on the internal state (stored state or memory elements) of the design. For the design to be considered reliable, you must correct such timing hazards.

PSpice A/D fully distinguishes between state uncertainty and time uncertainty. When a hazard occurs, PSpice A/D propagates hazard origin information along with the machine state through all digital devices. When a hazard propagates to a state-storage device primitive (JKFF, DFF, SRFF, DLTCH, RAM), PSpice A/D reports a PERSISTENT HAZARD.

Simulation Condition Messages

PSpice A/D produces warning messages in various situations, such as those that originate from the digital CONSTRAINT devices monitoring timing relationships of digital nodes. These messages are directed to the simulation output file and/or to the waveform data file. Options are available for controlling where and how many of these messages are generated, as summarized later in this section.

@ 2023 All Rights Reserved.

Table 14-12 summarizes the simulation message types, with a brief description of their meaning. Currently, the messages supported are specific to digital device timing violations and hazards.

Table 14-12 Simulation condition messages—timing violations

Message type	Severity level	Meaning
SETUP	WARNING	Minimum time required for a data signal to be stable *prior* to the assertion of a clock was not met.
HOLD	WARNING	Minimum time required for a data signal to be stable *after* the assertion of a clock was not met.
RELEASE	WARNING	Minimum time required for a signal that has gone inactive (usually a control such as CLEAR) to remain inactive before the asserting clock edge was not met.
WIDTH	WARNING	Minimum pulse width specification for a signal was not satisfied; that is, a pulse that was too narrow was observed on the node.
FREQUENCY	WARNING	Minimum or maximum frequency specification for a signal was not satisfied. Minimum frequency violations indicate that the period of the measured signal is too long, while maximum frequency violations describe signals changing too rapidly.
GENERAL	INFO	Boolean expression described within the GENERAL constraint checker was evaluated and produced a *true* result.

Table 14-13 Simulation condition messages—hazards

Message type	Severity level	Meaning
AMBIGUITY CONVERGENCE	WARNING	Convergence of conflicting rising and falling states (timing ambiguities) arrived at the inputs of a primitive and produced a pulse (glitch) on the output. See Digital Worst-case Timing Analysis for more information.

@ 2023 All Rights Reserved.

Table 14-13 Simulation condition messages—hazards

Message type	Severity level	Meaning
CUMULATIVE AMBIGUITY	WARNING	Signal ambiguities are additive, increased by propagation through each level of logic in the circuit. The ambiguities associated with both edges of a pulse increased to the point where they overlapped, which PSpice A/D reports as a cumulative ambiguity hazard. See Digital Worst-case Timing Analysis for more information.
SUPPRESSED GLITCH	WARNING	Pulse applied to the input of a primitive that is shorter than the active propagation delay was ignored by PSpice A/D; significance depends on the nature of the circuit. There might be a problem either with the stimulus, or with the path delay configuration of the circuit. See Digital Worst-case Timing Analysis for more information.
NET-STATE CONFLICT	WARNING	Two or more outputs attempted to drive a net to different states, which PSpice A/D reports as an X (unknown) state. This usually results from improper selection of a bus driver's enable inputs.
ZERO-DELAY-OSCILLATION	FATAL	Output of a primitive changed more than 50 times within a single digital time step. PSpice A/D aborted the run.
DIGITAL INPUT VOLTAGE	SERIOUS	Voltage on a digital pin was out of range, which means PSpice A/D used the state with a voltage range closest to the input voltage and continued the simulation.
PERSISTENT HAZARD	SERIOUS	Effects of any of the aforementioned logic hazards were able to propagate to either an external port or to any storage device in the circuit. See Persistent hazards on page 450 for more information.

Output control options

Four control options are available for managing the generation of simulation condition messages. These are described in Table 14-14.

@ 2023 All Rights Reserved.

To access these commands, select the Options tab in the Simulation Settings dialog box. You can set NOOUTMSG and NOPRBMSG by selecting the Output file option. You can set DIGERRDEFAULT and DIGERRLIMIT by selecting the Gate-level simulation tree structure.

Table 14-14 Simulation message output control options

This option...	Means this...
NOOUTMSG	Suppresses the recording of simulation condition messages in the simulation output file.
NOPRBMSG	Suppresses the recording of simulation condition messages in the waveform data file.
DIGERRDEFAULT=<*n*>	Establishes a default limit, *n*, to the number of condition messages that may be generated by any digital device that has a constraint checker primitive without a local default. If global or local defaults are unspecified, there is no limit.
DIGERRLIMIT=<*n*>	Establishes an upper limit, *n*, for the total number of condition messages that may be generated by any digital device. If this limit is exceeded, PSpice A/D aborts the run. By default, the total number of messages is 20.

Severity levels

PSpice assigns one of four severity levels to the messages:

- FATAL
- SERIOUS
- WARNING
- INFO (informational)

@ 2023 All Rights Reserved.

FATAL conditions cause PSpice A/D to cancel the simulation. Under all other severity levels, PSpice A/D continues to run. The severity levels are used to filter the classes of messages that are displayed when loading a data file.

@ 2023 All Rights Reserved.

@ 2023 All Rights Reserved.

15

Mixed Analog/Digital Simulation

Prior to simulation, netlisting translates the part instances and nets defined in your schematic into parts connected by nodes. The standard simulation netlist contains a flat view of the circuit. You can also create hierarchical netlists. PSpice A/D extracts the definitions for all parts modeled as subcircuits, viewing parts as a collection of primitive parts and node connections.

The digital primitives that make up a digital part determine the way that PSpice A/D processes an analog/digital interface to that part. Specifically, the I/O model for each digital primitive connected at the interface gives PSpice A/D the necessary information.

PSpice A/D recognizes three types of nodes: analog nodes, digital nodes, and interface nodes. The node type is determined by the types of parts connected to it. If all of the parts connected to a node are analog, then it is an analog node. If all of the parts are digital, then it is a digital node. If there is a combination of analog and digital parts, then it is an interface node.

PSpice A/D automatically breaks interface nodes into one purely analog and one or more digital nodes by inserting one or more analog/digital interface subcircuits.

PSpice A/D also automatically connects a power supply to the interface subcircuit to complete the generation of the interface.

To view simulation results at an analog/digital interface in your schematic using the graphical waveform analyzer:

- Place a marker on the appropriate interface net. The additional nodes created by PSpice A/D remain transparent.
- View results in PSpice A/D by selecting traces from the output variable list (from the Trace menu, choose Add Trace). If you use this approach, note the names PSpice A/D generates for the new nodes.

@ 2023 All Rights Reserved.

Interface Subcircuit Selection by PSpice A/D

Analog-to-digital (AtoD) and digital-to-analog (DtoA) interface subcircuits handle the translation between analog voltages/impedances and digital states, or vice-versa. The main component of an interface subcircuit is either a PSpice N part (digital input: digital-to-analog) or a PSpice O (that's the letter O, not the numeral zero) part (digital output: analog-to-digital).

PSpice N and O parts are neatly packaged into interface subcircuits in the model library. The standard model library shipped with your software installation includes interface subcircuits for each of the supported logic families: TTL, CD4000 series CMOS and high-speed CMOS (HC/HCT), ECL 10K, and ECL 100K. This frees you from ever having to define them yourself when using parts in the standard library. If you are creating custom digital parts in technologies other than those provided in the standard model library, you may need to create your own interface subcircuits.

Every digital primitive comprising the subcircuit description of a digital part has an I/O model describing its loading and driving characteristics. The name of the interface subcircuit actually inserted by PSpice A/D is specified by the I/O model of the digital primitive at the interface. The I/O model has parameters for up to four analog-to-digital (AtoD) and four digital-to-analog (DtoA) subcircuit names.

You can choose among four interface levels of subcircuit models, depending on the simulation accuracy you need. In some cases you may need more accurate simulations of the input/output stages of a digital part, while in other cases, a simpler, smaller model is enough.

Digital parts provided in the standard libraries only use interface levels 1 and 2. With the exception of the HC/HCT series, levels 3 and 4 reference the same subcircuits as levels 1 and 2. Table 15 summarizes the four interface levels.

The difference between levels 1 and 2 only occurs in the AtoD interfaces. In all cases, the level 1 DtoA interface is the same as the level 2 DtoA interface, except that the level 2 DtoA interface does not generate intermediate R, F, and X levels.

Table 15-1 Interface subcircuit models

Level	Subcircuits	Definition
1	AtoD1/DtoA1	AtoD generates intermediate R, F, and X levels
2	AtoD2/DtoA2	AtoD does not generate intermediate R, F, and X levels
3	AtoD3/DtoA3	(same as level 1)

@ 2023 All Rights Reserved.

Table 15-1 Interface subcircuit models

Level	Subcircuits	Definition
4	AtoD4/DtoA4	(same as level 2)

The OrCAD X libraries provide two different DtoA models in the HC/HCT series: the *simple* model and the *elaborate* model. You can use the simple model by specifying level 1 or 2, the elaborate model by specifying level 3 or 4. The elaborate model is noticeably slower than the simple model, so you should only use it if you are using a power supply level other than 5.0 volts.

The HC/HCT level 1 and 2 DtoA models produce accurate I-V curves given a fixed power supply of 5.0 volts and a temperature of 25°C. The level 3 and 4 DtoA models produce accurate I-V curves over the acceptable range of power supply voltages (2-6 volts), and they include temperature derating.

Level 1 interface

The level 1 AtoD interface generates intermediate logic levels (R, F, X) between the voltage ranges VILMAX and VIHMIN (specific voltages depend on the technology you are using). A steadily rising voltage on the input of the AtoD will transition from 0 to R at VILMAX and from R to 1 at VIHMIN. The F level is output for steadily falling voltages in a similar manner. The X level is produced if the input voltage starts in the threshold region or doubles back into a previously crossed threshold.

Level 1 (the default) strictly maps logic levels onto the changing input voltage. The exact switching voltage is assumed to be anywhere between VILMAX and VIHMIN due to temperature or power supply variations. Thus, it provides more accurate, less optimistic results.

This behavior may not be appropriate when the input rise and fall times are long, or when the input voltage never leaves the threshold region. If this is the case, you may want to use the level 2 interface.

Level 2 interface

The level 2 AtoD interface transitions directly from 0 to 1 and 1 to 0 without passing through intermediate R, F, or X levels. An exact switching voltage is assumed (again, the specific voltage depends on the technology you are using). It provides a more optimistic, and therefore less accurate, response than level 1. Level 2's behavior is appropriate when the input voltage oscillates around the threshold voltage.

@ 2023 All Rights Reserved.

Note: You can avoid simulations that get bogged down with the greater detail of R, F, and X states around these oscillations. You may want to specify level 2 on only those parts for which this behavior is critical to a successful simulation.

@ 2023 All Rights Reserved.

Setting a Default A/D Interface

For mixed-signal simulation, you can select the AtoD and DtoA interface level circuit-wide and on individual part instances.

- To select the default interface level circuit-wide, select one of the four Default A/D interfaces in the Simulation Settings dialog box by selecting the Gate-level simulation category under the Options tab. Part instances with the IO_LEVEL property set to 0 use this value.
- You can override the circuit-wide default on an individual part by specifying an IO_LEVEL property from 1 to 4, where:

  1: AtoD1 and DtoA1 (default)

  2: AtoD2 and DtoA2

  3: AtoD3 and DtoA3

  4: AtoD4 and DtoA4

For example, you can tell the simulator to use the level 2 interface subcircuits for a 7400 part by setting the IO_LEVEL property to 2. All other part instances continue to use the circuit-wide setting. By default, IO_LEVEL is set to 0, which tells the simulator to use the circuit-wide level defined in the Gate-level simulation category in the Simulation Settings dialog box.

@ 2023 All Rights Reserved.

Specifying Digital Power Supplies

Digital power supplies are used to power interface subcircuits that are automatically created by PSpice A/D when simulating analog/digital interfaces. They are specified as follows:

- PSpice A/D can instantiate them automatically.
- You can create your own digital power supplies and place them in your design.

When using parts from the standard libraries in your design, you can usually have PSpice A/D automatically create the necessary digital power supply. If you use custom digital parts created in technologies other than those provided in the standard model library, you may need to create your own digital power supplies.

Because digital power supplies are used only by analog/digital interface subcircuits, digital power supplies are not needed for digital-only designs. We recommend avoiding placing a power supply to a digital-only design because it may increase simulation time and memory usage.

Default Power Supply Selection by PSpice A/D

When PSpice A/D encounters an analog/digital interface, it creates the appropriate interface subcircuit and power supply according to the I/O model referenced by the digital part. The I/O model is specific to the digital part's logic family. The power supply provides reference or drive voltage for the analog side of the interface.

By default, PSpice A/D inserts one power supply subcircuit for every logic family in which a digital primitive is involved with an analog/digital interface. These power supply subcircuits create the digital power and ground nodes that are the defaults for all parts in that family. If multiple digital primitives from the same logic family are involved with analog/digital interfaces, one instance of the power supply subcircuit is created with all primitives connected to the power supply nodes.

Table 15-2 summarizes the default node names and values. For instance, TTL power supplies have a default value of 5.0 volts at analog/digital interfaces.

@ 2023 All Rights Reserved.

Table 15-2 Default digital power/ground pin connections

Logic family	Digital power/ ground pin properties	Default digital power/ground nodes
TTL	PSPICEDEFAULTNET (PWR)	$G_DPWR (5.0 volts) $G_DGND (0 volts)
	PSPICEDEFAULTNET (GND)	
CD4000	PSPICEDEFAULTNET (VDD)	$G_CD4000_VDD (5 volts) $G_CD4000_VSS (0 volts)
	PSPICEDEFAULTNET (VSS)	
ECL 10K	PSPICEDEFAULTNET (VEE)	$G_ECL_10K_VEE (-5.2 volts) $G_ECL_10K_VCC1 (0 volts) $G_ECL_10K_VCC2 (0 volts)
	PSPICEDEFAULTNET (VCC1)	
	PSPICEDEFAULTNET (VCC2)	
ECL 100K	PSPICEDEFAULTNET (VEE)	$G_ECL_100K_VEE (-4.5 volts) $G_ECL_100K_VCC1 (0 volts) $G_ECL_100K_VCC2 (0 volts)
	PSPICEDEFAULTNET (VCC1)	
	PSPICEDEFAULTNET (VCC2)	

The PSPICEDEFAULTNET pin properties have the same default values as the digital power and ground nodes created by the default power supply. These node assignments are passed from the part instance to the digital primitives describing its behavior, connecting any digital primitive affected by an analog connection to the correct power supply.

The default I/O models and power supply subcircuits are found in DIG_IO.LIB. The four default power supplies provided in the model library are DIGIFPWR (TTL), CD4000_PWR (CD4000 series CMOS), ECL_10K_PWR (ECL 10K), and ECL_100K_PWR (ECL 100K).

@ 2023 All Rights Reserved.

Creating Custom Digital Power Supplies

Each digital part model has optional digital power and ground nodes that you can use to specify custom power supplies. To do this, use one of the digital power supplies listed in Table 15-3 in your design and redefine the digital power supply nodes.

Table 15-3 Digital power supply parts in SPECIAL.OLB

Part type (PSpice A/D X model)	Part name
CD4000 power supply	CD4000_PWR
TTL power supply	DIGIFPWR
ECL 10K power supply	ECL_10K_PWR
ECL 100K power supply	ECL_100K_PWR

When creating custom power supplies, you can refer to the power supply definitions in DIG_IO.LIB for examples of power supply subcircuit definitions. The properties relevant to creating custom power supplies are shown in Table 15-4.

Table 15-4 Digital power supply properties

Part name	Property	Description
CD4000_PWR	VOLTAGE	CD4000 series CMOS power supply voltage
	PSPICEDEFAULTNET	CD4000 series CMOS hidden power supply pins for VDD and VSS
DIGIFPWR	VOLTAGE	TTL power supply voltage
	PSPICEDEFAULTNET	TTL hidden power (PWR) and ground (GND) pins

@ 2023 All Rights Reserved.

Part name	Property	Description
ECL_10K_PWR ECL_100K_PWR	VEE VCC1 VCC2	ECL power supply voltages
	PSPICEDEFAULTNET	ECL hidden power supply pins for VEE, VCC1 and VCC2

To create a custom digital power supply

Note: This procedure applies to all logic families.

1. Place the appropriate power supply part listed in Table 15-3 in your design (by logic family).
2. Rename the power supply power and ground pins (PSPICEDEFAULTNET properties).
3. Reset the power supply power and ground voltages as required.
4. For any digital part instance that uses the power supply, set its appropriate PSPICEDEFAULTNET pin properties to the power and ground pins created by the secondary power supply.

Overriding CD4000 power supply voltage throughout a design

Designs using CD4000 parts often require power supply voltages other than the default 5.0 volts supplied by the standard CD4000_PWR power supply part. If needed, you can override the power supply voltage for all CD4000 parts in a design.

The default power supply nodes used by CD4000 parts are named $G_CD4000_VDD and $G_CD4000_VSS as created by the power supply subcircuit CD4000_PWR. This supply defaults to 5.0 volts. You can override the voltage across these two nodes by defining values for the parameters named CD4000_VDD and CD4000_VSS that are referenced by the CD4000_PWR subcircuit definition.

To change the CD4000_PWR power supply to 12 volts, referenced to ground:

1. Place an instance of the PARAM pseudo-part from `SPECIAL.OLB`.
2. Create a new PARAM property as follows:

```
CD4000_VDD = 12.0V
```

DC4000_VSS is left at its default of 0 volts.

@ 2023 All Rights Reserved.

If the reference voltage also needs to be reset, the same method can be used to define the CD4000_VSS parameter by setting this property of the same PARAM instance. For example, if you want the supplies to go between -5 volts and +5 volts (a difference of 10 volts), set CD4000_VSS to -5V and CD4000_VDD to +10V; as a result, CD4000_VDD is 10 volts above CD4000_VSS, or +5 volts.

Creating a secondary CD4000, TTL, or ECL power supply

Designs using CD4000, TTL, or ECL parts may require power supply voltages in addition to the default 5.0 volts supplied by the standard CD4000_PWR power supply part.

To create a secondary power supply for any one of the CD4000, TTL, or ECL technologies, you must place the appropriate power supply part and create user-defined nodes with a new voltage value.

Note: Designs with TTL and ECL parts rarely require secondary power supplies. If needed, however, you can use this procedure to add a secondary power supply for TTL and ECL parts.

To create and use a secondary CD4000 power supply with nodes MY_VDD and MY_VSS and a voltage of 3.5 volts:

1. Place the CD4000_PWR power supply and modify the appropriate pin properties as follows:

```
VOLTAGE = 3.5V
PSPICEDEFAULTNET = MY_VDD
PSPICEDEFAULTNET = MY_VSS
```

2. Select a CD4000 part in the schematic to which the new power supply should apply, then change the appropriate pin properties as follows:

```
PSPICEDEFAULTNET = MY_VDD
PSPICEDEFAULTNET = MY_VSS
```


@ 2023 All Rights Reserved.

Interface Generation and Node Names

The majority of the interface generation process involves PSpice A/D determining whether analog and digital primitives are connected, and if so, inserting an interface subcircuit for each digital connection. This turns the interface node into a purely analog node, which now connects to the analog terminal of the interface subcircuit. To complete the original connection, PSpice A/D creates a new digital node between the digital terminal of the interface subcircuit and the digital primitive.

Because PSpice A/D must create new digital nodes, it must give them unique names. These node names are used in the output variables in the list of viewable traces when you choose Add Trace from the Trace menu. Name generation follows these rules:

- The analog node retains the name of the original interface node—either the labeled wire name in the design, or the node name automatically generated for an unlabeled wire.
- Each new digital node name consists of the labeled wire name in the design or the node name automatically generated for an unlabeled wire, appended with $AtoD or $DtoA. If the node is attached to more than one digital part, the second digital node is appended with $AtoD2 or $DtoA2, and so on.

The following figure shows a fragment of a mixed analog/digital circuit before and after the interface subcircuits have been added. The wires labeled 1 and 2 in the schematic representation are the interface nets connecting analog and digital parts. These translate to

@ 2023 All Rights Reserved.

interface nodes, which are processed by PSpice A/D to create the circuit fragment shown in the PSpice A/D representation.

Figure 15-1 Mixed analog/digital circuit before and after interface generation.

After interface generation, node 1 is a purely analog node, connecting the resistor, transistor, and the analog inputs of both AtoD subcircuits. Node 2 is also a purely analog node, connecting the resistor and the analog output of the DtoA interface. You can see that PSpice A/D inserted two new digital nodes, 1$AtoD and 1$AtoD2, which connect the outputs of the AtoD interfaces to the inverter inputs. It also created one digital node, 2$DtoA, to connect the output of U1 to the digital input of the DtoA interface.

The interface subcircuits PSpice A/D automatically generates are listed in the simulation output file under the section named Generated AtoD and DtoA Interfaces.

The lines that begin with “Moving…from analog node” indicate the new digital node names that were generated. Each of these are the interface subcircuit calls inserted by PSpice A/D.

In this example, the subcircuits named AtoD_STD and DtoA_STD are obtained from the I/O model that is referenced by the inverter primitive inside the subcircuit describing the 7404 part. The CAPACITANCE, DRVL (low-level driving resistance), and DRVH (high-level driving resistance) subcircuit parameter values come from the same I/O model.

@ 2023 All Rights Reserved.

```
**** Generated AtoD and DtoA Interfaces ****
*
* Analog/Digital interface for node 1
*
* Moving X1.U1:.A from analog node 1 to new digital node * 1$AtoD
X$1_AtoD1 1 1$AtoD $G_DPWR $G_DGND AtoD_STD
+     PARAMS: CAPACITANCE=   0
* Moving X2.U1:.A from analog node 1 to new digital node * 1$AtoD2
X$1_AtoD2 1 1$AtoD $G_DPWR $G_DGND AtoD_STD
+     PARAMS: CAPACITANCE=   0
*
* Analog/Digital interface for node 2
*
** Moving X1.U1.Y from analog node 2 to new digital node * 2$DtoA
X$2_DtoA1 2$DtoA 2 $G_DPWR $G_DGND DtoA_STD
+     PARAMS: DRVL=0 DRVH=0 CAPACITANCE=0
*
* Analog/Digital interface power supply subcircuit
*
X$DIGIFPWR 0 DIGIFPWR

.END ;(end of AtoD and DtoA interfaces)
```

Figure 15-2 Simulation output for mixed analog/digital circuit.

Following is the output file of the mixed analog/digital circuit:

After the interface subcircuit calls, PSpice A/D inserts one or more interface power supply subcircuits. The subcircuit name is specified in the I/O model for the digital primitive at the interface. In this example, PSpice A/D inserted DIGIFPWR, which is the power supply subcircuit used by all TTL models in the model library. DIGIFPWR creates the global nodes $G_DPWR and $G_DGND, which are the default nodes used by each TTL part.

@ 2023 All Rights Reserved.

@ 2023 All Rights Reserved.

16

Digital Worst-case Timing Analysis

Manufacturers of electronic components generally specify component parameters (such as propagation delays in the case of logic devices) as having tolerances. These are expressed as either an operating range, or as a spread around a typical operating point. The designer then has some indication of how much deviation from typical one might expect for any of these particular component delay values.

Realizing that any two (or more) instances of a particular type of component may have propagation delay values anywhere within the published range, designers are faced with the problem of ensuring that their products are fully functional when they are built with combinations of components having delay specifications that fall (perhaps randomly) anywhere within this range.

Historically, this has been done by making simulation runs using minimum (MIN), typical (TYP), and maximum (MAX) delays, and verifying that the product design is functional at these extremes. But, while this is useful to some extent, it does not uncover circuit design problems that occur only with certain combinations of slow and fast parts. True digital worst-case simulation, as provided by PSpice A/D, does just that.

Other tools called timing verifiers are sometimes used in the design process to identify problems that are indigenous to circuit definition. They yield analyses that are inherently pattern-independent and often pessimistic in that they tend to find more problems than will truly exist. In fact, they do not consider the actual usage of the circuit under an applied stimulus.

PSpice A/D does not provide this type of static timing verification. Digital worst-case timing simulation, as provided by PSpice A/D, is a pattern-dependent mechanism that allows a designer to locate timing problems subject to the constraints of a specific applied stimulus.

Digital Worst-case Analysis compared to Analog Worst-case Analysis

Digital worst-case timing simulation is different from analog worst-case analysis in several ways. Analog worst-case analysis is implemented as a sensitivity analysis for each parameter

@ 2023 All Rights Reserved.

which has a tolerance, followed by a projected worst-case simulation with each parameter set to its minimum or maximum value. This type of analysis is general since any type of variation caused by any type of parameter tolerance can be studied. But it is time consuming since a separate simulation is required for each parameter. This does not always produce true worst-case results, since the algorithm assumes that the sensitivity is monotonic over the tolerance range.

The techniques used for digital worst-case timing simulation are not compatible with analog worst-case analysis. It is therefore not possible to do combined analog/digital worst-case analysis and simulation and get the correct results. PSpice A/D allows digital worst-case simulation of mixed-signal and all-digital circuits; any analog sections are simulated with nominal values.

Systems containing embedded analog-within-digital sections do not give accurate worst-case results; they may be optimistic or pessimistic. This is because analog simulation can not model a signal that will change voltage at an unknown point within some time interval.

@ 2023 All Rights Reserved.

Starting Digital Worst-case Timing Analysis

To set up a digital worst-case timing analysis, do the following:

1. In the *Simulation Settings* dialog box, click the *Options* tab.
2. Select the *General* node in the *Gate-level Simulation* tree Structure of the *Options* tab.
3. In the Timing Mode frame, check Worst-case (min/max)
4. In the Initialize all flip-flops drop-down list, select X.
5. Set the Default I/O level for A/D interfaces to 1.
6. Click *OK*.

7. Start the simulation.

Simulator Representation of Timing Ambiguity

PSpice A/D uses the five-valued state representation {0,1,R,F,X}, where R and F represent rising and falling transitions, respectively. Any R or F transitions can be thought of as ambiguity regions. Although the starting and final states are known (Example: R is a 0 → 1 transition), the exact time of the transition is not known, except to say that it occurs somewhere within the *ambiguity region*. The ambiguity region is the time interval between the earliest and the latest time that a transition could occur.

@ 2023 All Rights Reserved.

Timing ambiguities propagate through digital devices via whatever paths are sensitized to the specific transitions involved. This is normal logic behavior. The delay values (MIN, TYP, or MAX) skew the propagation of such signals by whatever amount of propagation delay is associated with each primitive instance.

When worst-case (MIN/MAX) timing operation is selected, both the MIN and the MAX delay values are used to compute the duration of the timing ambiguity result that represents a primitive's output change.

For example, consider the model of a BUF device in the following figure.

```
U5 BUF $G_DPWR $G_DGND IN1 OUT1 ; BUFFER model
+ T_BUF IO_STD

.MODEL T_BUF UGATE (            ; BUF timing model
+ TPLHMN=15ns TPLHTY=25ns TPLHMX=40ns
+ TPHLMN=12ns TPHLTY=20ns TPHLMX=35ns)
```

Figure 16-1 Timing ambiguity example one.

The application of the instantaneous 0-1 transition at 5nsec in this example produces a corresponding output result. Given the delay specifications in the timing model, the output edge occurs at a MIN of 15nsec later and a MAX of 40nsec later. The region of ambiguity for the output response is from 20 to 45nsec (from TPLHMN and TPLHMX values). Similar calculations apply to a 1-0 transition at the input, using TPHLMN and TPHLMX values.

@ 2023 All Rights Reserved.

Propagation of Timing Ambiguity

As signals propagate through the circuit, ambiguity is contributed by each primitive having a nonzero MIN/MAX delay spread. Consider the following example that uses the delay values of the previous BUF model.

Figure 16-2 Timing ambiguity example two.

This accumulation of ambiguity may have adverse effects on proper circuit operation. In the following example, consider ambiguity on the data input to a flip-flop.

Figure 16-3 Timing ambiguity example three.

The simulator must predict an X output, because it is not known with any certainty when the data input actually made the 0-1 transition. If the cumulative ambiguity present in the data signal had been less, the 1 state would be latched up correctly.

The following figure illustrates the case of unambiguous data change (settled before the clock could transition) being latched up by a clock signal with some ambiguity. The Q output will change, but the time of its transition is a function of both the clock's ambiguity and that contributed by the flip-flop MIN/MAX delays.

Figure 16-4 Timing ambiguity example four.

Identification of Timing Hazards

Timing hazard is the term applied to situations where the response of a device cannot be properly predicted because of uncertainty in the arrival times of signals applied to its inputs.

@ 2023 All Rights Reserved.

For example, the following figure shows the following signal transitions (0-1, 1-0) being applied to the AND gate.

Figure 16-5 Timing hazard example.

The state of the output does not (and should not) change, since at no time do both input states qualify the gate, and the arrival times of the transitions are known.

Convergence Hazard

In cases where there are ambiguities associated with the signal transitions 0-R-1 and 1-F-0—which have a certain amount of overlap—it is no longer certain which of the transitions happens first.

The output could pulse (0-1-0) at some point because the input states may qualify the gate. On the other hand, the output could remain stable at the 0 state. This is called a *convergence hazard* because the reason for the glitch occurrence is the convergence of the conflicting ambiguities at two primitive inputs.

Gate primitives (including LOGICEXP primitives) that are presented with simultaneous opposing R and F levels may produce a pulse of the form 0-R-0 or 1-F-1.

For example, a two-input AND gate with the inputs produces the following output:

Figure 16-6 Convergence hazard example.

This output (0-R-0) should be interpreted as *a possible single pulse, no longer than the duration of the R level*.

Note: Other types of primitives, such as flip-flops, may produce an X instead of an R-0 or F-1 in response to a convergence hazard.

The actual device's output may or may not change, depending on the transition times of the inputs.

@ 2023 All Rights Reserved.

Critical Hazard

It is important to note that the glitch predicted could propagate through the circuit and may cause incorrect operation. If the glitch from a timing hazard becomes latched up in an internal state (such as flip-flop or ram), or if it causes an incorrect state to be latched up, it is called a *critical hazard* because it definitely causes incorrect operation.

Otherwise, the hazard may pose no problem. The following figure shows the same case as above, driving the data input to a latch:

Figure 16-7 Critical hazard example.

As long as the glitch always occurs well before the leading edge of the clock input, it will not cause a problem.

@ 2023 All Rights Reserved.

Cumulative Ambiguity Hazard

In worst-case mode, simple signal propagation through the network will result in a buildup of ambiguity along the paths between synchronization points. See Glitch Suppression due to Inertial Delay. The cumulative ambiguity is illustrated in the following figure.

Figure 16-8 Cumulative ambiguity hazard example one.

The rising and falling transitions applied to the input of the buffer have a 1nsec ambiguity. The delay specifications of the buffer indicate that an additional 2nsec of ambiguity is added to each edge as they propagate through the device. Notice that the duration of the stable state 1 has diminished due to the accumulation of ambiguity.

The following figure shows the effects of additional cumulative ambiguity:

Figure 16-9 Cumulative ambiguity hazard example two.

The X result is predicted here because the ambiguity of the rising edge propagating through the device has increased to the point where it will overlap the later falling edge ambiguity. Specifically, the rising edge should occur between 3nsec and 12nsec; but, the subsequent falling edge applied to the input predicts that the output starts to fall at 10nsec. This situation is called a *cumulative ambiguity hazard*.

Another cause of cumulative ambiguity hazard involves circuits with asynchronous feedback. The simulation of such circuits under worst-case timing constraints yields an overly pessimistic result due to the unbounded accumulation of ambiguity in the feedback path. A simple example of this effect is shown in the following figure:

Figure 16-10 Cumulative ambiguity hazard example three.

@ 2023 All Rights Reserved.

Due to the accumulation of ambiguity in the loop, the output signal will eventually become X, because the ambiguities of the rising and falling edges overlap. However, in the hardware implementation of this circuit, a continuous phase shift with respect to absolute time is what will actually occur (assuming normal deviations of the rise and fall delays from the nominal values).

Note: If this signal were used to clock another circuit, it would become the reference and the effects of the phase shift could be ignored. You can do this by setting the NAND gate's model parameter, MNTYMXDLY=2 to utilize typical delay values for that one gate only (all other devices continue to operate in worst-case mode).

@ 2023 All Rights Reserved.

Reconvergence Hazard

PSpice A/D recognizes situations where signals having a common origin reconverge on the inputs of a single device. In the following figure, the relative timing relationship between the two paths (U2, U3) is important.

Figure 16-11 Reconvergence hazard example one.

Given the delay values shown, it is impossible for the clock to change before the data input, since the MAX delay of the U2 path is smaller than the MIN delay of the U3 path. In other words, the overlap of the two ambiguity regions could not actually occur.

PSpice A/D recognizes this type of situation and does not produce the overly pessimistic result of latching an X state into the Q-output of U4. This factors out the 15 nsec of common ambiguity attributed to U1 from the U2 and U3 signals.

Figure 16-12 Reconvergence hazard example two.

The result in the following figure does not represent what is actually propagated at U2 and U3, but is a computation to determine that U2 must be stable at the earliest time U3 might change. This is why an X level should not be latched.

In the event that discounting the common ambiguity does not preclude latching the X (or, in the case of simple gates, predicting a glitch), the situation is called a *reconvergence hazard*. This is the same as a convergence hazard with the conflicting signal ambiguities having a common origin.

@ 2023 All Rights Reserved.

To use digital worst-case simulation effectively, find the areas of the circuit where signal timing is most critical and use constraint checkers where appropriate. These devices identify specific timing violations, taking into account the actual signal ambiguities (resulting from the elements' MIN/MAX delay characteristics). See the *PSpice A/D Reference Guide* for more information about digital primitives.

The most common areas of concern include:

- data/clock signal relationships
- clock pulse-widths
- bus arbitration timing

Signal ambiguities that converge (or reconverge) on wired nets or buses with multiple drivers may also produce hazards in a manner similar to the behavior of logic gates. In such cases, PSpice A/D factors out any common ambiguity before reporting the existence of a hazard condition.

The use of constraint checkers to validate signal behavior and interaction in these areas of your design identifies timing problems early in the design process. Otherwise, a timing-related failure is only identifiable when the circuit does not produce the expected simulation results. See Timing Simulation Methodology for information on digital worst-case timing simulation methodology.

@ 2023 All Rights Reserved.

Glitch Suppression due to Inertial Delay

Signal propagation through digital primitives is performed by the simulator subject to constraints such as the primitive's function, delay parameter values, and the frequency of the applied stimulus. These constraints are applied both in the context of a normal, well-behaved stimulus, and a stimulus that represents timing hazards.

Timing hazards may not necessarily result in the prediction of an X or glitch output from a primitive; these are due to the delay characteristics of the primitive, which PSpice A/D models using the concept of inertial delay.

A device presented with a combination of rising and falling input transitions (assuming no other dominant inputs) produces a glitch due to the uncertainty of the arrival times of the transitions.

Figure 16-13 Glitch suppression example one.

However, when the duration of the conflicting input stimulus is less than the inertial delay of the device, the X result is automatically suppressed by the simulator because it would be overly pessimistic.

Figure 16-14 Glitch suppression example two.

In the analysis of reconvergent fanout cases (where common ambiguity is recognized), it is possible that conflicting signal ambiguities may still overlap at the inputs to a primitive, even

@ 2023 All Rights Reserved.

after factoring out the commonality. In such cases, where the amount of overlap is less than the inertial delay of the device, the prediction of a glitch is also suppressed by the simulator.

Figure 16-15 Glitch suppression example three.

In this case, factoring out the 15nsec common ambiguity still results in a 5nsec overlap of conflicting states. The glitch is suppressed, however, because 5nsec is less than TPLHMX-TPLHMN (the computed inertial delay value of the AND gate, 6nsec).

Note: Glitch suppression can be overridden by setting the pulse-width rejection threshold parameter (TPWRT) in the device's I/O Model.

Timing Simulation Methodology

Note: This is not intended to be a comprehensive discussion of the application of digital worst-case timing simulation in the design process. Rather, it is a suggested starting point for understanding the results of your simulation.

Combining component tolerances and the circuit design's functional response to a specific stimulus presents a challenge. You must make sure that all the finished circuits will operate properly. Well-designed systems have a high degree of immunity from the effects of varying combinations of individual component tolerances.

Digital worst-case timing simulation can help identify design problems, depending upon the nature of the stimulus applied to the design. You can use the simulation of signal propagation through the network to observe the timing relationships among various devices and make adjustments to the design.

Digital worst-case timing simulation does not yield such results without an applied stimulus; it is not a static timing analysis tool. The level of confidence that you establish for your design's timing-dependent characteristics is directly a function of the applied stimulus.

Generally, the most productive way to define a stimulus is to use functional testing: a stimulus designed to operate the design in a normal manner, exercising all of the important features in combination with a practical set of data. For example, if you were designing a digital ADDER

@ 2023 All Rights Reserved.

circuit, you would probably want to ensure that no timing race conditions existed in the carry logic.

Your timing simulation methodology should include these key steps:

- Accurate specification of device delay characteristics.
- Functional specification of circuit behavior, including all “don’t care” states or conditions.
- A set of stimuli designed to verify the operation of all functions of the design.

One common design verification strategy is stepwise identification of the sections of the design that are to be exercised by particular subsets of the stimulus, followed by verification of the response against the functional specification.

Complete this phase using normal (not digital worst-case) simulation, with typical delays selected for the elements. The crucial metric here is the state response of the design. Note that (with rare exception) this response consists of defined states and does not include X’s.

The second phase of design verification is to use digital worst-case simulation, reapplying the functionally correct stimulus, and comparing the resulting state response to that obtained during normal simulation. For example, in the case of a convergence or reconvergence hazard, look for conflicting rise/fall inputs. In the case of cumulative ambiguity, look for successive ambiguity regions merging within two edges forming a pulse. Investigate differences at primary observation points (such as circuit outputs and internal state variables)—particularly those due to X states (such as critical hazards)—to determine their cause.

Starting at those points, use the waveform analyzer and the circuit schematic to trace back through the network. Continue until you find the reason for the hazard.

After you identify the appropriate paths and know the relative timing of the paths, you can do either of the following:

- Modify the stimulus (in the case of a simple convergence hazard) to rearrange the relative timing of the signals involved.

  Note: Modifying the stimulus is not generally effective for reconvergent hazards, because the problem is between the source of the reconvergent fanout and the location of the hazard. In this case, discounting the common ambiguity did not preclude the hazard.

- Change one or both of the path delays to rearrange the relative timing, by adding or removing logic, or by substituting component types with components that have different delay characteristics.

@ 2023 All Rights Reserved.

In the case of the cumulative ambiguity hazard, the most likely solution is to shorten the path involved. You can do this in either of two ways:

- Add a synchronization point to the logic, such as a flip-flop—or gating the questionable signal with a clock (having well-controlled ambiguity)—before its ambiguity can grow to unmanageable duration.
- Substitute faster components in the path, so that the buildup of ambiguity happens more slowly.

@ 2023 All Rights Reserved.

@ 2023 All Rights Reserved.

17

Analyzing Waveforms with PSpice

You can use the waveform analysis features of PSpice A/D to visually analyze and interactively manipulate the waveforms generated from the simulation data.

PSpice A/D uses high-resolution graphics so you can view the results of a simulation both on the screen and in printed form. On the screen, waveforms appear as plots displayed in Probe windows within the PSpice A/D workspace.

In effect, waveform analysis is a software oscilloscope. Performing a PSpice simulation corresponds to building or changing a breadboard, and performing waveform analysis corresponds to looking at the breadboard with an oscilloscope.

With waveform analysis you can:

- View simulation results in multiple Probe windows
- Compare simulation results from multiple circuit designs in a single Probe window
- Display simple voltages, currents, and noise data
- Display complex arithmetic expressions that use the basic measurements
- Display Fourier transforms of voltages and currents, or of arithmetic expressions involving voltages and currents
- For mixed analog/digital simulations, display analog and digital waveforms simultaneously with a common time base
- Add text labels and other annotation symbols for clarification

PSpice A/D generates two forms of output: the simulation output file and the waveform data file. The calculations and results reported in the simulation output file act as an audit trail of the simulation. However, the graphical analysis of information in the waveform data file is the most informative and flexible method for evaluating simulation results.

@ 2023 All Rights Reserved.

Elements of a Plot

A single plot consists of the analog (lower) area and the digital (upper) area.

Figure 17-1 Analog and digital areas of a plot.

You can display multiple plots simultaneously. If you display only analog waveforms, the entire plot will be an analog area. Likewise, if you display only digital waveforms, the entire plot will be a digital area.

Elements of a Probe Window

A Probe window is a separately managed waveform display area. A Probe window can include multiple analog and digital plots.

@ 2023 All Rights Reserved.

Because a Probe window is a window object, you can minimize and maximize windows, or move and scale the windows, within the PSpice A/D workspace. The toolbar in the Probe window applies to the active window. The following figure shows two plots displayed together:

Tip

From the View menu, choose Toolbar to display or hide the toolbar.

Figure 17-2 Two Probe windows.

You can display information from one or more waveform data files in one Probe window. After the first file is loaded, you can load other files into the same Probe window by appending them in PSpice A/D (using the Append Waveform command under the File menu).

Managing Multiple Probe Windows

You can open any number of Probe windows. Each Probe window is a tab on the worksheet displayed in the middle of the workspace.

The same waveform data file can be displayed in more than one Probe window. You can tile the windows to compare data.

Only one Probe window is active at any given time. It is identified by a highlighted title bar or a topmost tab. Menu, keyboard, and mouse operations affect only the active Probe window. You can switch to another Probe window by clicking another tab or title bar.

@ 2023 All Rights Reserved.

Printing multiple windows

You can print all or selected Probe windows, with up to nine windows on a single page. When you choose Print from the File menu, a list of all open Probe windows appears. Each Probe window is identified by the unique identifier in parentheses in its title bar.

The arrangement of Probe windows on the page can be customized using the Page Setup dialog box. You can print in either portrait (vertical) or landscape (horizontal) orientation. You can also use Print Preview to view all of the Probe windows as they will appear when printed.

Toggling between display modes

You can choose from two different display modes in PSpice A/D: default and alternate.

The default display mode in PSpice A/D includes the main Probe window, plus the output window and the simulation status window. This provides all possible information about the simulation run and contains all of the toolbars and settings.

The alternate display mode shows only the Probe window with any waveforms that have been plotted. This mode gives you only the plots you are interested in seeing without the additional simulation data normally provided by PSpice A/D. By default, the alternate display mode is set to be visible at all times.

The toolbar and window settings are saved for each mode. Any changes you make in the settings will become the new default the next time you choose that display mode.

The alternate display mode can be very handy when you want to see the waveforms superimposed on the schematic diagram for easy debugging and testing of the circuit. You can customize the alternate display mode to view various toolbars or other PSpice A/D windows, according to your own preferences.

To toggle between the standard and the alternate display modes

➡ From the *View* menu, choose *Alternate Display* or click the *Alternate Display* toolbar button.

Keeping the Probe window visible at all times

Like any other application running under Windows, the PSpice A/D window will remain in the forefront of the desktop only as long as it is the active window. In order to keep the PSpice A/D window visible at all times, you can use the push pin feature.

By keeping the Probe window on top of other active windows, you can easily view the schematic page at the same time you see the corresponding waveform for that circuit. This

@ 2023 All Rights Reserved.

allows you to cross-probe quickly and easily without having to activate the Probe window each time.

Note: The push pin button is a toggle: clicking on it when it is enabled will disable the “on top” function.

➡ To make the Probe window visible at all times, click the push pin button in the toolbar.

@ 2023 All Rights Reserved.

Setting up Waveform Analysis

You can do the following settings:

- You can set the theme of the PSpice A/D application using the *Probe Settings* dialog box.
- You can configure the probe display using the *Probe Settings* dialog box or by editing the `PSpice.ini` file.
- You can configure the print colors only using the `PSpice.ini` file.

To set up application theme, do the following:

1. Choose *Tools – Options*.

   The *Probe Settings* dialog box opens.
2. Click the *Color Settings* tab.
3. Select `Dark` or `Light` option from the *Application Theme* drop-down list.
4. Click *OK*.

Note: The standard Windows operating system dialog boxes do not follow the application theme when launched from PSpice A/D.

To edit display colors in the *Probe Settings* dialog box, do the following:

1. Choose *Tools – Options* to open the Probe Settings dialog box.
2. Click the *Color Settings* tab.
3. Select a background color from the *Background* list.
4. Select a foreground color for the graph lines and texts from the *Foreground* list.
5. Select the colors from the *Available Colors* list and click *Add* to add them to the *Trace Colors* list.
6. Select a color in the *Trace Colors* list

   a. Click *Up* or *Down* to specify the order of the color.

   b. Click *Remove* to remove a color from the *Trace Colors* list.

   The colors will be applied to traces in the sequence they are displayed.
7. Click *OK*.

@ 2023 All Rights Reserved.

Important

You must restart PSpice A/D for the trace color changes.

In the `PSpice.ini` file, you can control the following print and display color settings for probe windows:

- The colors used to display traces
- The colors used for the Probe window foreground and background
- The order colors are used to display traces
- The number of colors used to display traces

To edit the `PSpice.ini` file to modify display and print colors, do the following:

1. In a standard text editor (such as Notepad), open `PSpice.ini`.

   The default location of `PSpice.ini` is:
   `%HOME%\cdssetup\OrCAD_PSpice\<release_version>\PSpice.ini`

2. Scroll to the `[PROBE DISPLAY COLORS]` or `[PROBE PRINTER COLORS]` section of this file.

3. Add or modify a color entry. Colors for all items are specified as `<item name>=<color>`.

   The item names and what they represent are listed in the table. Valid item names include:

   - BACKGROUND
   - FOREGROUND
   - TRACE_1 through TRACE_12

   Here are the color names you can specify:

black	blue
brightblue	brightcyan
brightgreen	brightmagenta
brightred	brightwhite
brightyellow	brown
cyan	darkblue
darkcyan	darkgray

@ 2023 All Rights Reserved.

black	blue
darkgreen	darkmagenta
darkpink	darkred
green	lightblue
lightgray	lightgreen
magenta	mustard
orange	pink
purple	red
white	yellow

4. If you added or deleted trace number entries, set `NUMTRACECOLORS=<n>` to the new number of traces, where `<n>` is between 1 and 12. This item represents the number of trace colors displayed on the screen or printed before the color order repeats.
5. Save the file.

Important

After editing `PSpice.ini`, you must restart PSpice A/D to ensure your changes take effect.

Item Name	Description	Default
BACKGROUND	specifies the color of window background	BLACK
FOREGROUND	specifies the default color for items not explicitly specified	WHITE
TRACE_1	specifies the first color used for trace display	BRIGHTGREEN
TRACE_2	specifies the second color used for trace display	BRIGHTRED
TRACE_3	specifies the third color used for trace display	BRIGHTBLUE
TRACE_4	specifies the fourth color used for trace display	BRIGHTYELLOW

@ 2023 All Rights Reserved.

Item Name	Description	Default
TRACE_5	specifies the fifth color used for trace display	BRIGHTMAGENTA
TRACE_6	specifies the sixth color used for trace display	BRIGHTCYAN

When you want to copy probe plots to the clipboard and paste them into a black and white document, do the following:

1. Choose *Window – Copy to Clipboard*.

   The Copy to Clipboard - Color Filter dialog box appears.

2. Choose *change all colors to black* in the *Foreground* section.
3. Click *OK*.

Configuring Trace Color Schemes

In the Probe Settings dialog box, you can set options for how the available colors and the color order specified in the `PSpice.ini` file are used to display the traces in a probe window. You can use:

- a different color for each trace
- the same color for all the traces that belong to the same y-axis
- the available colors in sequence for each y-axis
- the same color for all the traces that belong to the same waveform data file

To configure trace color schemes in the Probe Settings dialog box, do the following:

1. Choose *Tools – Options* to open the Probe Settings dialog box.
2. Under *Trace Color Scheme*, choose one of the following options:

Table 17-1

Options	To do this...
Normal	Use a different color for each trace (for up to 12 traces, depending on the number of colors set in the `PSpice.ini` file).

@ 2023 All Rights Reserved.

Table 17-1

Options	To do this...
Match Axis	Use the same color for all the traces that belong to the same y-axis. The title of the axis (by default, 1, 2, etc.) is the same color as its traces.
Sequential Per Axis	Use the available colors in sequence for each y-axis.
Unique by File	Use the same color for all the traces in one Probe window that belong to the same waveform data file.

3. Click *OK*.

PSpice A/D saves the selected color scheme for future waveform analyses.

@ 2023 All Rights Reserved.

Viewing Waveforms

If you are using one of the schematic capture application, you can either view waveforms automatically after you run a simulation, or you can monitor the progress of the simulation as it is running.

You do not need to exit PSpice A/D if you are finished examining the simulation results for one circuit and want to begin a new simulation from within the schematic capture application. However, PSpice A/D unloads the old waveform data file for a circuit each time that you run a new simulation of the circuit. After the simulation is complete, the new or updated waveform data file is loaded for viewing.

Setting up a Waveform Display

You can configure the way you want to view the waveforms in PSpice A/D by defining display settings in the *Probe Window* tab in the Simulation Settings dialog box.

The display settings in the *Probe Window* tab are explained in the following table:

Table 17-2

Settings	Waveform Display
Display Probe window when profile is opened.	Waveforms are displayed only when a .DAT file is opened from within PSpice A/D.

@ 2023
All Rights Reserved.

Settings	Waveform Display
Display Probe window... during simulation.	Waveforms are displayed as the simulation progresses (“marching waveforms”). **Note:** For digital data, marching waveforms are not supported.
Display Probe window... after simulation has completed.	Waveforms are displayed only after the full simulation has completed and all data has been calculated.
Show... All markers on open schematics.	Waveforms are displayed for those nets that have markers attached in the schematic.
Show... Last plot	Waveforms are displayed according to the last display configuration that was used in the Probe window.
Show... Nothing.	No waveforms are displayed.

Viewing Waveforms while Simulating

While a simulation is in progress, you can monitor the results for the data section being written by PSpice A/D. This function is only available when the *Display Probe* window during simulation option is enabled in the *Probe Window* tab of the *Simulation Settings* dialog box.

To monitor results during a simulation

1. From the design entry tool’s PSpice menu, choose Edit Simulation Profile to display the Simulation Settings dialog box.
2. Click the *Probe Window* tab.
3. Select Display Probe window and then click during simulation.
4. Click OK to close the Simulation Settings dialog box.
5. From the PSpice menu, choose Run to start the simulation.

   One Probe window is displayed in monitor mode.

   Note: If you open a new Probe window (from the Window menu, choose New Window) while monitoring the data, the new window also starts in monitor mode because it is associated with the same waveform data file.

   Note: During a multi-run simulation (such as Monte Carlo, parametric, or temperature),

@ 2023 All Rights Reserved.

PSpice A/D displays only the data for the most recent run in the Probe window.

6. Do one of the following to select the waveforms to be monitored:

   - ❑ From PSpice A/D's Trace menu, choose Add, and enter one or more trace expressions.
   - ❑ From Capture's PSpice menu, point to Markers, then choose and place one or more markers. Similarly, from Design Entry HDL's PSpice Simulator menu point to Probes and choose and place one or more probes.

     For more information, see Using schematic page markers to add traces on page 501.

   The Probe window monitors the waveforms for as long as the most recent data section is being written. After that data section is finished, the window changes to manual mode. To see the full set of runs, you must update the display by using the Add Trace command under the Trace menu.

Configuring update intervals

You can define the frequency at which PSpice A/D updates the waveform display as follows:

- At fixed time intervals (every *n* sec)
- According to the percentage of simulation completed (every *n* %), where *n* is user-defined

The default setting (Auto) updates traces each time PSpice A/D gets new data from a simulation.

To change the update interval

1. From the Tools menu, choose Options.
2. In the Auto-Update Interval frame, choose the interval type (sec or %), then type the interval in the text box.

Interacting with waveform analysis during simulation

The functions that change the x-axis domain (that set a new x-axis variable) can not be accessed while the simulation is running. If you have enabled the display of waveforms during simulation and wish to reconfigure the x-axis settings (as explained below), you must wait until the simulation run has finished.

@ 2023 All Rights Reserved.

The following table shows how to enable the functions that change the x-axis domain.

Table 17-3

Enable this function...	By doing this...
Fast Fourier transforms	**1.** From the Plot menu, choose Axis Settings. **2.** In the Processing Options frame, select Fourier.
Performance analysis	**1.** From the Plot menu, choose Axis Settings. **2.** In the Processing Options frame, select Performance Analysis.
New x-axis variable	**1.** From the Plot menu, choose Axis Settings, then click the X Axis tab. **2.** Click the Axis Variable button. **3.** In the X Axis Variable dialog box, specify a new x-axis variable.
Measurement evaluation	**1.** From the Trace menu, select Evaluate Measurement. **2.** In the Evaluate Measurement dialog box, specify a measurement.
Load a completed data section	**1.** From the File menu, choose Append Waveform (.DAT). **2.** Select a .DAT file to append.

Pausing a simulation and viewing waveforms

You can pause a simulation to analyze waveforms before the simulation is finished. After you pause the simulation, you can either resume the simulation or end it.

@ 2023 All Rights Reserved.

To pause a simulation

1. From PSpice A/D's Simulation menu, choose Pause.
2. In the Probe window, view the waveforms generated before you paused the simulation.
3. Do one of the following:
   - ❑ From the Simulation menu, choose Run to resume the simulation.
   - ❑ From the Simulation menu, choose Stop to stop the simulation.

Using schematic page markers to add traces

You can place markers on a schematic page to identify the nodes for which you want waveforms displayed in Probe.

You can place markers:

- Before simulation, to limit results written to the waveform data file and automatically display those traces in PSpice A/D.
- During or after simulation, with PSpice A/D running, to automatically display traces in the active Probe window.

After simulation, the color of the marker you place is the same as its corresponding waveform analysis trace. If you change the color of the trace, the color of the marker on the schematic page changes accordingly.

The Markers submenu also provides options for controlling the display of marked results in PSpice A/D, after initial marker placement, and during or after simulation.

Power markers allow you to measure the power dissipation of a particular device. You can use these markers in the same way you use current and voltage markers. Power markers are annotated with "W" and are placed on devices that have PSpice models. The corresponding power dissipation waveforms for the devices will be calculated and displayed in Probe.

Place markers on subcircuit nodes. This allows you to perform cross-probing between the front-end design entry tool and PSpice A/D at the lower level circuits of a hierarchical design.

To place markers on a schematic page

1. From Capture's PSpice menu, point to Markers, then choose the marker type you want to place. Similarly, from Design Entry HDL's PSpice Simulator menu, point to Probes, then

@ 2023 All Rights Reserved.

choose the probe type you want to place (Some of the markers are from the Advanced submenu.)

Table 17-4

Waveform	Markers/Probes menu command	Advanced submenu command
voltage	Voltage Level	not required
voltage differential	Voltage Differential	not required
current	Current Into Pin	not required
digital signal	Voltage Level	not required
dB[1]	Advanced	db Magnitude of Voltage db Magnitude of Current
phase[1]	Advanced	Phase of Voltage Phase of Current
group delay[1]	Advanced	Group Delay of Voltage Group Delay of Current
real[1]	Advanced	Real Part of Voltage Real Part of Current
imaginary[1]	Advanced	Imaginary Part of Voltage Imaginary Part of Current
power	Power Dissipation	not required

1. You can use these markers instead of the built-in functions provided in output variable expressions. However, these markers are only available after defining a simulation profile for an AC Sweep/Noise analysis.

@ 2023 All Rights Reserved.

2. Point to the wires or pins you wish to mark and click to place the chosen markers.
3. Right-click and select End Mode to stop placing markers.
4. If you have not simulated the circuit yet, from the PSpice menu, choose Run.

   After simulation, the color of the marker is the same as its corresponding waveform analysis trace. If you change the color of the trace, the color of the marker changes accordingly.

To hide or delete marked results

1. From Capture's PSpice menu, point to Markers, then choose one of the following:

Table 17-5

Choose this option...	To do this...
Hide All	Hide traces in the waveform analysis display for all markers placed on any page or level of the schematic.
Delete All	Remove all markers from the schematic and all corresponding traces from the waveform analysis display.

@ 2023 All Rights Reserved.

Using Display Control

You can create displays to save the contents of a Probe window. You can view a display again at a later time with a different simulation so long as the new simulation has identically named variables.

Once the display is saved, you can copy it, edit it, and delete it.

To save a display

1. Set up the plots, traces, labels, and axes in the Probe window you want to save.
2. From the Window menu, choose Display Control.

   The Display Control dialog box appears.
3. Click on the Displays tab.
4. In the New Name text box, type a name for the display.
5. Do one of the following:
   - ❑ To save the display in the current .PRB file, click Save.
   - ❑ To save the display in another .PRB file, click Save To. Specify the name and location of the file. Click OK.

@ 2023 All Rights Reserved.

6. Click Close.

To copy a display

1. From the Window menu, choose Display Control.

   The Display Control dialog box appears.
2. Click on the Displays tab.
3. Click the name of the display to copy.
4. Click Copy To.
5. Specify the name and location of the copied display.
6. Click OK.
7. Click Close.

To delete a display

1. From the Window menu, choose Display Control.

   The Display Control dialog box appears.
2. Click on the Displays tab.
3. Do one of the following:
   - To delete a display from the current .PRB file, click the name, then click Delete.
   - To delete a display from a global or remote .PRB file, click Delete From, then select the .PRB file.
4. Click Close.

To use a saved display

1. From the Window menu, choose Display Control.

   The Display Control dialog box appears.
2. Click on the Displays tab.
3. Do one of the following:
   - To use a display listed here, click the name.

@ 2023 All Rights Reserved.

- To use a display from another .PRB, click Load. Select the file. Click OK. Click the name of the display.

4. Click Restore.

Note: You can use a saved display to display traces as long as the current data file has variables with the same names as the variables in the display file.

To load displays from another .PRB file

1. From the Window menu, choose Display Control.

   The Display Control dialog box appears.

2. Click on the Displays tab.
3. Click Load.
4. Select the file.
5. Click OK.

@ 2023 All Rights Reserved.

Using plot window templates

PSpice A/D provides plot window templates that allow you to create and reuse custom displays in Probe using defined arguments. A plot window template is a plot window consisting of one or more arguments used to represent node voltage, pin current, power or digital names within a display. An argument provides the means to replace a fixed node voltage or pin current name with a node voltage or pin current name you choose.

You can create unique plot window templates for a particular design or general templates that can be applied to various designs. A set of some of the more commonly used templates are predefined and included with PSpice A/D.

To work with plot window templates, from the Window menu, choose Display Control, and click the Templates tab. Here you can customize plot window templates in various ways.

Creating a plot window template

In order to create and save a new plot window template, you must first set up the active plot window in Probe with the configuration you want. The active plot window will be the basis for the template properties you save.

Note: A plot window template that is saved to a simulation profile that you define cannot be used with markers in OrCAD X Capture. The list of plot window templates displayed in PSpice A/D includes user-defined templates, but the list in OrCAD X Capture does not show these.

@ 2023 All Rights Reserved.

To view user-defined templates in OrCAD X Capture, you must copy the templates to either the local or global .PRB file.

To create a new plot window template

1. In PSpice A/D, from the Window menu, choose Display Control.
2. Click the Templates tab.

3. In the New Name text box, enter the name for the new template you want to create.
4. Click Save or Save To.

@ 2023 All Rights Reserved.

The Save Plot Window Template – Step 1 of 2 dialog box appears.

5. In the Description text box, type in a description for the template, if you would like one. (This is optional.)

6. If you clicked Save To, choose the .PRB file you wish to save the template to by selecting the appropriate radio button under the Save Template In frame. (The default is the local .PRB file. For the Save function, the Local File is the only option.)

   - Local File – the .PRB file for the current simulation in PSpice A/D.
   - Global File – the .PRB file to be used globally for all Probe displays.
   - Other File – another .PRB file stored elsewhere on your hard disk or network drive. Use the Browse button to locate the file on a particular drive.

7. Click the Next> button.

@ 2023 All Rights Reserved.

The Save Plot Window Template – Step 2 of 2 dialog box appears. The number of Node/Pin Name arguments that are listed here is determined by the current display.

8. Define the association of each argument by selecting the node or pin name from the drop-down list under the column Node/Pin Name.

   This drop-down list shows all of the available node voltage, pin current, power or digital names. If the drop-down list does not appear, click in the text box to activate the drop-down button.

9. For each argument, set the Type of argument to be used by selecting the argument name from the drop-down list under the column Type.

   This drop-down list shows all of the available argument types (any, current, power, voltage). If the drop-down list does not appear, click in the text box to activate the drop-down button.

10. For each argument, under the Description column, type in a description for the argument, if you would like one. (This is optional.)

    The description you enter here will be displayed in the status line of Capture when placing a marker associated with the argument.

11. If desired, change the order of the arguments by using the Arrow buttons to move an argument up or down in the listing. Or, you can delete an argument by selecting it and clicking the Delete button.

12. Click Finish.

Note: At least one argument is required to create a plot window template. The maximum number of arguments allowed is the number of unique node voltage, pin current, power or digital names in the active display.

@ 2023 All Rights Reserved.

Modifying a plot window template

Modifying a plot window template is essentially the same as creating a new template. In order to modify a plot window template, that particular template must be the active plot window in Probe. If the active display is not the template you want to modify, use the Restore button to make a different template the active display in Probe

To modify a plot window template

1. From the Window menu, choose Display Control.
2. Click the Templates tab.

3. Select the template you want to modify by clicking on its name in the list of loaded templates. If the template you are looking for is not in the list, use the Restore button to make it the active display.

@ 2023 All Rights Reserved.

4. Click the Save button to display the Save Plot Window Template – Step 1 of 2 dialog box.

5. Make the desired changes, then click Next to display the Save Plot Window Template – Step 2 of 2 dialog box.

6. Make the desired changes, then click Finish.

The modifications will be saved and the display will be updated automatically.

Note: If an argument assignment no longer applies because the node voltage, pin current, power or digital names are mapped to an argument that has changed, then information regarding that argument will not be available in the Step 2 of 2 dialog box.

@ 2023 All Rights Reserved.

Deleting a plot window template

You can easily delete a plot window template from the list of loaded templates. This does not erase the template from your system. It only removes it from the list of templates you can access and erases it from the .PRB file.

To delete a plot window template

1. From the Window menu, choose Display Control.
2. Click the Templates tab.
3. Click on the name of the plot window template you want to delete.
4. Click Delete.

Copying a plot window template

You can copy a plot window template into another .PRB file to make it available for use later with that file.

To copy a plot window template

1. From the Window menu, choose Display Control.
2. Click the Templates tab.
3. Click on the name of the plot window template you want to copy.
4. Click Copy To.

   The Probe File for Save Template dialog box appears.
5. Choose the .PRB file you wish to save the template to by selecting the appropriate radio button under the Save Template In frame. (The default is the local .PRB file.)
   - Local File – the .PRB file for the current simulation in PSpice A/D.
   - Global File – the .PRB file to be used globally for all Probe displays.
   - Other File – another .PRB file stored elsewhere on your hard disk or network drive. Use the Browse button to locate the file on a particular drive.
6. Click OK.

@ 2023 All Rights Reserved.

Restoring a plot window template

In order to make a plot window template the active display in Probe, you must restore it. This process recalls a previously defined plot window template and sets up a new plot window in Probe using the arguments associated with that template. In order for the arguments in the template to apply, you must replace the node voltage names or pin current names for each argument contained in the restored template.

Note: You can only restore plot window templates that are already loaded. If you want to restore a plot window template that does not appear in the list, you must first load it.

To restore a plot window template

1. From the Window menu, choose Display Control.
2. Click the Templates tab.
3. Click Restore.

   The Restore Plot Window Template dialog box appears.

4. Reassign the node voltage names or pin current names for each argument in the list.
5. Click OK.

   A new Probe window will be created and the restored plot window template will be displayed.

@ 2023 All Rights Reserved.

Note: You may also restore a plot window template by choosing the Add Trace command from the Trace menu, and then selecting Plot Window Templates from the drop-down list in the Functions or Macros frame.

Viewing the properties of a plot window template

You can view the properties of a plot window template and change the description fields for the template or arguments it contains.

Note: A plot window template that is saved to a simulation profile that you define cannot be used with markers in OrCAD X Capture. The list of plot window templates displayed in PSpice A/D includes user-defined templates, but the list in OrCAD X Capture does not show these. To view user-defined templates in OrCAD X Capture, you must copy the templates to either the local or global .PRB file.

To view the properties of a plot window template

1. From the Window menu, choose Display Control.
2. Click the Templates tab.
3. Click on the name of the plot window template you want to view.
4. Click Properties.

@ 2023 All Rights Reserved.

The Plot Window Template Properties dialog box appears.

5. Change the Description field for the template, or change the description for any of the Arguments, as desired.
6. Click Finish to exit and save any changes.

Note: When viewing the properties of a template, you can only edit the description fields. No other changes are allowed. If you want to modify the arguments or assignments, you need to modify the template.

Loading a plot window template

You can load a plot window template from another .PRB file, and add it to the list of available templates. When you load a template, you do not make it the active display in Probe. You are only adding it to the list of available templates. (To view the newly loaded template, you need to restore it.)

If a duplicate template is loaded, then the one you are loading will replace the current one in the list. If you close the data file and reopen it, any plot window templates that you loaded earlier will have to be loaded again to make them available. (Loaded templates are not saved with the data file.)

@ 2023 All Rights Reserved.

To load a plot window template

1. From the Window menu, choose Display Control.
2. Click the Templates tab.
3. Click Load.

   The Load Displays dialog box appears.
4. Locate the .PRB file that contains the plot window template you want to load.
5. Select the file and then choose Open.

   The loaded templates will be listed in the Display Control dialog box.

Placing plot window template markers in Capture

Place a marker in Capture that represents a plot window template. The marker will restore the associated template when you run the simulation in PSpice A/D. Markers for plot window templates are distinguished from other markers (for voltage, current, or power) by being square rather than round in shape.

A simulation profile must be active in order to place a marker for a plot window template. The analysis type defined in the profile will determine what type of template will be loaded (either for AC, DC or transient analysis).

When placing a plot window template marker, the argument description for the template being placed will appear in the status bar of Capture. Markers will continue to be placed until all arguments for the template have been satisfied. If an active simulation exists, then the template markers will turn black; otherwise, they will remain gray.

If an argument type is set to "Any" rather than a specific type, the marker type will depend on the marker placement location. If a marker is placed on a pin, then it will be assumed to be a

@ 2023 All Rights Reserved.

current marker. If a marker is placed on a node, it will be assumed to be a voltage marker. If a marker is placed on a device, it will be assumed to be a power marker.

To place a plot window template marker

1. In Capture, from the PSpice menu, choose Markers, then select Plot Window Templates.

   The Plot Window Templates dialog box appears.

2. Click on the template you want to associate with the marker you will place.
3. Click the Place button.

   A plot window template marker will appear and be attached to the cursor.
4. Place the marker at a particular location on the schematic page.
5. Continue to place markers at the appropriate locations until all the arguments for the template have been satisfied.

Note: PSpice A/D does not have to be running in order for you to place a marker for a plot window template. The list of loaded templates comes from either the default PSPICE.PRB file or from the .PRB file for the active profile, if that exists.

@ 2023 All Rights Reserved.

Caution

Plot Window Templates are displayed properly only if one template is added at a time. Therefore, in a situation where multiple Plot Window Templates are added before simulation, all the templates will be ignored. In cases where simulation is done first and markers are added later on, every thing works fine, since only one Plot Window Template gets added at a time.

Limiting Waveform Data File Size

When PSpice A/D performs a simulation, it creates a waveform data file. The size of this file for a transient analysis is roughly equal to:

(# transistors)·(# simulation time points)·24 bytes

The size for other analysis types is about 2.5 times smaller. For long runs, especially transient runs, this can generate waveform data files that are several megabytes in size. Even if this does not cause a problem with disk space, large waveform data files take longer to read in and take longer to display traces on the screen.

You can limit waveform data file size by:

- placing markers on your schematic before simulation and having PSpice A/D restrict the saved data to these markers only
- excluding data for internal subcircuits
- suppressing simulation output

Limiting file size using markers

One reason that waveform data files are large is that, by default, PSpice A/D stores *all net voltages and device currents* for each step (for example, time or frequency points). However, if you have placed markers on your schematic prior to simulation, PSpice A/D can save only the results for the marked wires and pins.

To limit file size using markers

1. From Capture's PSpice menu, choose *Edit Simulation Profile*.

   The Simulation Settings dialog box appears.

@ 2023 All Rights Reserved.

2. Click the *Data Collection* tab.

3. In the *Data Collection Options* section, choose the desired option for each type of marker (Voltages, Current, Power, Digital, Noise).

Table 17-6

Option	Description
All	All data will be collected and stored. (*This is the default setting*.)
All but Internal Subcircuits	All data will be collected and stored except for internal subcircuits of hierarchical designs (top level data only).
At Markers only	Data will only be collected and stored where markers are placed.
None	No data will be collected.

4. Check the *Save data in the CSDF format (.CSD)* if you want the data to be stored in this format.

   Note: By default, the probe data has an accuracy of *64-bit*. You can change this to *32-bit*.

5. Click *OK*.

6. From the PSpice menu, select *Markers*, then choose the marker type you want to place.

@ 2023 All Rights Reserved.

The color of the marker on the schematic page is the same as its corresponding waveform analysis trace. If you change the color of the trace, the color of the marker changes accordingly.

7. Point to the wires or pins you wish to mark and click to place the chosen markers.
8. Right-click and select *End Mode* to stop placing markers.
9. From the PSpice menu, choose *Run* to start the simulation.

   When the simulation is complete, the corresponding waveforms for the marked nodes or devices will be displayed in Probe.

Limiting file size by excluding internal subcircuit data

By default, PSpice A/D saves data for all internal nodes and devices in subcircuit models in a design. You can exclude data for internal subcircuit nodes and devices.

To limit file size by excluding data for internal subcircuits

1. From the Capture's PSpice menu, choose *Edit Simulation Profile*.

   The Simulation Settings dialog box appears.
2. Click the *Data Collection* tab.
3. In the Data Collection Options section, choose `All but Internal Subcircuits` for each marker type.
4. Click *OK*.
5. From the PSpice menu, choose *Run* to start the simulation.

Limiting file size by suppressing the first part of simulation output

Long transient simulations create large waveform data files because PSpice A/D stores many data points. You can suppress a part of the data from a transient run by setting the simulation analysis to start the output at a time later than 0. This does not affect the transient calculations themselves—these always start at time 0. This delay only suppresses the output for the first part of the simulation.

Note: Suppressing part of the data from a run also limits the size of the PSpice output file.

@ 2023 All Rights Reserved.

To limit file size by suppressing the first part of transient simulation output

1. From Capture's PSpice menu, choose Edit Simulation Profile to display the Simulation Settings dialog box.
2. Click the Analysis tab.
3. From the Analysis type list, select the Time Domain (Transient) option.
4. In the Start saving data after text box, type a delay time.
5. Click OK to close the Simulation Settings dialog box.
6. From the PSpice menu, choose Run to start the simulation.

   The simulation begins, but no data is stored until after the delay has elapsed.

Viewing large data files

At times there are situations where you cannot reduce the size of the data file being generated, but would like to load large data files on to the Probe for viewing the trace.

As PSpice A/D depends on system memory for loading and displaying any trace, if the data file is so large that cannot be loaded, the trace does not get displayed at all. Usually data files with size greater than 2 GB are considered to be large data files. Whether a data file can be categorized as large depends on the size of the data file and the number of traces within the data file. For example, a data file that has 10 traces and is of 2 GB may not be a large data file. But if a data file has single trace and is of 2 GB or more, it can be termed as a large data file.

Tip

The percentage progress of loading data file is displayed on the status bar. when loading large data files or loading files across network You can use this information to approximate the time taken.

Important

By default, any data file with more than 1 million data points per trace is considered as large data file by PSpice A/D.

Large data files are usually loaded faster compared to the smaller data files in terms of rate of loading. However, PSpice A/D provides options to optimize the loading and viewing of large data files. To load and view data files with more than 1 million data points, PSpice A/D provides you with the following options.

@ 2023 All Rights Reserved.

- Displaying fewer data points
- Displaying Partial Trace

Displaying fewer data points

If a data file is too large to be loaded on to PSpice A/D, you can choose to display the complete trace that has been created using a fewer data points.

For example, if the complete trace uses 3 million data points, the trace displayed on the probe window will be created using only 1 million data points.

In this option though the complete trace is displayed, it is not very accurate. The number of data points used to construct the complete trace depends on the system memory available for loading data points. By default, the limit is set to 1 million data points per part of the trace. If required you can increase this limit. For more information see, Changing threshold for large data file.

@ 2023 All Rights Reserved.

Displaying Partial Trace

Another method of loading and displaying a large data file on to PSpice, is to break the complete trace into multiple smaller parts. You can then view each part separately in the PSpice A/D probe window. See the following figure:

Figure 17-3 Displaying partial traces

Number of parts into which the complete trace can be divided is governed by the number of data points in the trace.

For example, if a complete trace has 4.5 million data points, and at any given point of time PSpice A/D is configured to load only 1 million data points, the complete trace will be divided into 5 parts. First four part traces will be of 1 million data points each and the last part trace will be of 500000 data points.

@ 2023 All Rights Reserved.

The size of the partial trace is also influenced by the number of data points per part. By default, the limit is set to 1 million data points per part of the trace. If required you can increase this limit. For more information see, Changing threshold for large data file.

When you display partial traces, marching waveforms are not supported.

@ 2023
All Rights Reserved.

You can also use the *Load Next Part* button in the *Probe* toolbar for loading and viewing the next part of the trace. The down-arrow button is used to display the next 5 or the previous 5 parts of the trace or both as the case may be.

Loading a large data file

If you are trying to open a large data file in PSpice A/D, the Large Data File dialog box appears.

@ 2023
All Rights Reserved.

To load and display the data file, you can either select the *Use fewer data points to display complete trace option* or the *Use all data points to display trace in parts* option.

To load the complete data file, select the *Ignore this warning* option. When you select this option, the data file may or may not get loaded depending on your system memory.

For example, on some machines a data file with 5 million data points may open without any problems while on some other machine the same data file might be too large to be loaded in one go.

To make your selection the default setting, select the *Always use this option* check box. Once this check box is selected, next time when you try to open a large data file, the Large Data File dialog box does not appear. Instead, the option selected by you previously is used to open the large data file.

The following two options are disabled for marching waveforms:

- Use fewer data points to display complete trace
- Use all data points to display trace in parts

Switching Modes

While viewing the trace from a large data file, you can switch from one mode to another using one of the following methods.

- Using the View drop-down menu
  - **a.** From the *View* menu in PSpice A/D select *Large Data File Mode*.
  - **b.** In the Large Data File Mode submenu, select the mode in which you want to view the trace. The mode currently selected, has a check mark against it.
    - Select *Display Fewer Data Points* when you want to view the complete trace that may not be very accurate.
    - Select *Display in parts* when you want to view the partial but an accurate representation of the trace.

- Using the toolbar

@ 2023
All Rights Reserved.

a. To switch modes, select the *Toggle Large Data File Mode* button in the Probe toolbar.

Changing threshold for large data file

By default, in PSpice A/D any data file with more than 1 million data points is categorized as a large data (`.dat`) file. As a user you can modify this threshold by specifying a higher the number of data points in the Probe Settings dialog box.

Important

You cannot reduce the limit to lower than 1 million data points.

To increase the number of data points complete the following steps:

1. From the *Tools* menu in PSpice A/D choose *Options*.

   Note: This option is enabled only if a `.dat` or `.sim` file is open in PSpice A/D.

@ 2023
All Rights Reserved.

2. Select the *Large Data File* tab.

3. In the *Data points in one part* text box specify the desired number of data points and click OK.

The changes you have made in the Probe Setting dialog box, will be effective from the next session.

Caution

Any changes to the threshold for defining large data file should be made keeping in account your system memory so as to avoid failure in opening data files. For example, on a machine low on system memory, a data file or a part of data file with 2 million data points might fail to open. On the other hand, for a high-end machine a limit of 3 million data points might work fine.

@ 2023
All Rights Reserved.

Using simulation data from multiple files

You can load simulation data from multiple files into the same Probe window by appending waveform data files.

When more than one waveform data file is loaded, you can add traces using all loaded data, data from only one file, or individual data sections from one or more files.

Appending waveform data files

To append a waveform data file

1. In PSpice A/D, from the File menu, choose Append Waveform (.DAT).
2. Select a `*.DAT` file to append, and click OK.
3. If the file has multiple sections of data for the selected analysis type, the Available Sections dialog box appears. Do one of the following:
   - Click the sections you want to use.
   - Click the All button to use all sections.
4. Click OK.

@ 2023 All Rights Reserved.

Importing traces

Besides using the simulation data from .DAT files, you can also import trace information stored in other file formats. Using PSpice A/D you can import the traces stored in a tabular format in a `.txt` file or a `.csv` file.

Traces in tabular format

Frequency	I(U1.C1)	I(U1.U3A:OUT)	V(U1.OUT)
1.000000000000e+002	9.153080338913e-008	4.385988900734e-007	1.655074006569e-(
1.047128548051e+002	9.590496492142e-008	4.592854964129e-007	1.733089367376e-(
1.096478196143e+002	1.004941801824e-007	4.809493978440e-007	1.814783717038e-(
1.148153621497e+002	1.053099039612e-007	5.036370543716e-007	1.900330519110e-(
1.202264434617e+002	1.103643108887e-007	5.273970825797e-007	1.989911736160e-(
1.258925411794e+002	1.156703566844e-007	5.522804907949e-007	2.083718179127e-(
1.318256738556e+002	1.212418870332e-007	5.783407451049e-007	2.181948925362e-(
1.380384264603e+002	1.270936530019e-007	6.056339502295e-007	2.284813530653e-(

To import a trace

1. From the File drop-down menu choose Import.
2. In the Import File dialog box, select the text file to be imported in PSpice A/D.

   The Import Traces dialog box appears. All the nodes listed in the source file are listed in the X-axis drop-down list and the Available Nodes list.
3. In the Import Traces dialog box, specify the name and the location of the .DAT file in which the imported trace is to be stored.
4. From the X-Axis drop-down list box, select the node name to be plotted on the X-Axis.
5. Specify a name for the X-axis. Select one of the following options for naming the X-axis.
   - **a.** Time
   - **b.** Frequency
   - **c.** Sweep Variable: When you select Sweep Variable, you need to specify the variable name in the enabled text box.
6. From the Available Traces list box, select the traces that are to be imported in PSpice A/D.

@ 2023 All Rights Reserved.

7. Click Add.

8. The selected traces appear in the Import Trace list box. To import all the traces available in the source file, click the Add All.

9. Select OK to import the selected trace(s) into the specified data file.

Important

Renaming of X-axis is useful when you want to import a trace and append it to an existing data file for comparing traces. To be able to append two traces successfully, you need to ensure that for both the traces the range and the name of the variable plotted on the X-axis should be same.

@ 2023 All Rights Reserved.

Adding traces from specific loaded waveform data files

If two or more waveform data files have identical simulation output variables, trace expressions that include those variables generate traces for each file. However, you can specify which waveform data file to use in the trace expression. You can also determine which waveform data file was used to generate a specific trace.

To add a trace from a specific loaded waveform data file

1. In PSpice A/D, from the Trace menu, choose Add Trace to display the Add Traces dialog box.

   The Simulation Output Variables list in the Add Traces dialog box contains the output variables for all loaded waveform data files.

2. In the Trace Expression text box, type an expression using the following syntax:

   trace_expression@f*n*

   where *n* is the numerical order (from left to right) of the waveform data file as it appears in the PSpice title bar, or

   trace_expression@*s*@f*n*

   where *s* is a specific data section of a specific waveform data file.

   Example: To plot the V(1) output for data section 1 from the second data file loaded, type the following trace expression:

   ```
   V(1)@1@f2
   ```

   You can also use the name of the loaded data file to specify it. For example, to plot the V(1) output for all data sections of a loaded data file, MYFILE.DAT, type the following trace expression:

   ```
   V(1)@"MYFILE.DAT"
   ```

3. Click OK.

@ 2023 All Rights Reserved.

To identify the source file for an individual trace

1. In the trace legend, double-click the symbol for the trace you want to identify.

Figure 17-4 Trace legend symbols.

The Section Information dialog box appears, containing the trace name and—if there is more than one waveform data file loaded in the plot—the full path for the file from which the trace was generated.

Also listed is information about the simulation that generated the waveform data file and the number of data points used in the following figure:

Figure 17-5 Section information message box.

Comparing measured data with simulated data

You can compare measured data with simulated data in the Probe. You need to import the measured data into the circuit to enable comparison. To import the data, perform the following steps:

@ 2023 All Rights Reserved.

1. Put the measured data into a text file as a list of two column rows. The first column specifies the time and the second column specifies the value at that time. Separate the columns using space.

2. Place a VPWL_file or IPWL_file part on the schematic.

3. Set the FILE_ATTRIBUTE property of the part to the path of the text file. If you placed the measured data text file in the profile folder, specify only the name of the text file. Otherwise, specify the absolute path to the file.

4. Terminate the placed VPWL_file or IPWL_file part with a resistor to ground.

Note: Terminating the placed part to ground with a resistor ensures that the voltage across the resistor or current through the resistor represents the imported measured data.

Probe displays the value of voltage or current as a trace when you run the simulation. You can compare this trace with the traces for simulated values.

Saving Simulation Results in ASCII Format

The default waveform data file format is binary. However, you can save the waveform data file in the Common Simulation Data Format (CSDF) instead.

Caution

Data files saved in the CSDF format are two or more times the size of binary files.

When you first open a CSDF data file, PSpice A/D converts it back to the .DAT format. This conversion takes two or more times as long as opening a .DAT file. PSpice A/D saves the new .DAT file for future use.

To save simulation results in ASCII format

1. From PSpice A/D's Simulation menu, choose *Edit Settings* to open the Simulation Settings dialog box.
2. Click the *Data Collection* tab.
3. Select the *Save data in the CSDF format (.CSD)* check box.
4. Click *OK*.

@ 2023 All Rights Reserved.

PSpice A/D writes simulation results to the waveform data file in ASCII format (as `*.CSD` instead of `*.DAT`), following the CSDF convention.

@ 2023 All Rights Reserved.

User Interface Features For Waveform Analysis

PSpice A/D provides direct manipulation techniques and shortcuts for analyzing waveform data. These techniques are described below.

Many of the menu commands in PSpice A/D have equivalent keyboard shortcuts. For instance, after placing a selection rectangle in the analog portion of the plot, you can type *Ctrl+A* instead of choosing Area from the View menu. For a list of shortcut keys, see *PSpice A/D Help*.

Zoom Regions

PSpice A/D provides a direct manipulation method for marking the zoom region in either the digital or the analog area of the plot.

Tip

You can zoom in or zoom out by using the mouse wheel. Press the Control key and scroll the mouse wheel to zoom in or zoom out. Alternatively, you can press `I` or `i` to zoom in and `o` or `O` to zoom out in the Probe window.

To zoom in or out

1. Do one of the following on the toolbar:
   - Click the View In toolbar button to zoom in by a factor of 2 around the point you specify.
   - Click the View Out toolbar button to zoom out by a factor of 2 around the point you specify.

To zoom in the digital area using the mouse

1. In the digital area, drag the mouse pointer left or right to produce two vertical bars.

@ 2023 All Rights Reserved.

Note: Click the mouse anywhere on the plot to remove the vertical bars without zooming.

2. From the View menu, point to Zoom, then choose Area.

   PSpice A/D changes the plot display to the area in between the selection bars. If the plot includes an analog area, it is zoomed in as well.

To zoom in the analog area using the mouse

1. Drag the mouse pointer to make a selection rectangle as shown below.

   Note: Click anywhere on the plot to remove the selection rectangle without zooming.

2. From the View menu, point to Zoom, then choose Area.

   PSpice A/D changes the plot to display the region within the selection rectangle. The digital portion of the display, if present, is also zoomed.

Scrolling Traces

By default, when a plot is zoomed or when a digital plot contains more traces than can be displayed in the visible area, standard scroll bars appear to the right or at the bottom of the plot area as necessary. These can be used to pan through the data. You can configure scroll bars so they are always present or are never displayed.

@ 2023 All Rights Reserved.

Tip

You can also use the mouse wheel to scroll in the Probe window.

To configure scroll bars

1. In PSpice A/D, from the Tools menu, choose Options.
2. In the Use Scroll Bars frame, choose one of the scroll bars options, as described below.

Choose this option...	To do this...
Auto	Have scroll bars appear when a plot is zoomed or when additional traces are displayed in the plot but are not visible (default).
Never	Never display scroll bars. This mode provides maximum plot size and is useful on VGA and other low resolution displays.
Always	Display scroll bars at all times. However, they are disabled if the corresponding axis is full scale.

Showing and Hiding Traces

You can show or hide one or more traces by right-clicking on a trace and choosing one of the following options:

- *Hide Trace*: Hides the selected trace.
- *Hide All Traces*: Hides all displayed traces.
- *Show All Traces*: Shows all traces.

You can also right-click on the probe window to choose either *Show All Traces* or *Hide All Traces*.

@ 2023 All Rights Reserved.

Sizing Digital Plots

Sizing bars can be used to change the digital plot size instead of choosing Digital Size from the Plot menu. The digital trace name sizing bar is at the left vertical boundary of the digital plot. If an analog plot area is displayed simultaneously with the digital plot, there is an additional plot sizing bar at the bottom horizontal boundary of the digital plot.

To set the digital plot size using the mouse

1. Display at least one digital trace and one analog trace in the Probe window for which you want to set the digital size.
2. To change the bottom position of the digital Probe window, do the following:
   - **a.** Place the mouse pointer between the analog and digital parts of the plot.
   - **b.** Click the plot separator.
   - **c.** Drag the plot separator until you have the digital size you want.
3. To change the left side of the digital Probe window, do the following:
   - **a.** Place the mouse pointer at the left edge of the digital Probe window you want to resize.
   - **b.** Click the left edge.
   - **c.** Drag the left edge of the digital Probe window to adjust the space available for displaying digital trace names.

To set the digital plot size using menu options

1. Display at least one digital trace in the plot for which you want to set the digital size.
2. From the Plot menu, choose Digital Size.
3. In the Digital Size dialog box, set the following:
   - ❑ Percentage of Plot to be Digital
   - ❑ Length of Digital Trace Name
4. Click OK.

@ 2023 All Rights Reserved.

Modifying Trace Expressions and Labels

You can modify trace expressions, text labels, and ellipse labels that are currently displayed within the Probe window, thus eliminating the need to delete and recreate any of these objects.

1. To place a label, click Plot, point to Label and then choose the desired type of object you want to place.

   Note: You can click *Font* and specify a different font style for each label text.

   For information about adding labels (including text, line, poly-line, arrow, box, circle, ellipse, and mark), refer to *PSpice A/D Help*.

To modify trace expressions

1. Click the trace name to select it (selection is indicated by a color change).
2. From the Edit menu, choose Modify Object.

   Note: You can also double-click the trace name to modify the trace expression.

3. In the Modify Trace dialog box, edit the trace expression just as you would when adding a trace.

   For more information on adding traces, see Adding traces from specific loaded waveform data files on page 533 and To add traces using output variables on page 553.

To modify text and ellipse labels

1. Click the text or ellipse to select it (selection is indicated by a color change).
2. From the Edit menu, choose Modify Object.

   Note: You can also double-click a text or ellipse label to modify it.

3. Edit the label by doing one of the following:
   - ❑ In the Ellipse Label dialog box, change the inclination angle.
   - ❑ In the Text Label dialog box, change the text label.

To modify default label font

1. Choose *Tools - Options* to open the Probe Settings dialog box
2. Click the *Font Settings* tab.

@ 2023 All Rights Reserved.

3. Click *Change Label Font* to open the Font dialog box.
4. Specify the type, style, size, color, and effects.
5. Click *OK*.

   You must restart PSpice A/D for the font changes.

Moving and Copying Trace Names and Expressions

Trace names and expressions can be selected and moved or copied, either within the same Probe window or to another Probe window.

To copy or move trace names and expressions

1. Click one or more (*Shift*+click) trace names. Selected trace names are highlighted.
2. From the Edit menu, choose Copy or Cut to save the trace names and expressions to the clipboard. Cut removes trace names and traces from the Probe window.
3. In the Probe window where traces are to be added, do one of the following:
   - To add trace names to the end of the currently displayed set, choose Paste from the Edit menu.
   - To add traces before a currently displayed trace name, select the trace name and then choose Paste from the Edit menu.
4. Click OK.

Tip

When adding a trace to a Probe window, you can make the trace display name different from the trace expression:

a. From the Trace menu, choose Add Trace.

b. In the Trace Expression text box, enter a trace expression using the syntax:

```
trace_expression[;display_name]
```

Here are some considerations when copying or moving trace names and expressions into a different Probe window:

- If the new Probe window is reading the same waveform data file, the copied or moved trace names and expressions display traces that are identical to the original selection set.

@ 2023 All Rights Reserved.

- If the new Probe window is reading a *different* waveform data file, the copied or moved names and expressions display different traces generated from the new data.

  For example, suppose two waveform data files, `MYSIM.DAT` and `YOURSIM.DAT` each contain a V(2) waveform. Suppose also that two Probe windows are currently displayed where window A is loaded with `MYSIM.DAT`, and window B is loaded with `YOURSIM.DAT`.

  When V(2) is copied from window A to window B, the trace looks different because it is determined by data from `YOURSIM.DAT` instead of `MYSIM.DAT`.

Copying and Moving Labels

Labels can be selected and moved or copied, either within the same Probe window or to another Probe window.

For information about adding labels (including text, line, poly-line, arrow, box, circle, ellipse, and mark), refer to *PSpice Help*.

To copy labels

1. Select one or more (*Shift*+click) labels, or select multiple labels by drawing a selection rectangle. Selected labels are highlighted.
2. From the Edit menu, choose Copy or Cut to save the labels to the clipboard.

   Cut removes labels from the Probe window.
3. Switch to the Probe window where labels are to be added, and from the Edit menu, choose Paste.
4. Click on the new location to place the labels.

To move labels

1. Select one or more (*Shift*+click) labels, or select multiple labels by drawing a selection rectangle. Selected labels are highlighted.
2. Move the labels by dragging them to a new location.

Tabulating Trace Data Values

You can generate a table of data points reflecting one or more traces in the Probe window and use this information in a document or spreadsheet.

@ 2023 All Rights Reserved.

To view the trace data values table

1. Select one or more (*Shift*+click) trace names. Selected trace names are highlighted.
2. From the Edit menu, choose Copy or Cut to save the trace data point values to the Clipboard.

   Cut removes traces from the Probe window.
3. In Clipboard Viewer, from the Display menu, choose either Text or OEM Text.

To export the data points to other applications

1. Select one or more (*Shift*+click) trace names. Selected trace names are highlighted.
2. From the Edit menu, choose Copy or Cut to save the trace data point values to the Clipboard.

   Cut removes traces from the Probe window.
3. Paste the data from the Clipboard into a text editor, a spreadsheet program, or a technical computing program (such as Mathcad).
4. Save the file.

Note: Saving the data directly to a file from Clipboard Viewer can create superfluous data at the beginning of the file.

Using Cursors

When one or more traces are displayed, you can use cursors to display the exact coordinates of two points on the same trace, or points on two different traces. In addition, differences are shown between the corresponding coordinate values for the two cursors.

The cursor information is shown in a dockable cursor window. The cursor window displays the trace name, trace color, and the different Y-axis values, including differences on the Y-axis of cursor values.

Trace Color	Trace Name	Y1	Y2	Y1 - Y2		Y1(Cursor1) - Y2(Cursor2)		-12.446		
	X Values	12.192m	0.000	12.192m		Y1 - Y1(Cursor1)	Y2 - Y2(Cursor2)	Max Y	Min Y	Avg Y
CURSOR 2	v(out)	18.006	12.477	5.5290		17.975	0.000	18.006	12.477	15.242
CURSOR 1	I(R3)	30.598m	64.552m	-33.954m		0.000	-12.412	64.552m	30.598m	47.575m

@ 2023 All Rights Reserved.

Tip

You can select *Show non-dockable (old) cursor window* in the *Cursor Settings* tab of the Options dialog box to change to the non-dockable cursor window. You can move the cursor window anywhere over the Probe window by dragging the box to another location.You can also select *Show cursor window when cursor is off* to manage the display of the cursor window.

Displaying Cursors

To display both cursors

1. Choose *Trace – Cursor – Display*.

   The dockable cursor window appears, showing the current position of the cursor on the x-axis and y-axis. As you move the cursors, the values in this window change.

   In the analog area of the plot (if any), both cursors are initially placed on the trace listed first in the trace legend. The corresponding trace symbol is outlined with a dashed line.

   In the digital area of the plot (if any), both cursors are initially placed on the trace named first along the y-axis. The corresponding trace name is outlined with a dashed line.

Moving Cursors

To move cursors along a trace using menu commands

1. Choose *Trace – Cursor*.
2. Next, choose *Peak*, *Trough*, *Slope*, *Min*, *Max*, *Point*, or *Search Commands*.

For more information about the cursor commands, see *PSpice A/D Help*.

To move cursors along a trace using the mouse

1. Use the right and left mouse buttons as described in the table.

Table 17-7 Mouse actions for cursor control

Click this...	To do this with the cursors...
cursor assignment	

@ 2023 All Rights Reserved.

Table 17-7 Mouse actions for cursor control, *continued*

Click this...	To do this with the cursors...
Left-click the analog trace symbol or digital trace name.	Associate the first cursor with the selected trace.
Right-click the analog trace symbol or digital trace name.	Associate the second cursor with the selected trace.
cursor movement	
Left-click in the display area.	Move the first cursor to the closest trace segment at the X position.
Right-click in the display area.	Move the second cursor to the closest trace segment at the X position.

Note: For a family of curves (such as from a nested DC sweep), you can use the mouse or the arrow keys to move the cursor to one of the other curves in the family. You can also click the desired curve.

To move cursors along a trace using the keyboard

1. Use key combinations as described in the table.

Table 17-8 Key combinations for cursor control

Us this key combination...	To do this with the cursors...
Ctrl+Left arrow key and *Ctrl*+Right arrow key	Change the trace associated with the first cursor.
Shift+*Ctrl*+Left arrow key and *Shift*+*Ctrl*+Right arrow key	Change the trace associated with the second cursor.
Left arrow key and Right arrow key	Move the first cursor along the trace.
Shift+Left arrow key and *Shift*+Right arrow key	Move the second cursor along the trace.
Home	Move the first cursor to the beginning of the trace.

@ 2023 All Rights Reserved.

Table 17-8 Key combinations for cursor control, *continued*

Us this key combination...	To do this with the cursors...
Shift+Home	Move the second cursor to the beginning of the trace.
End	Move the first cursor to the end of the trace.
Shift+End	Move the second cursor to the end of the trace.

Applying Cursors to a Different Trace

If you want to apply the cursors to a different trace, click the trace symbol in the plot legend for the trace you want to change to.

To freeze the cursor, do one of the following:

➡ Choose *Trace – Cursor – Freeze* to freeze the cursor locations on the current trace.

OR

1. Choose *Tools – Options*.

   The Probe Settings dialog box opens.

2. Select the *Cursor Settings* tab.
3. Select *Show non-dockable(old) cursor window*.

   The Probe Cursor window appears.

4. Click the Probe Cursor window (below the title bar) to freeze the cursor locations on the current trace.

For more information about the cursor commands, see *PSpice A/D Help*.

Changing Cursor Properties

1. Choose *Tools – Options*.

   The Probe Settings dialog box opens.

2. Specify the width and color for the cursors in the *Cursor Settings* tab.

@ 2023 All Rights Reserved.

The cursor color is affected by the background color of the PSpice A/D probe window. As a result, you might see the cursors with a different color in the probe window. However, the colors are updated in the Cursor Settings tab also. Close the Probe Settings dialog box and open it again to view the changed color.

Dumping and Copying Cursor Information

To dump cursor:

1. Open the dockable cursor window.
2. Right-click in this window and choose *Dump to CSV File*. The CSV file is created at the location of the `.dat` file.

To open the CSV file location, choose *File – Open File Location*. The CSV file can be opened using tools such as Microsoft Excel.

To copy cursor values:

1. Select the cursor values that you want to copy in the dockable cursor window.
2. Right-click and choose *Copy*.
3. Open any text editor and paste the content from clipboard.

Example: Using Cursors from Non-Dockable (old) Cursor Window

Following figure shows both cursors positioned on the Out signal in the digital area of a plot, and both cursors on the V(1) waveform in the analog area of the plot.

Cursor 1 is positioned on the first trough (dip) of the V(1) waveform. Cursor 2 is positioned on the second peak of the same waveform.

Note: To position a cursor on the next trough of a waveform, from the Trace menu, point to Cursor, then choose Trough. To position a cursor on the next peak of a waveform, from the Trace menu, point to Cursor, then choose Peak.

In the Probe Cursor window, cursor 1 and cursor 2 coordinates are displayed (A1 and A2, respectively) with their difference shown in the bottom line (dif). The logic state of the Out signal is also displayed to the right of the cursor coordinates.

The mouse buttons are also used to associate each cursor with a different trace by clicking appropriately on either the analog trace symbol in the legend or on the digital trace name (see Table 17-7 on page 545). These are outlined in the pattern corresponding to the associated

@ 2023 All Rights Reserved.

Cursors positioned on a trough and peak of V(1)

cursor's crosshair pattern. In the above example, right-clicking the V(2) symbol will associate cursor 2 with the V(2) waveform. The analog legend now appears as shown below.

The Probe Cursor window also updates the A2 coordinates to reflect the X and Y values corresponding to the V(2) waveform.

For more information about cursors, see *PSpice A/D Help*.

@ 2023 All Rights Reserved.

Tracking Digital Simulation Messages

PSpice A/D provides explanatory messages for errors that occur during a digital simulation with their corresponding waveforms. You can view messages from:

- the Simulation Message Summary dialog box, or
- the waveform display.

See Simulation Condition Messages for information on the message types that can be displayed by PSpice A/D.

Message Tracking from the Message Summary

A message summary is available for simulations where diagnostics have been logged to the waveform data file. You can display the message summary:

- When loading a waveform data file (click OK when the Simulation Errors dialog box appears).
- Anytime by choosing Simulation Messages from the View menu.

@ 2023 All Rights Reserved.

The Simulation Message Summary dialog box

The Simulation Message Summary dialog box lists message header information.

You can filter the messages displayed in the list by selecting a severity level from the Minimum Severity Level drop-down menu. Messages are categorized (in decreasing order of severity) as FATAL, SERIOUS, WARNING, or INFO (informational).

When you select a severity level, the Message Summary displays only those messages with the chosen severity *or higher*. By default, the minimum severity level displayed is SERIOUS.

Example: If you select WARNING as the minimum severity level, the Simulation Message Summary dialog box will display WARNING, SERIOUS, and FATAL messages.

To display waveforms associated with messages

1. In the Simulation Message Summary dialog box, double-click a message.

   For most message conditions, a Probe window appears that contains the waveforms associated with the simulation condition, along with detailed message text.

@ 2023 All Rights Reserved.

Persistent hazards

If a PERSISTENT HAZARD message is displayed, two plots appear in the figure, containing the following:

- the waveforms that initially caused the timing violation or hazard (lower plot)
- the primary outputs or internal state devices to which the condition has propagated (upper plot)

Figure 17-6 Waveform display for a persistent hazard.

Message Tracking from the Waveform

Trace segments with associated diagnostics are displayed in the foreground color specified in your `PSpice.ini` file. This color is different from those used for standard state transitions.

To display explanatory message text

➡ Double-click within the tagged region of a trace.

@ 2023 All Rights Reserved.

Trace Expressions

Traces are referred to by output variable names. Output variables are similar to the PSpice output variables specified in the Simulation Settings dialog box for noise, Monte Carlo, worst-case, transfer function, and Fourier analyses. However, there are additional alias forms that are valid for trace expressions. Both forms are discussed here.

To add traces using output variables

1. From the Trace menu, choose Add Trace to display the Add Traces dialog box.
2. Construct a trace expression using any combination of these controls:
   - In the Simulation Output Variables frame, click output variables.

     You can display a subset of the available simulation output variables by selecting or clearing the variable type check boxes in the Simulation Output Variables frame. Variable types not generated by the circuit simulation are dimmed.
   - In the Functions or Macros frame, select operators, functions, constants, or macros.
   - In the Trace Expression text box, type in or edit output variables, operators, functions, constants, or macros.

     For more information about trace expressions, see Analog Trace Expressions on page 561 and Digital Trace Expressions on page 563.
3. If you want to change the name of the trace expression as it displays in the Probe window, use the following syntax:

   trace expression;*display name*
4. Click OK.

Basic Output Variable Form

This form is representative of those used for specifying some PSpice A/D analyses.

<*output*>[*AC suffix*](<*name*>[,*name*])

Placeholder	Description
<*output*>	type of output quantity: V for voltage or I for current (digital values do not require a prefix)

@ 2023 All Rights Reserved.

Placeholder	Description
[*AC suffix*]*	quantity to be reported for an AC analysis.
<*name*>[,*name*]	specifies either the *net* or (+ *net*, - *net*) pair for which the voltage is to be reported, or the *device* for which a current is reported, where: ■ *net* specifies either the *net* or *pin id* (<*fully qualified device name*>:<*pin name*>) ■ *device name* specifies the fully qualified device name; A fully qualified device name consists of the full hierarchical path followed by the device's reference designator.

Output Variable Form for Device Terminals

This form can only be specified for trace expressions. The primary difference between this and the basic form is that the terminal symbol appears before the *net* or *device name* specification (whereas the basic form treats this as the *pin name* within the *pin id*).

<*output*>[*terminal*]*[*AC suffix*](<*name*>[,*name*])

Table 17-9

Placeholder	Description
<*output*>	type of output quantity: V for voltage, I for current, or N for noise (digital values do not require a prefix)
[*terminal*]*	one or more terminals for devices with more than two terminals;
[*AC suffix*]*	quantity to be reported for an AC analysis;
<*name*>[,<*name*>])	*net*, *net* pair, or fully qualified *device name;*

@ 2023 All Rights Reserved.

Table 17-10 on page 555 summarizes the valid output formats. The table provides examples of equivalent output variables. Note that some of the output variable formats are unique to trace expressions.

Table 17-10 Output variable formats

Format	Meaning
Voltage variables	
V[*ac*](< *+analog net* > [,< *-analog net* >])	Voltage between + and - *analog net ids*
V<*pin name*>[*ac*](< *device* >)	Voltage at *pin name* of a *device*
V< *x* >[*ac*](< *3 or 4-terminal device* >)	Voltage at non-grounded terminal *x* of a *3 or 4-terminal device*
V< *z* >[*ac*](< *transmission line device* >)	Voltage at end *z* of a *transmission line device (z is either A or B)*
Current variables	
I[*ac*](< *device* >)	Current into a *device*
I< *x* >[*ac*](< *3 or 4-terminal device* >)	Current into terminal *x* of a *3 or 4-terminal device*
I< *z* >[*ac*](< *transmission line device* >)	Current into end *z* of a *transmission line device (z is either A or B)*
Digital signal and bus variables	
< *digital net* >[;< *display name* >]	Digital state at *digital net* labeled as *display name*
{< *digital net* >*}[;< *display name* >] [;< *radix* >]	Digital bus labeled as *display name* and of specified *radix*
Sweep variables	
< *DC sweep variable* >	name of any variable used in the DC sweep analysis
FREQUENCY	AC analysis sweep variable
TIME	transient analysis sweep variable
Noise variables	

@ 2023 All Rights Reserved.

Table 17-10 Output variable formats, *continued*

Format	Meaning
V[db](ONOISE)	total RMS-summed noise at output net
V[db](INOISE)	total equivalent noise at input source
NTOT(ONOISE)	sum of all noise contributors in the circuit
N< *noise type* >(< *device name* >)	contribution from *noise type* of *device name* to the total output noise

@ 2023 All Rights Reserved.

Basic Form	An alias equivalent	Description
V(NET3,NET2)	(same)	voltage between analog nets labeled NET3 and NET2
V(C1:1)	V1(C1)	voltage at pin1 of capacitor C1
VP(Q2:B)	VBP(Q2)	phase of voltage at base of bipolar transistor Q2
V(T32:A)	VA(T32)	voltage at port A of transmission line T32
I(M1:D)	ID(M1)	current through drain of MOSFET device M1
QA	(same)	digital state at net QA
{IN1, IN2, IN3}; MYBUS;X	(same)	digital bus made of 3 digital nets (IN1, IN2, IN3) named MYBUS displayed in hexadecimal
VIN	(same)	voltage source named VIN
FREQUENCY	(same)	AC analysis sweep variable
NFID(M1)	(same)	flicker noise from MOSFET M1

Suffix	Output Variables
none	magnitude
DB	magnitude in decibels
G	group delay (-*d*PHASE/*d*FREQUENCY)
I	imaginary part
M	magnitude
P	phase in degrees
R	real part

Two-terminal Device Type	Device Type Letter
capacitor	C

@ 2023 All Rights Reserved.

Two-terminal Device Type	Device Type Letter
diode	D
voltage-controlled voltage source	E
current-controlled current source	F
voltage-controlled current source	G
current-controlled voltage source	H
independent current source	I
inductor	L
resistor	R
voltage-controlled switch	S
independent voltage source	V
current-controlled switch	W

@ 2023 All Rights Reserved.

Three & four-terminal device type	Device type letter	Terminal IDs
GaAs MOSFET	B	D (drain) G (gate) S (source)
Junction FET	J	D (drain) G (gate) S (source)
MOSFET	M	D (drain) G (gate) S (source) B (bulk, substrate)
Bipolar transistor	Q	C (collector) B (base) E (emitter) S (substrate)
transmission line	T	A (*near* side) B (*far* side)
IGBT	Z	C (collector) G (gate) E (emitter)

Table 17-11 Noise types by device type

Device type	Noise types	Meaning

@ 2023 All Rights Reserved.

Table 17-11 Noise types by device type

B (GaAsFET)	FID	flicker noise
	RD	thermal noise associated with RD
	RG	thermal noise associated with RG
	RS	thermal noise associated with RS
	SID	shot noise
	TOT	total noise
D (diode)	FID	flicker noise
	RS	thermal noise associated with RS
	SID	shot noise
	TOT	total noise
Digital Input	RHI	thermal noise associated with RHI
	RLO	thermal noise associated with RLO
	TOT	total noise
Digital Output	TOT	total noise
J (JFET)	FID	flicker noise
	RD	thermal noise associated with RD
	RG	thermal noise associated with RG
	RS	thermal noise associated with RS
	SID	shot noise
	TOT	total noise
M (MOSFET)	FID	flicker noise
	RB	thermal noise associated with RB
	RD	thermal noise associated with RD
	RG	thermal noise associated with RG
	RS	thermal noise associated with RS
	SID	shot noise
	TOT	total noise
Q (BJT)	FIB	flicker noise
	RB	thermal noise associated with RB
	RC	thermal noise associated with RC
	RE	thermal noise associated with RE
	SIB	shot noise associated with base current
	SIC	shot noise associated with collector current
	TOT	total noise
R (resistor)	TOT	total noise
Iswitch	TOT	total noise
Vswitch	TOT	total noise

@ 2023 All Rights Reserved.

Analog Trace Expressions

Trace Expression Aliases

Analog trace expressions vary from the output variables used in simulation analyses because analog net values can be specified by:

<*output variable*>[;*display name*]

as opposed to the <*output variable*> format used in analyses. With this format, the analog trace expression can be displayed in the analog legend with an optional alias.

Arithmetic Functions

Arithmetic expressions of analog output variables use the same operators as those used in simulation analyses (by means of part property definitions in Capture). You can also include intrinsic functions in expressions. The intrinsic functions available for trace expressions are similar to those available for PSpice math expressions, but with some differences, as shown in the table. A complete list of PSpice arithmetic functions can be found in the table.

Probe Function	Description	Available in PSpice A/D
ABS(x)	|x|	YES
SGN(x)	+1 (if x>0), 0 (if x=0), -1 (if x<0)	YES
SQRT(x)	$x^{1/2}$	YES
EXP(x)	e^x	YES
LOG(x)	*ln*(x)	YES
LOG10(x)	*log*(x)	YES
M(x)	magnitude of x	YES
P(x)	phase of x (degrees)	YES
R(x)	real part of x	YES
IMG(x)	imaginary part of x	YES
G(x)	group delay of x (seconds)	NO
PWR(x,y)	$\|x\|^y$	YES

@ 2023 All Rights Reserved.

Probe Function	Description	Available in PSpice A/D
SIN(x)	*sin*(x)	YES
COS(x)	*cos*(x)	YES
TAN(x)	*tan*(x)	YES
ATAN(x) ARCTAN(x)	tan^{-1} (x)	YES
d(x)	derivative of x with respect to the x-axis variable	YES
s(x)	integral of x over the range of the x-axis variable	YES
AVG(x)	running average of x over the range of the x-axis variable	NO
AVGX(x,d)	running average of x from X_axis_value(x)-d to X_axis_value(x)	NO
RMS(x)	running RMS average of x over the range of the x-axis variable	NO
DB(x)	magnitude in decibels of x	NO
MIN(x)	minimum of the real part of x	NO
MAX(x)	maximum of the real part of x	NO
ENVMIN(x,d)	Envelope of x.Valley lows selected have a minimum number of d consecutive datapoints.	NO
ENVMAX(x,d)	Envelope of x. Peaks selected have a minimum number of d consecutive datapoints.	NO

Note: For AC analysis, PSpice A/D uses complex arithmetic to evaluate trace expressions. If the result of the expression is complex, then its magnitude is displayed.

Rules for Numeric Values Suffixes

Explicit numeric values are entered in trace expressions in the same form as in simulation analyses (by means of part properties in Capture), with the following exceptions:

@ 2023 All Rights Reserved.

- Suffixes M and MEG are replaced with m (milli, 1E-3) and M (mega, 1E+6), respectively.
- MIL and mil are not supported.
- With the exception of the m and M scale suffixes, PSpice A/D is not case sensitive; therefore, upper and lower case characters are equivalent.

  Example: V(5) and v(5) are equivalent in trace expressions.

Unit suffixes are *only* used to label the axis; they never affect the numerical results. Therefore, it is always safe to leave off a unit suffix.

Example: The quantities 2e-3, 2mV, and .002v all have the same numerical value. For axis labeling purposes, PSpice A/D recognizes that the second and third forms are in volts, whereas the first is dimensionless.

PSpice A/D also knows that W=V·A, V=W/A, and A=W/V. So, if you add this trace:

V(5)*ID(M13)

the axis values are labeled with W.

The units to use for trace expressions are shown in the table.

Table 17-12 Output units for trace expressions

Symbol	Unit
V	volt
A	amps
W	watt
d	degree (of phase)
s	second
Hz	hertz

Digital Trace Expressions

Digital output variables in trace expressions vary from those used in simulation analyses as follows:

- Digital net values are specified by:

  <*digital net*>[;*display name*]

@ 2023 All Rights Reserved.

as opposed to the <*digital net*> format used for analyses. With this format, the digital signal can be displayed on the digital plot with an optional alias.

- The output from several digital nets can be collected into a single output of higher radix known as a bus.

  Example: You can request that four bus lines be displayed together as one hexadecimal digit. You can combine up to 32 digital signals into a bus.

  A bus is formed by enclosing a list of digital net names (separated by blanks or commas) within braces according to the format:

  {<*high-order net*> [*mid-order net*]* <*low-order net*>}

  The elements of the bus definition, taken left to right, specify the output values of the bus from high order to low order.

  Example: { Q2, Q1, Q0 } specifies a 3-bit bus whose high-order bit is the digital value at net Q2.

By definition, a *digital signal* is any digital net value or a logical expression involving digital nets. For the digital output variable formats described earlier, you can use a digital signal expression everywhere a net name is expected. You can also form buses into expressions using both logical and arithmetic operators.

As a result, the generalized form for defining a digital trace is:

<*digital trace expression*> [;*display name* [;*radix*]]

Exception: You can display your radix designation option with the digital trace expression by leaving the display name blank and using the following syntax:

```
digital trace expression;;radix
```

Table 17-13

Placeholder	Description
digital trace expression	expression of digital buses or digital signals.
display name	name that will be displayed on the screen; if no display name is specified, the actual trace expression is used; if a display name is given, it is available for use in subsequent trace definitions.

@ 2023 All Rights Reserved.

Table 17-13

Placeholder	Description
radix	applies only to bus expressions and denotes the radix in which the bus value is to be displayed; the radix is specified as: H or X hexadecimal (default) D decimal O octal B binary

The table presents the operators available for digital signal and bus expressions listed in order of precedence (high to low).

Table 17-14 Digital logical and arithmetic operators

Operator	Meaning
()	grouping
~	logical complement
*	multiplication (bus values only)
/	division (bus values only)
+	addition (bus values only)
-	subtraction (bus values only)
&	and
^	exclusive or
|	or

An arithmetic or logical operation between two bus operands results in a bus value that is as wide as is necessary to contain the result. Prior to the operation, if necessary, the shorter operand is extended to the width of the longer operand by zero-filling on the high-order end.

An arithmetic or logical operation between a bus operand and a signal operand results in a bus value. Prior to the operation, the signal is converted to a bus of width one, then extended if necessary.

@ 2023 All Rights Reserved.

You can use signal constants in signal expressions. Specify them as shown in the table.

Table 17-15 Signal constants for digital trace expressions

Signal Constant	Meaning
’0	low
’1	high
’F	falling
’R	rising
’X	unknown
’Z	high impedance

You can use bus constants in bus expressions. Specify them as strings of the form:

r'ddd

Table 17-16

This placeholder...	Means this...
r	case-insensitive radix specifier (x, h, d, o, or b)
ddd	string of digits appropriate to the specified radix

Table 17-17 Example notations for bus constants:

This notation...	Has this radix...
x'3FFFF	hexadecimal
h'5a	hexadecimal
d'79	decimal
o'177400	octal
b'100110	binary

@ 2023 All Rights Reserved.

18

Measurement Expressions

This section describes how to put together measurement expressions using the measurement definitions included with PSpice A/D.

Measurement expressions evaluate the characteristics of a waveform. A measurement expression is made by choosing the waveform and the waveform calculation you want to evaluate.

The waveform calculation is defined by a measurement definition such as rise time, bandpass bandwidth, minimum value, and maximum value.

For example, if you want to measure the risetime of your circuit output voltage, use the following expression:

```
Risetime(v(out))
```

Related Topics

- Creating Custom Measurement
- Measurement Definitions Included in PSpice A/D

Creating Measurement Expressions

Before you create a measurement expression, do the following:

1. Design a circuit in a design entry tool.
2. Set up a PSpice simulation:
   - Time Domain (transient)
   - DC Sweep
   - AC Sweep/Noise
3. Run the circuit in PSpice A/D.

@ 2023 All Rights Reserved.

Make sure the circuit is valid and you have the results you expect.

Composing a Measurement Expression

These steps show you how to create a measurement expression in PSpice.

First select a measurement definition, and then select output variables to measure. The two combined become a measurement expression.

Work in the *Simulation Results* view in PSpice A/D. In the side toolbar, click .

1. From the Trace menu in PSpice A/D, select *Measurements*.

   The *Measurements* dialog box appears.

2. Select the measurement definition you want to evaluate.

3. Click *Eval* (evaluate).

   The *Arguments for Measurement Evaluation* dialog box appears.

4. Click the *Name of trace to search* button.

   The *Traces for Measurement Arguments* dialog box appears.

   Note: You will only be using the Simulation Output Variables list on the left side. Ignore the Functions or Macros list.

5. Deselect the output types you do not need (if you want to simplify the list).

6. Click the output variable you want to evaluate.

   The output variable appears in the *Trace Expression* field.

7. Click *OK*.

   The *Arguments for Measurement Evaluation* dialog box reappears with the output variable you chose in the *Name of trace to search* field.

8. Click *OK*.

   Your new measurement expression is evaluated and displayed in the PSpice A/D window.

9. Click *OK* in the *Display Measurement Evaluation* pop-up box to continue working in PSpice A/D.

   Your new measurement expression is saved, but it no longer displays in the window. The only way to get another graphical display is to redo these steps.

@ 2023 All Rights Reserved.

You can see a numerical evaluation by following the next steps.

Viewing Results of Measurement Evaluations

To view the results of the measurement evaluations, do the following:

1. From the *View* menu in PSpice A/D, select *Measurement Results*.

   The *Measurement Results* table displays below the plot window.

2. Click the box in the *Evaluate* column.

   The PSpice A/D calculation for your measurement expression appears in the *Value* column.

Example of Measurement Expression

First you select a measurement definition, and then you select an output variable to measure. The two combined become a measurement expression.

Work in the Simulation Results view in PSpice A/D. In the side toolbar, click on .

1. From the *Trace* menu in PSpice A/D, select *Measurements*.

   The Measurements dialog box appears.

2. Select the measurement definition you want to evaluate.

@ 2023
All Rights Reserved.

3. Click the *Eval* button.

The *Arguments for Measurement Evaluation* dialog box appears.

4. Click the *Name of trace to search* button.

The *Traces for Measurement Arguments* dialog box appears.

Note: You will only be using the *Simulation Output Variables* list on the left side.

@ 2023
All Rights Reserved.

Ignore the *Functions or Macros* list.

@ 2023 All Rights Reserved.

5. Uncheck the output types you do not need (if you want to simplify the list).

6. Click on the output variable you want to evaluate.

   The output variable appears in the *Trace Expression* field.

7. Click *OK*.

@ 2023 All Rights Reserved.

The *Arguments for Measurement Evaluation* dialog box reappears with the output variable you chose in the *Name of trace to search* field.

8. Click *OK*.

Your new measurement expression is evaluated and displayed in the PSpice A/D window.

9. Click *OK* in the *Display Measurement Evaluation* pop-up box to continue working in PSpice A/D.

Your new measurement expression is saved, but does not display in the window. The only way to get another graphical display is to redo these steps. You can see a numerical evaluation by following the next steps.

10. Click *Close*.

@ 2023
All Rights Reserved.

Viewing Results of Measurement Evaluations

1. From the *View* menu, select *Measurement Results*.

   The *Measurement Results* table displays below the plot window.

2. Click the box in the *Evaluate* column.

   A check mark appears in the *Evaluate* column check box and the PSpice A/D calculation for your measurement expression appears in the *Value* column.

Measurement Definitions Included in PSpice A/D

Definition	Finds the. . .
Bandwidth	Bandwidth of a waveform (you choose dB level)
Bandwidth_Bandpass_3dB	Bandwidth (3dB level) of a waveform
Bandwidth_Bandpass_3dB_XRange	Bandwidth (3dB level) of a waveform over a specified X-range

@ 2023
All Rights Reserved.

Definition	**Finds the. . .**
CenterFrequency	Center frequency (dB level) of a waveform
CenterFrequency_XRange	Center frequency (dB level) of a waveform over a specified X-range
ConversionGain	Ratio of the maximum value of the first waveform to the maximum value of the second waveform
ConversionGain_XRange	Ratio of the maximum value of the first waveform to the maximum value of the second waveform over a specified X-range
Cutoff_Highpass_3dB	High pass bandwidth (for the given dB level)
Cutoff_Highpass_3dB_XRange	High pass bandwidth (for the given dB level)
Cutoff_Lowpass_3dB	Low pass bandwidth (for the given dB level)
Cutoff_Lowpass_3dB_XRange	Low pass bandwidth (for the given dB level) over a specified range
DutyCycle	Duty cycle of the first pulse/period
DutyCycle_XRange	Duty cycle of the first pulse/period over a range
Falltime_NoOvershoot	Falltime with no overshoot.
Falltime_StepResponse	Falltime of a negative-going step response curve
Falltime_StepResponse_XRange	Falltime of a negative-going step response curve over a specified range
GainMargin	Gain (dB level) at the first 180-degree out-of-phase mark
Max	Maximum value of the waveform
Max_XRange	Maximum value of the waveform within the specified range of X
Min	Minimum value of the waveform
Min_XRange	Minimum value of the waveform within the specified range of X
NthPeak	Value of a waveform at its nth peak
Overshoot	Overshoot of a step response curve
Overshoot_XRange	Overshoot of a step response curve over a specified range

@ 2023 All Rights Reserved.

Definition	**Finds the. . .**
Peak	Value of a waveform at its nth peak
Period	Period of a time domain signal
Period_XRange	Period of a time domain signal over a specified range
PhaseMargin	Phase margin
PowerDissipation_mW	Total power dissipation in milli-watts during the final period of time (can be used to calculate total power dissipation, if the first waveform is the integral of V(load)
Pulsewidth	Width of the first pulse
Pulsewidth_XRange	Width of the first pulse at a specified range
Q_Bandpass	Calculates Q (center frequency / bandwidth) of a bandpass response at the specified dB point
Q_Bandpass_XRange	Calculates Q (center frequency / bandwidth) of a bandpass response at the specified dB point and the specified range
Risetime_NoOvershoot	Risetime of a step response curve with no overshoot
Risetime_StepResponse	Risetime of a step response curve
Risetime_StepResponse_XRange	Risetime of a step response curve at a specified range
SettlingTime	Time from <begin_x> to the time it takes a step response to settle within a specified band
SettlingTime_XRange	Time from <begin_x> to the time it takes a step response to settle within a specified band and within a specified range. `SBAND_PERCENT` is the percentage of the specified number. For example, if you want the band percentage to be within `8%`, then use `8`. The tool internally converts it to `0.08`.
SlewRate_Fall	Slew rate of a negative-going step response curve
SlewRate_Fall_XRange	Slew rate of a negative-going step response curve over an X-range

@ 2023 All Rights Reserved.

Definition	Finds the. . .
SlewRate_Rise	Slew rate of a positive-going step response curve
SlewRate_Rise_XRange	Slew rate of a positive-going step response curve over an X-range
Swing_XRange	Difference between the maximum and minimum values of the waveform within the specified range
XatNthY	Value of X corresponding to the nth occurrence of the given Y_value, for the specified waveform
XatNthY_NegativeSlope	Value of X corresponding to the nth negative slope crossing of the given Y_value, for the specified waveform
XatNthY_PercentYRange	Value of X corresponding to the nth occurrence of the waveform crossing the given percentage of its full Y-axis range; specifically, nth occurrence of Y=Ymin+(Ymax-Ymin)*Y_pct/100
XatNthY_Positive Slope	Value of X corresponding to the nth positive slope crossing of the given Y_value, for the specified waveform
YatFirstX	Value of the waveform at the beginning of the X_value range
YatLastX	Value of the waveform at the end of the X_value range
YatX	Value of the waveform at the given X_value
YatX_PercentXRange	Value of the waveform at the given percentage of the X-axis range
ZeroCross	X-value where the Y-value first crosses zero
ZeroCross_XRange	X-value where the Y-value first crosses zero at the specified range
XatMinY	Value of X corresponding to the minimum Y value, for the specified waveform
XatMaxY	Value of X corresponding to the maximum Y value, for the specified waveform

@ 2023 All Rights Reserved.

Creating Custom Measurement

Measurement definitions establish rules to locate interesting points and compute values for a waveform. In order to do this, a measurement definition needs:

- A measurement definition name
- A marked point expression

  These are the calculations that compute the final point on the waveform.
- One or more search commands

  These commands specify how to search for the interesting points.

You need to keep the following things in mind:

- Decide what you want to measure.
- Examine the waveforms you have and choose which points on the waveform are needed to calculate the measured value.
- Compose the search commands to find and mark the desired points.
- Use the marked points in the Marked Point Expressions to calculate the final value for the waveform.
- Test the search commands and measurements.

An easy way to create a new definition is to use the PSpice *Trace* menu. Select Measurements to open the *Measurements* dialog box and then do the following:

1. Select the definition most similar to your needs
2. Click *Copy* and follow the prompts to rename and edit.

Writing a New Measurement Definition

1. From the PSpice A/D *Trace* menu, choose *Measurements*.

   The *Measurements* dialog box appears.
2. Click *New*.

   The *New Measurement* dialog box appears.
3. Type a name for the new measurement in the *New Measurement name* field.

   Make sure local file is selected.

@ 2023 All Rights Reserved.

This stores the new measurement in a .prb file local to the design.

4. Click OK.

   The Edit New Measurement dialog box appears.

5. Type in the marked expression.
6. Type in any comments you want.
7. Type in the search function.

Note: For syntax information, see Measurement Definition Syntax on page 582

Your new measurement definition is now listed in the Measurements dialog box.

Using the new measurement definition

Your new measurement definition is now listed in the Measurements dialog box.

Note: For steps on using a definition in a measurement expression to evaluate a trace, see Composing a Measurement Expression on page 568.

Let us take an example to create measurements.

1. From the PSpice A/D Trace menu, choose Measurements.

@ 2023 All Rights Reserved.

The Measurements dialog box appears.

2. Click New.

@ 2023 All Rights Reserved.

The *New Measurement* dialog box appears.

3. Type in a name in the *New Measurement name* field.
4. Make sure *use local file* is selected.

   This stores the new measurement in a .prb file local to the design.
5. Click *OK*.

   The *Edit New Measurement* dialog box appears.

6. Type in the marked expression:

```
point707(1) = y1
```


@ 2023
All Rights Reserved.

7. Type in the search function.

```
{
  1|Search forward level(70.7%, p) !1;
}
```

Note: The search function is enclosed within curly braces.

Always place a semicolon at the end of the last search function.

8. Type in any explanatory comments you want:

```
*
*#Desc#* Find the .707 value of the trace.
*
*#Arg1#* Name of trace to search
*
```

Similarly, you can create complex measurements. For example, you might want to determine the time at which a current wave crosses a specified value, say 2mA, and then determine the voltage value at this time. You can use the following syntax:

```
<Measurement Name>(1,2,Y_value,n_occur)=y2
*
*#Desc#* Find the value of voltage at the time at which
current crosses Y_value for the nth time.
*#Arg1#* Name of current trace to search
*#Arg2#* Name of voltage trace to search
*#Arg3#* Y value
*#Arg4#* nth occurrence
*
{
1| search forward for n_occur:level (Y_value,positive)
!1 ;
2| search forward Xvalue (x1) !2 ;
}
```

Using the new measurement definition

Your new measurement definition is now listed in the Measurements dialog box.

Measurement Definition Syntax

Check out the existing measurement definitions in PSpice A/D for syntax examples.

@ 2023 All Rights Reserved.

1. From the *Trace* menu in PSpice A/D, choose *Measurements*.

   The *Measurement* dialog box appears.

2. Highlight any example, and select *View* to examine the syntax.

@ 2023 All Rights Reserved.

Measurement definition: fill in the place holders

measurement_name (1, [2, …, n][, subarg1, subarg2, …, subargm]) = marked_point_expression

{

1| search_commands_and_marked_points_for_expression_1;

2| search_commands_and_marked_points_for_expression_2;

n| search_commands_and_marked_points_for_expression_n;

Measurement name syntax

Can contain any alphanumeric character (A-Z, 0-9) or underscore _ , up to 50 characters in length. The first character should be an upper or lower case letter.

Examples of valid function names: Bandwidth, CenterFreq, delay_time, DBlevel1.

Comments syntax

A comment line always starts with an asterisk. Special comment lines include the following examples:

```
*#Desc#*
```

The measurement description

```
*#Arg1#*
```

Description of an argument used in the measurement definition.

@ 2023 All Rights Reserved.

These comment lines will be used in dialog boxes, such as the *Arguments for Measurement Evaluation* box.

Marked Point Expressions syntax

A marked point expression calculates a single value, which is the value of the measurement, based on the X and Y coordinates of one or more marked points on a curve. The marked points are found by the search command.

All the arithmetic operators (+, -, *, /, ( )) and all the functions that apply to a single point (for example, ABS(), SGN(), SIN(), SQRT()) can be used in marked point expressions.

The result of the expression is one number (a real value).

Marked point expressions differ from a regular expression in the following ways:

- Marked point coordinate values (for example, x1, y3), are used instead of simulation output variables (v(4), ic(Q1)).
- Multiple-point functions such as d(), s(), AVG(), RMS(), MIN(), and MAX() cannot be used.
- Complex functions such as M(), P(), R(), IMG(), and G() cannot be used.
- One additional function called MPAVG can also be used. It is used to find the average Y value between 2 marked points. The format is:

  MPAVG(p1, p2,[<.fraction>])

@ 2023 All Rights Reserved.

where p1 and p2 are marked X points and fraction (expressed in decimal form) specifies the range. The range specified by [<.fraction>] is centered on the midpoint of the total range. The default value is 1.

Example:

The marked point expression

MPAVG (x1, x5, .2)

will find the halfway point between x1 and x5 and will calculate the average Y value based on the 20 percent of the range that is centered on the halfway point.

Search command syntax

search [direction] [/start_point/] [#consecutive_points#] [(range_x [,range_y])]

[for]

[repeat:] <condition>

Brackets indicate optional arguments.

You can use uppercase or lowercase characters, because searches are case independent.

[direction]

```
forward or backward
```

The direction of the search. Search commands can specify either a forward or reverse direction. The search begins at the origin of the curve.

[Forward] searches in the normal X expression direction, which may appear as backwards on the plot if the X axis has been reversed with a user-defined range.

Forward is the default direction.

```
[/start_point/]
```


@ 2023 All Rights Reserved.

The starting point to begin a search. The current point is the default.

Use this…	To start the search at this…
^	the first point in the search range
Begin	the first point in the search range
$	the last point in the search range
End	the last point in the search range
xn	a marked point number or an expression of marked points, for example, x1 (x1 - (x2 - x1) / 2)

```
[#consecutive points#]
```

Defines the number of consecutive points required for a condition to be met. Usage varies for individual conditions; the default is 1.

A peak is a data point with one neighboring data point on both sides that has a lower Y value than the data point.

If [#consecutive_points#] is 2 and <condition> is PEak, then the peak searched for is a data point with two neighboring data points on both sides with lower Y values than the marked data point.

```
[(range_x[,range_y])]
```

Specifies the range of values to confine the search.

The range can be specified as floating-point values, as a percent of the full range, as marked points, or as an expression of marked points. The default range is all points available.

Examples

This range…	Means this…
(1n,200n)	X range limited from 1e-9 to 200e-9, Y range defaults to full range

@ 2023 All Rights Reserved.

This range…	Means this…
(1.5,20e-9,0,1m)	both X and Y ranges are limited
(5m,1,10%,90%)	both X and Y ranges are limited
(0%,100%,1,3)	full X range, limited Y range
(,,1,3)	full X range, limited Y range
(,30n)	X range limited only on upper end

```
[for] [repeat:]
```

Specifies which occurrence of <condition> to find.

If repeat is greater than the number of found instances of <condition>, then the last <condition> found is used.

Example

The argument:

2:LEvel

would find the second level crossing.

```
<condition>
```

Must be exactly one of the following:

- LEvel(value[,posneg])
- SLope[(posneg)]
- PEak
- TRough
- MAx
- MIn
- POint
- XValue(value)

@ 2023 All Rights Reserved.

Each <condition> requires just the first 2 characters of the word. For example, you can shorten LEvel to LE.

If a <condition> is not found, then either the cursor is not moved or the measurement is not evaluated.

```
LEvel(value[,posneg])
```

[,posneg] Finds the next Y value crossing at the specified level. This can be between real data points, in which case an interpolated artificial point is created.

At least [#consecutive_points#]-1 points following the level crossing point must be on the same side of the level crossing for the first point to count as the level crossing.

[,posneg] can be Positive (P), Negative (P), or Both (B). The default is Both.

(value) can take any of the following forms:

Value form	Example
a floating number	1e5
	100n
	1
a percentage of full range	50%
a marked point	x1
	y1
or an expression of marked points	(x1-x2)/2
a value relative to startvalue	.-3 ⇒ startvalue -3
	.+3 ⇒ startvalue +3
a db value relative to startvalue	.-3db ⇒ 3db below startvalue
	.+3db ⇒ 3db above startvalue
a value relative to max or min	max-3 ⇒ maxrng -3
	min+3 ⇒ minrng +3
a db value relative to max or min	max-3db ⇒ 3db below maxrng
	min+3db ⇒ 3db above minrng

@ 2023 All Rights Reserved.

```
decimal point ( . )
```

A decimal point (.) represents the Y value of the last point found using a search on the current trace expression of the measurement. If this is the first search command, then it represents the Y value of the startpoint of the search.

```
SLope[(posneg)]
```

Finds the next maximum slope (positive or negative as specified) in the specified direction.

[(posneg)] refers to the slope going Positive (P), Negative (N), or Both (B). If more than the next [#consecutive_points#] points have zero or opposite slope, the Slope function does not look any further for the maximum slope.

Positive slope means increasing Y value for increasing indices of the X expression.

The point found is an artificial point halfway between the two data points defining the maximum slope.

The default [(posneg)] is Positive.

```
PEak
```

Finds the nearest peak. At least [#consecutive_points#] points on each side of the peak must have Y values less than the peak Y value.

```
TRough
```

Finds nearest negative peak. At least [#consecutive_points#] points on each side of the trough must have Y values greater than the trough Y value.

```
MAx
```

Finds the greatest Y value for all points in the specified X range. If more than one maximum exists (same Y values), then the nearest one is found.

MAx is not affected by [direction], [#consecutive_points#], or [repeat:].

```
MIn
```

Finds the minimum Y value for all points in the specified X range.

MIn is not affected by [direction], [#consecutive_points#], or [repeat:].

```
POint
```

Finds the next data point in the given direction.

```
XValue(value)
```

Finds the first point on the curve that has the specified X axis value.

@ 2023 All Rights Reserved.

The (value) is a floating-point value or percent of full range.

XValue is not affected by [direction], [#consecutive_points#], [(range_x [,range_y])], or [repeat:].

(value) can take any of the following forms:

Value form	Example
a floating number	1e5
	100n
	1
a percentage of full range	50%
a marked point	x1
	y1
or an expression of marked points	(x1+x2)/2
a value relative to startvalue	.-3 ⇒ startvalue -3
	.+3 ⇒ startvalue +3
a db value relative to startvalue	.-3db ⇒ 3db below startvalue
	.+3db ⇒ 3db above startvalue
a value relative to max or min	max-3 ⇒ maxrng -3
	min+3 ⇒ minrng +3

Syntax Example

The measurement definition is made up of:

- A measurement name
- A marked point expression
- One or more search commands enclosed within curly braces

This example also includes comments about:

- The measurement definition

@ 2023 All Rights Reserved.

- What arguments it expects when used
- A sample command line for its usage

Any line beginning with an asterisk is considered a comment line.

Risetime definition

```
Risetime(1) = x2-x1
*
*#Desc#* Find the difference between the X values
where the trace first
*#Desc#* crosses 10% and then 90% of its maximum
value with a positive
*#Desc#* slope.
*#Desc#* (i.e. Find the risetime of a step
response curve with no
*#Desc#* overshoot. If the signal has overshoot,
use GenRise().)
*
*#Arg1#* Name of trace to search
*
* Usage:
*Risetime(<trace name>)
*
   {
      1|Search forward level(10%, p) !1
        Search forward level(90%, p) !2;
   }
```

The name of the measurement is Risetime. Risetime will take 1 argument, a trace name (as seen from the comments).

The first search function searches forward (positive x direction) for the point on the trace where the waveform crosses the 10% point in a positive direction. That point's X and Y coordinates will be marked and saved as point 1.

The second search function searches forward in the positive direction for the point on the trace where the waveform crosses the 90% mark. That point's X and Y coordinates will be marked and saved as point 2.

The marked point expression is x2-x1. This means the measurement calculates the X value of point 2 minus the X value of point 1 and returns that number.

@ 2023 All Rights Reserved.

19

Other Output Options

This section describes the following:

- How to monitor the numerical values for voltages or currents on up to three nets in your circuit as the simulation proceeds.
- How to generate additional line plots and tables of voltage and current values to the PSpice A/D output file.
- How to save digital output states to a file that you can use later as input to another circuit.

Related Topics

- Viewing Analog Results in PSpice A/D Window
- Writing Additional Results to PSpice Output File
- Creating Test Vector Files
- Exporting Trace to CSV File

@ 2023 All Rights Reserved.

Viewing Analog Results in PSpice A/D Window

The schematic capture application provides a special WATCH1 part that lets you monitor voltage values for up to three nets in your schematic as a DC sweep, AC sweep or transient analysis proceeds. Results are displayed in PSpice A/D.

To display voltage values in the PSpice A/D window, do the following:

1. Place and connect a WATCH1 part (from the PSpice library SPECIAL) on an analog net.
2. In the schematic capture application such as Capture, double-click the WATCH1 part instance to display the Parts spreadsheet.

   In the schematic capture application such as Design Entry HDL, choose *Text - Attributes* and click the WATCH1 part to display the Attributes dialog box. :
3. In the ANALYSIS property column, type DC, AC, or TRAN (transient) for the type of analysis results you want to see.
4. Enter values in the LO and HI properties columns to define the lower and upper bounds, respectively, on the values you expect to see on this net.

   Note: If the results move outside of the specified bounds, PSpice A/D pauses the simulation so that you can investigate the behavior.
5. Repeat step 1 through 4 for up to two more WATCH1 instances.
6. Start the simulation.

@ 2023 All Rights Reserved.

For example, in the schematic fragment shown below, WATCH1 parts are connected to the Mid and Vcc nets. After starting the simulation, PSpice A/D displays voltages on the Mid and Vcc nets.

Writing Additional Results to PSpice Output File

The schematic capture application provide special parts that let you save additional simulation results to the PSpice output file as either line-printer plots or tables.

To view the PSpice output file after running a simulation:

➡ From the *Simulation* menu, choose *Examine Output*.

Generating Plots of Voltage and Current Values

You can generate voltage and current line-printer plots for any DC sweep, AC sweep, or transient analysis.

To generate plots of voltage or current to the output file, do the following:

@ 2023 All Rights Reserved.

1. Place and connect any of the following parts (from the PSpice library SPECIAL.OLB).

Table 19-1

Use this part...	To plot this...
VPLOT1	Voltage on the net that the part terminal is connected to.
VPLOT2	Voltage differential between the two nets that the part terminals are connected to.
IPLOT	Current through a net. (Insert this part in series, like a current meter.)

2. Double-click the part instance to display the Parts spreadsheet.
3. Click the property name for the analysis type that you want plotted: DC, AC, or TRAN.
4. In the columns for the analysis type that you want plotted (DC, AC or TRAN), type any non-blank value such as `Y`, `YES` or `1`.
5. If you selected the AC analysis type, enable an output format:
   a. Click the property name for one of the following output formats: MAG (magnitude), PHASE, REAL, IMAG (imaginary), or DB.
   b. Type any non-blank value such as Y, YES or 1.
   c. Repeat step a and b for as many AC output formats as you want to see plotted.

   Note: If you do not enable a format, PSpice A/D defaults to MAG.
6. Repeat step 2 through 5 for any additional analysis types you want plotted.

Note: If you do not enable an analysis type, PSpice A/D reports the transient results.

Generating Tables of Voltage and Current Values

You can generate tables of voltage and current values on nets for any DC sweep, AC sweep, or transient analysis.

To generate tables of voltage or current to the output file, do the following:

@ 2023 All Rights Reserved.

1. Place and connect any of the following parts (from the PSpice library SPECIAL.OLB).

Table 19-2

Use this part...	To tabulate this...
VPRINT1	Voltage on the net that the part terminal is connected to.
VPRINT2	Voltage differential between the two nets that the part terminals are connected to.
IPRINT	Current through a cut in the net. (Insert this part in series, like a current meter.)

2. Double-click the part instance to display the Parts spreadsheet.
3. Click the property name for the analysis type that you want tabulated: DC, AC, or TRAN.
4. In the columns for the analysis type that you want plotted (DC, AC or TRAN), type any non-blank value such as `Y`, `YES` or `1`.
5. If you selected the AC analysis type, enable an output format.
   a. Click the property name for one of the following output formats: MAG (magnitude), PHASE, REAL, IMAG (imaginary), or DB.
   b. Type any non-blank value such as Y, YES or 1.
   c. Repeat step a and b or as many AC output formats as you want to see tabulated.

   Note: If you do not enable a format, PSpice A/D defaults to MAG.
6. Repeat step 2 through 5 for any additional analysis types you want plotted.

Note: If you do not enable an analysis type, PSpice A/D reports the transient results.

Generating Tables of Digital State Changes

You can generate a table of digital state changes during a transient analysis for any net.

@ 2023 All Rights Reserved.

➡ To generate a table of digital state changes to the output file, place a PRINTDGTLCHG part (from the PSpice library SPECIAL.OLB) and connect it to the net that you are interested in.

@ 2023 All Rights Reserved.

Creating Test Vector Files

The design entry tools provide a special VECTOR part that lets you save digital simulation results to a vector file. Whenever any net with an attached VECTOR part changes state, PSpice A/D writes a line of *time-value* data to the vector file using the same format as the file stimulus device. This means that you can use the vector file to drive inputs for another simulation. To find out about vector file syntax, refer to the *PSpice A/D Reference Guide*.

To find out about setting up digital stimuli, see Defining a Digital Stimulus.

To generate a test vector file from your circuit, do the following:

1. Place a VECTORn part (from the PSpice library SPECIAL.OLB) and connect it to a wire or bus at the output of a digital part instance.
2. Double-click the VECTORn part instance to display the Parts spreadsheet.

3. Set the part properties as described below.

Property	Definition
POS	Column position in the file. Valid values range from 1 to 255.
FILE	Name of the vector file. If left blank, PSpice A/D creates a file named SCHEMATIC_NAME.VEC.
RADIX	If the VECTOR part is attached to a bus, the numerical notation for a bus. Valid values are B[inary], O[ctal], and H[ex].
BIT	If the VECTOR part is attached to a wire, the bit position within a single hex or octal digit.

@ 2023 All Rights Reserved.

Property	Definition
SIGNAMES	Names of the signals that appear in the header of the file. If left blank, PSpice A/D defaults to the following: ■ For a wire, the label (name) on the wire. ■ For a bus, a name derived from the position of each signal in the bus (from MSB to LSB).

Note: You can group separate signal values to form a hex or octal value by specifying the same POS property and defining RADIX as Hex or Octal. Define the bit position within the value using the BIT property.

4. Repeat step 1 through 3 for or as many test vectors as you want to create.

@ 2023 All Rights Reserved.

Exporting Trace to CSV File

To export a trace to a CSV file, do the following:

1. Choose *File - Export Comma Separated File (.csv file)*

   The Export to Comma Separated File appears.

2. In the *File* name box, specify the name and location of the CSV file.

3. Select the filters on the right of the dialog box, such as Analog, Voltages, Currents, and so on.

4. Select the output variables from the *Available Output Variables* list.

5. Click *Add* to add the selected variables.

   You can use the Add, Add All, Remove, and Remove All buttons to modify the list of output variables to export.

6. Select *Enable* to specify tolerance values for the variables.

7. Click *OK*.

The output variables are exported to the specified CSV file.

Tip

To open the CSV file, choose *File - Open File Location*.

@ 2023 All Rights Reserved.

@ 2023 All Rights Reserved.

20

Bias Point Display

Bias conditions are used to set up the correct operation of a circuit. If the results of a simulation are not what you expect to see, the bias conditions are the first parameters you should check.

After simulating with PSpice A/D, you can display bias point information on the schematic page in schematic capture application. Bias voltages are displayed next to their corresponding nets (nodes), bias currents are displayed next to their corresponding device pins and the bias powers are displayed next to their corresponding power sources. By seeing this data on your schematic, you can quickly focus in on potential problem areas of your design.

PSpice A/D calculates and saves the bias point current, voltages and powers for every simulation. The schematic capture application reads all of this information and can display currents on all pins that have models, voltages for every net in your circuit or powers for every power source in your circuit.

Note: Bias point information is available for all analysis types except DC Sweep. The bias point display feature does not work for DC Sweeps.

You can choose to display bias point information in the following ways:

- currents on all modeled pins throughout the design
- currents on only selected modeled pins
- voltages on all wires throughout the design
- voltages on only selected wires
- powers on all power sources throughout the design

@ 2023 All Rights Reserved.

- powers on only selected power sources

example of current bias point

example of voltage bias point

For currents on pins, a "+" value for the bias point means current is flowing into the pin, while a "-" value means current is flowing out of the pin. By default, only voltages are displayed. You can toggle the currents on or off by using the corresponding toolbar buttons or menu commands.

Storage and Update of Bias Information

Not all pins and nets will have bias information. Buses, for example, do not have bias information, since they are not distinct wires. Bias point data and the locations of the displayed values are saved as part of the schematic page. Distinct bias point information is saved for each simulation profile. The data is updated whenever you open a schematic page, when you resimulate, when you activate a different simulation profile, or when you change the display characteristics of the labels (such as color, font or precision).

If a page is reused (hierarchical subcircuits), the position of the bias point will be stored with the page and will be the same for all occurrences of that page, although the values will be different for each to accurately reflect the hierarchy of the circuit.

Bias Point Data for Multiple Analyses

If you have set up more than one analysis, you should keep in mind the following:

- The schematic capture application displays the bias information for the last analysis you ran. PSpice A/D runs analyses in this order: DC, AC, Transient. This means that if you perform a multi-run analysis like Parametric, Monte Carlo, Sensitivity/Worst-case or Temperature, you will see bias values for the last run only.

@ 2023 All Rights Reserved.

- A given voltage or current source can have a different DC value and initial transient value at TIME=0. This means that the initial transient bias calculation can be different from the DC (small-signal) bias point.
- For a Transient analysis, only the initial transient bias values are shown.

Displaying Bias Point Values

You have the option of enabling the display of bias information for all nets, pins and power sources in the circuit, or of enabling only selected nets, pins and power sources. You can also disable bias point display all together for situations such as large digital designs where the legibility of the schematic might be reduced.

To enable or disable the display of all bias information, do the following:

1. In the schematic tool application, from the *PSpice* menu, choose *Bias Points*. A submenu appears as shown in the following figure:

@ 2023
All Rights Reserved.

2. From the submenu, select *Enable*.

By clearing this option, you disable the bias display feature entirely. The presence of the check mark indicates that this option is enabled. (The default setting is to have this option enabled.)

To enable or disable the display of all bias points, do the following:

1. In the schematic capture application, from the *PSpice* menu, choose *Bias Points*. A submenu appears.
2. From the submenu, select:

   Enable Bias Current Display or click the toolbar button ( ).

   -or-

   Enable Bias Voltage Display or click the toolbar button ( ).

   -or-

   Enable Bias Power Display or click the toolbar button ( ).

By checking this option, you enable the corresponding bias display feature. The presence of the check mark indicates that this option is enabled. (The default setting is to have this option disabled.)

Note: Enabling bias display simply allows the feature to work, displaying whatever bias values were previously selected and toggled on. To learn how to toggle on or off just one specific bias display, click Toggling a specific bias display.

To turn on or off the display of selected bias points, do the following:

1. In the schematic capture application, click a pin, or hold the CTRL key and click on several pins to select them. You can also click on a part to select all of the pins for that part.
2. From the *PSpice* menu, choose Bias Points. A submenu appears.
3. From the submenu, select *Toggle Selected Bias Current*.

   -or-

   Click the toolbar button ( ) after selecting the pins.

The presence of the check mark indicates that this option is turned on. (The default setting is to have this option turned off.)

@ 2023 All Rights Reserved.

Note: You can also turn off selected bias points by pressing the DELETE key. If you use the DELETE key, the bias point labels will be restored to their default locations the next time you display them.

Important

Since the current bias point of a capacitor equals zero, you will not see current bias point displays for these devices. Similarly, for other two terminal devices such as resistors, the current bias point will only be displayed on the positive (+) pin, because the negative (-) pin does not have a bias point value to display.

To turn on display of bias points after turning off selected bias point displays, do the following:

1. From the *Edit* menu, choose *Select All*.
2. Click the toolbar button three times: Once to toggle on the bias displays that were previously turned off; once more to turn off all bias displays, and a third time to turn them all back on.

Controlling the Display of Bias Points

You can set the precision of the bias point value that is displayed, and you can change the color or font of the label. You can also define whether bias point data should be printed with the schematic. These settings are all defined using the *Bias Point Preferences* dialog box (see figure below).

To set the precision of the bias point values, do the following:

1. In the schematic capture application, from the *PSpice* menu, choose *Bias Points*. A submenu appears.

@ 2023
All Rights Reserved.

2. From the submenu, select *Preferences*. The *Bias Point Preferences* dialog box appears.
3. In the *Displayed Precision* text box, enter the number of decimals you wish to display for the bias point values.
4. Click *OK*.

This setting applies globally to all current, voltage and power bias points in your design. (The default setting is 4.)

To set the color of the bias point values:, do the following:

1. In schematic capture application, from the *PSpice* menu, choose *Bias Points*. A submenu appears.
2. From the submenu, select *Preferences*. The *Bias Point Preferences* dialog box appears.
3. Click the Color box for current, voltage or power and select the color you wish to display from the color palette.
4. Click *OK*.

These settings apply globally to all current and voltage bias points in your design.

To set the fonts for the bias point values, do the following:

1. In schematic capture application, from the PSpice menu, choose Bias Points. A submenu appears.
2. From the submenu, select Preferences. The Bias Point Preferences dialog box appears.
3. Click in the Font box for current, voltage or power and select the font type, style, and size you wish to display from the font dialog box.
4. Click OK.

These settings apply globally to all current, voltage and power bias points in your design. (The default setting is Arial 7.)

To print the bias point values, do the following:

1. In schematic capture application, from the *PSpice* menu, choose *Bias Points*. A submenu appears.
2. From the submenu, select *Preferences*. The *Bias Point Preferences* dialog box appears.

@ 2023 All Rights Reserved.

3. Click the Print check box for current, voltage or power. The bias point values will then print with the schematic page.
4. Click *OK*.

These settings apply globally to all current, voltage and power bias points in your design. The presence of the check mark indicates that this option is enabled. (The default setting is to have this option enabled.)

Moving Bias Points

Bias points have a default position near their corresponding wire or pin. You can select the bias point labels and relocate them to make the schematic page more legible. A dashed leader line extends from the label to the wire or pin it is associated with to indicate where the label belongs.

To move a bias point label, do the following:

1. Click the label you wish to move to select it.
2. While holding down the left mouse button, drag the label to the new location.

Once you move a label, the new location will be saved with the schematic page so the bias point will be displayed there again the next time you open that page.

Updating Bias Point Values

The bias point values displayed in schematic capture application reflect the data from the last simulation that was performed in PSpice A/D. In some cases, these may not be the most recent values, depending on when they were last updated. There are three ways to update the values.

To update the bias point values by resimulating, do the following:

➤ In schematic capture application, from the PSpice menu, choose Run.

By re-running the simulation in PSpice A/D, you update the bias points to the correct values.

To update the bias point values by changing profiles, do the following:

1. In Project Manager of your schematic capture application, under the File tab, double-click on the Simulation Profiles folder to see all the defined profiles for this project.
2. Right-click on the profile you want to make active. A shortcut menu appears.

@ 2023 All Rights Reserved.

3. From the shortcut menu, select *Make Active*.

By opening a new simulation profile, you update the bias points to reflect the correct values that were last calculated for that profile.

To update the bias point values by changing the display, do the following:

1. In schematic capture application, from the *PSpice* menu, choose *Bias Points*. A submenu appears.
2. From the submenu, select *Preferences*. The *Bias Point Preferences* dialog box appears.
3. Click *OK* without changing any settings.

By opening the *Preferences* dialog box, you force schematic capture application to update the bias points with the correct values.

Menu Commands for Bias Point Display

The following table explains the menu commands for bias point display. You can access these commands by selecting Bias Points from the *PSpice* menu in schematic capture application.

To see a map of these commands as they appear under the PSpice menu in schematic capture application, click *Bias Point* menu.

Menu command...	**Function...**
Enable	Enables or disables the bias point display capability.
Enable Bias Current Display	Enables or disables the display of current bias points for all modeled pins in the entire design.
Toggle Selected Bias Current	Turns on or off the display of current bias points for selected modeled pins.
Enable Bias Voltage Display	Enables or disables the display of voltage bias points for all wires in the entire design.
Toggle Selected Bias Voltage	Turns on or off the display of voltage bias points for selected wires.
Enable Bias Power Display	Enables or disables the display of power bias points for all wires in the entire design.

@ 2023 All Rights Reserved.

Menu command...	Function...
Toggle Selected Bias Power	Turns on or off the display of power bias points for selected wires.
Preferences	Provides controls for displaying the precision, color, font and printability of the bias point values.

Important

In order to access the PSpice menu in schematic capture application, you must be working with a project that can be simulated. You must create a project (not a design) and select the Analog or Mixed-Signal Circuit Wizard option in order to be able to simulate the design with PSpice A/D.

Toolbar Controls for Bias Point Display

The following table explains the control buttons in the Bias Point toolbar buttons on the PSpice toolbar.

The Bias Point toolbar buttons on the Pspice toolbar in schematic capture application is shown in the following figure:

Toolbar Icon	Description
Enable Bias Voltage Display	Enables or disables the display of voltage bias points for all wires in the entire design.
Toggle Selected Bias Voltage	Turns on or off the display of voltage bias points for selected wires.
Enable Bias Current Display	Enables or disables the display of current bias points for all modeled pins in the entire design.

@ 2023 All Rights Reserved.

Toolbar Icon	Description
Toggle Selected Bias Current	Turns on or off the display of current bias points for selected modeled pins.

Toggling a Specific Current Bias Display

To turn on the display of a specific current bias point:

1. In schematic capture application, from the Edit menu, choose Select All.
2. Click the Toggle Selected Bias Current button in the Bias Point toolbar ( ). This will either turn off all current bias values for every pin on the schematic page, or they will all be turned on, depending on the previous state. Click the button again, if necessary, to turn everything off. If there is no change in the display either way, be sure that Enable Bias Current Display is checked.
3. Click the mouse on an empty area of the schematic page to deselect everything.
4. Click the specific pin you want to display the current on, in order to select it.
5. Click the Toggle Selected Bias Current button in the Bias Point toolbar ( ).

You should now see only the current bias point value for the specific pin you are interested in. All other current bias points on the schematic page should be turned off.

Toggling a Specific Voltage Bias Display

To turn on the display of a specific voltage bias point:

1. In schematic capture application, from the Edit menu, choose Select All.
2. Click the Toggle Selected Bias Voltage button in the Bias Point toolbar ( )

   This will either turn off all voltage bias values for every net on the schematic page, or they will all be turned on, depending on the previous state. Click the button again, if necessary, to turn everything off. If there is no change in the display either way, be sure that Enable Bias Voltage Display is checked.
3. Click the mouse on an empty area of the schematic page to deselect everything.
4. Click the specific net you want to display the voltage on, in order to select it.

@ 2023 All Rights Reserved.

5. Click the Toggle Selected Bias Voltage button in the Bias Point toolbar ( ).

You should now see only the voltage bias point value for the specific pin you are interested in. All other voltage bias points on the schematic page should be turned off.

Toggling a Specific Power Bias Display

To turn on the display of a specific power bias point:

1. In schematic capture application, from the Edit menu, choose Select All.
2. Click the Toggle Selected Bias Power button in the Bias Point toolbar ( )

   This will either turn off all power bias values for every power source on the schematic page, or they will all be turned on, depending on the previous state. Click the button again, if necessary, to turn everything off. If there is no change in the display either way, be sure that Enable Bias Power Display is checked.
3. Click the mouse on an empty area of the schematic page to deselect everything.
4. Click the specific power source you want to display the voltage on, in order to select it.
5. Click the Toggle Selected Bias Voltage button in the Bias Point toolbar ( ).

You should now see only the voltage bias point value for the specific pin you are interested in. All other voltage bias points on the schematic page should be turned off.

@ 2023 All Rights Reserved.

@ 2023 All Rights Reserved.

A

Setting Initial State

This appendix includes the following sections:

- Save and Load Bias Points on page 616
- Setpoints on page 618
- Setting Initial Conditions on page 620

@ 2023 All Rights Reserved.

Save and Load Bias Points

Save Bias Point and Load Bias Point are used to save and restore bias point calculations in successive PSpice simulations.

Saving and restoring bias point calculations can decrease simulation times when large circuits are run multiple times and can aid convergence. If the circuit uses high gain components, or if the circuit's behavior is nonlinear around the bias point, this feature is not useful.

Save/Load Bias Point affect the following types of analyses:

- transient
- DC
- AC

Save Bias Point

Save bias point is a simulation control function that allows you to save the bias point data from one simulation for use as initial conditions in subsequent simulations. Once bias point data is saved to a file, you can use the load bias point function to use the data for another simulation.

To use save bias point

1. In the *Simulation Settings* dialog box, click the *Analysis* tab.
2. Under *Options*, select Save Bias Point.
3. Complete the *Save Bias Point* dialog box.
4. Click *OK*.

@ 2023 All Rights Reserved.

Load Bias Point

Load bias point is a simulation control function that allows you to set the bias point as an initial condition. A common reason for giving PSpice A/D initial conditions is to select one out of two or more stable operating points (set or reset for a flip-flop, for example).

To use load bias point

1. Run a simulation using the *Save Bias Point* option in the *Simulation Settings* dialog box.
2. Before running another simulation, click the *Analysis* tab in the *Simulation Settings* dialog box.
3. Under *Options*, select *Load Bias Point*.
4. Specify a bias point file to load. Include the path if the file is not located in your working directory, or use the Browse button to find the file.
5. Click *OK*.

@ 2023 All Rights Reserved.

Setpoints

Pseudo-components that specify initial conditions are called setpoints. These apply to the analog portion of your circuit.

Figure A-1 Setpoints.

The example in the figure includes the following:

IC1	a one-pin symbol that allows you to set the initial condition on a net for both small-signal and transient bias points
IC2	a two-pin symbol that allows you to set initial condition between two nets

Using IC symbols sets the initial conditions for the bias point only. It does not affect the DC sweep. If your circuit design contains both an IC symbol and a NODESET symbol for the same net, the NODESET symbol is ignored.

To specify the initial condition, edit the value of the VALUE property to the desired initial condition. PSpice A/D attaches a voltage source with a 0.0002 ohm series resistance to each net to which an IC symbol is connected. The voltages are clamped this way for the entire bias point calculation.

NODESET1 is a one-pin symbol which helps calculate the bias point by providing a initial guess for some net. NODESET2 is a two-pin symbol which helps calculate the bias point between two nets. Some or all of the circuit's nets may be given an initial guess. NODESET symbols are effective for the bias point (both small-signal and transient bias points) and for the first step of the DC sweep. It has no effect during the rest of the DC sweep or during the transient analysis itself.

Unlike the IC pseudo-components, NODESET provides only an initial guess for some net voltages. It does not clamp those nodes to the specified voltages. However, by providing an initial guess, NODESET symbols may be used to break the tie (in a flip-flop, for instance) and make it come up in a desired state. To guess at the bias point, enter the initial guess in the

@ 2023 All Rights Reserved.

Value text box for the VALUE property. PSpice A/D attaches a voltage source with a 0.0002 ohm series resistance to each net to which an IC symbol is connected.

These pseudo-components are netlisted as PSpice A/D .IC and .NODESET commands. Refer to these commands in the *PSpice A/D Reference Guide* for more information. Setpoints can be created for inductor currents and capacitor voltages using the IC property described in Setting Initial Conditions.

@ 2023 All Rights Reserved.

Setting Initial Conditions

The IC property allows initial conditions to be set on capacitors and inductors. These conditions are applied during all bias point calculations. However, if you select the Skip Initial Transient Solution check box in the Transient Analysis Setup dialog box, the bias point calculation is skipped and the simulation proceeds directly with transient analysis at TIME=0. Devices with the IC property defined start with the specified voltage or current value; however, all other such devices have an initial voltage or current of 0.

Note: Skipping the bias point calculation can make the transient analysis subject to convergence problems.

Applying an IC property for a capacitor has the same effect as applying one of the pseudo-components IC1 or IC2 across its nodes. PSpice A/D attaches a voltage source with a 0.002 ohm series resistance in parallel with the capacitor. The IC property allows the user to associate the initial condition with a device, while the IC1 and IC2 pseudo-components allow the association to be with a node or node pair. See Setpoints for more information about IC1 and IC2.

In the case of initial currents through inductors, the association is only with a device, and so there are no corresponding pseudo-components. The internal implementation is analogous to the capacitor. PSpice A/D attaches a current source with a 1 Gohm parallel resistance in series with the inductor.

@ 2023 All Rights Reserved.

B

Convergence and “time step too small errors”

This appendix discusses common errors and convergence problems in PSpice A/D.

- Introduction on page 622
- Diagnostics on page 625
- Bias Point (DC) Convergence on page 626
- DC Sweep Convergence on page 630
- Transient Convergence on page 631

@ 2023 All Rights Reserved.

Introduction

In order to calculate the bias point, DC sweep and transient analysis for analog devices PSpice A/D must solve a set of nonlinear equations which describe the circuit's behavior. This is accomplished by using an iterative technique—the Newton-Raphson algorithm—which starts by having an initial approximation to the solution and iteratively improves it until successive voltages and currents converge to the same result.

In a few cases PSpice A/D cannot find a solution to the nonlinear circuit equations. This is generally called a “convergence problem” because the symptom is that the Newton-Raphson repeating series cannot converge onto a consistent set of voltages and currents. The following discussion gives some background on the algorithms in PSpice A/D and some guidelines for avoiding convergence problems.

The transient analysis has the additional possibility of being unable to continue because the time step required becomes too small from something in the circuit moving too fast. This is also discussed below.

Note: The AC and noise analyses are linear and do not use an iterative algorithm, so the following discussion does not apply to them. Digital devices are evaluated using boolean algebra; this discussion does not apply to them either.

Newton-Raphson Requirements

The Newton-Raphson algorithm is *guaranteed to converge to a solution*. However, this guarantee has some conditions:

- The nonlinear equations must have a solution.
- The equations must be continuous.
- The algorithm needs the equations' derivatives.
- The initial approximation must be close enough to the solution.

Each of these can be taken in order. Remember that the PSpice A/D algorithms are used in computer hardware that has finite precision and finite dynamic range that produce these limits:

- Voltages and currents in PSpice A/D are limited to +/-1e10 volts and amps.
- Derivatives in PSpice A/D are limited to 1e14.
- The arithmetic used in PSpice A/D is double precision and has 15 digits of accuracy.

@ 2023 All Rights Reserved.

Is there a solution?

Yes, for any physically realistic circuit. However, it is not difficult to set up a circuit that has no solution within the limits of PSpice numerics.

Consider, for example, a voltage source of one megavolt connected to a resistor of one micro-ohm. This circuit does not have a solution within the dynamic range of currents (+/- 1e10 amps). Here is another example:

```
V1    1,    0    5v
D1    1,    0    DMOD
.MODEL      DMOD(IS=1e-16)
```

The problem here is that the diode model has no series resistance. To find out more about the diode equations, refer to the *Analog Devices* chapter in the *PSpice A/D Reference Guide*.

It can be shown that the current through a diode is:

$$I = IS*e^{V/(N*k*T/q)}$$

N defaults to one and k*T at room temperature is about .025 volts. So, in this example the current through the diode would be:

$$I = 1e\text{-}16*e^{200} = 7.22e70 \text{ amps}$$

This circuit also does not have a solution within the limits of the dynamic range of PSpice A/D. In general, be careful of components without limits built into them. Extra care is needed when using the expressions for controlled sources (such as for behavioral modeling). It is easy to write expressions with very large values.

Are the equations continuous?

The device equations built into PSpice A/D are continuous. The functions available for behavioral modeling are also continuous (there are several functions, such as int(x), which cannot be added because of this). So, for physically realistic circuits the equations can also be continuous. Exceptions that come are usually from exceeding the limits of the numerics in PSpice A/D. This example tries to approximate an ideal switch using the diode model:

```
.MODEL DMOD(IS=1e-16 N=1e-6)
```

The current through this diode is:

$$I = 1e\text{-}16*e^{V/(N*.025)} = 1e\text{-}16*e^{V/25e\text{-}9}$$

@ 2023 All Rights Reserved.

Because the denominator in the exponential is so small, the current I is essentially zero for V < 0 and almost infinite for V > 0. Even if there are external components that limit the current, the “knee” of the diode's I-V curve is so sharp that it is almost a discontinuity.

Note: Avoid unrealistic model parameters. Behavioral modeling expressions need extra care.

Are the derivatives correct?

The device equations built into PSpice A/D include the derivatives, and these are correct. Depending on the device, the physical meaning of the derivatives can be small-signal conductance, transconductance or gain.

Unrealistic model parameters can exceed the limit of `1e14`, but it requires some effort. The main thing to look at is the behavioral modeling expressions, especially those having denominators.

Discontinuities in models characteristics and their derivatives cause:

1. Ambiguity in calculation of derivatives at point of discontinuity.
2. Conductance calculated in nth iteration cannot become a good guess for next iteration.
3. Sudden switching of operating regions (example diode switching from off to on) and hence false convergence.

Note: Transient analysis convergence failures are usually due to model discontinuities or unrealistic circuit, source, or parasitic modeling.

Is the initial approximation close enough?

Newton-Raphson is guaranteed to converge only if the analysis is started close to the answer. Also, there is no measurement that can tell how close is close enough.

PSpice A/D gets around this by making heavy use of continuity. Each analysis starts from a known solution and uses a variable step size to find the next solution. If the next solution does not converge PSpice A/D reduces the step size, falls back and tries again.

Incorrect initial estimates can cause convergence failure or even false convergence. Consider following scenarios:

- Power electronic circuits may NOT require tight current/voltage tolerances. Setting the value of ABSTOL to 1u will help in the case of circuits that have currents that are larger than several amps.

@ 2023 All Rights Reserved.

- Unless the circuit conducts kilo-Amperes of current, however, setting ABSTOL to a value that is greater than 1u will cause more convergence problems than it will solve.
- PSpice A/D does not always converge when relaxed tolerances are used. For example, setting the tolerance option, RELTOL, to a value which is greater than .01 can actually cause convergence problems
- Setting GMIN to a value between 1n and 10n will often solve convergence problems.
- Setting GMIN to a value, which is greater than 10n, may cause convergence problems.

Diagnostics

If PSpice A/D encounters a convergence problem it inserts into the output file a message that looks like the following.

```
ERROR -- Convergence problem in transient analysis at Time =  7.920E-03

Time step =  47.69E-15, minimum allowable step size =  300.0E-15

    These voltages failed to converge:

        V(x2.23)  =         1230.23 / -68.4137
        V(x2.25)  =        -1211.94 / 86.6888

    These supply currents failed to converge:

        I(X2.L1)   =        -36.6259 / 2.25682
        I(X2.L2)   =        -36.5838 / 2.29898

    These devices failed to converge:

        X2.DCR3   X2.DCR4   x2.ktr    X2.Q1     X2.Q2

    Last node voltages tried were:

NODE      VOLTAGE    NODE      VOLTAGE  NODE      VOLTAGE     NODE         VOLTAGE

(    1)   25.2000    (    3)   4.0000   (    4)   0.0000      (    6)      25.2030

(x2.23)   1230.2000  (X2.24)   9.1441   (x2.25)   -1211.9000  (X2.26)      256.9700

(X2.28)   -206.6100  (X2.29)   75.4870  (X2.30)   -25.0780    (X2.31)      26.2810

(X3.34)   1.771E-06  (X3.35)   1.0881   (X3.36)   .4279       (X2.XU1.6)   1.2636
```

The message always includes the banner (`ERROR -- convergence problem ...`) and the trailer (`Last node voltages tried were ...`). It cannot include all three of the middle blocks.

The `Last node voltages tried...` trailer shows the voltages tried at the last Newton-Raphson iteration. If any of the nodes have unreasonable large values this is a clue

@ 2023 All Rights Reserved.

that these nodes are related to the problem. “These voltages failed to converge” lists the specific nodes which did not settle onto consistent values. It also shows their values for the last two iterations. “These supply currents failed converge” does the same for currents through voltage sources and inductors. If any of the listed numbers are +/- 1e10 then that is an indication that the value is being clipped from an unreasonable value. Finally, “These devices failed to converge” shows devices whose terminal currents or core fluxes did not settle onto consistent values.

The message gives a clue as to the part of the circuit which is causing the problem. Looking at those devices and/or nodes for the problems discussed above is recommended.

Bias Point (DC) Convergence

The hardest part of the whole process is getting started, that is, finding the bias point. PSpice A/D first tries with the power supplies set to 100%. A solution is not guaranteed, but most of the time the PSpice algorithm finds one. If not, then the power supplies are cut back to almost zero. They are cut to a level small enough that *all nonlinearities are turned off*. When the circuit is linear a solution can be found (very near zero, of course). Then, PSpice A/D works its way back up to 100% power supplies using a variable step size.

Once a bias point is found, the transient analysis can be run. It starts from a known solution (the bias point) and steps forward in time. The step size is variable and is reduced as needed to find further solutions.

In case the circuit fails to converge, you should first check the circuit topology and connectivity, followed by modeling of circuit components, and finally check if Pspice options are set properly.

Checking circuit topology and connectivity

- Make sure that all of the circuit connections are valid. Check for incorrect node numbering or dangling nodes. Also, verify component polarity.
- Check for syntax mistakes. Make sure that you used the correct PSPICE units (i.e. MEG for 1E6, not M, which means mili in simulations).
- PSpice A/D checks for the following conditions and provides messages if they occur:
  - “Floating node" or "No DC path to ground" messages:
    - Make sure that there's a DC path from every node to ground.
    - Make sure that there are at least two connections at every node.

@ 2023 All Rights Reserved.

- ❍ Make sure that capacitors and/or current sources are not connected in series.
- ❍ Make sure that no (groups of) nodes are isolated from ground by current sources and/or capacitors.

❑ "Voltage source or inductor loop" message:

- ❍ Make sure that there are no loops of only inductors and/or voltage sources.

■ Place the ground (node 0) somewhere in the circuit. Be careful when you use floating grounds (e.g. chassis ground); you may need to connect a large resistor from the floating node to ground. All nodes will be reported as floating if "0 ground" is not used.

■ Make sure that voltage/current generators use realistic values, and verify that the syntax is correct.

■ Make sure that dependent source gains are correct, and that E/G element expressions are reasonable. Verify that division by zero or LOG(0) cannot occur.

■ Make sure that there are no unrealistic model parameters; especially if you have manually entered the model into the netlist.

■ Avoid using digital components, unless really necessary. Initialize the nodes with valid digital value to ensure the state is not ambiguous.

Modeling Circuit Components

■ Semiconductors

The first consideration for semiconductors is to avoid physically unrealistic model parameters. As PSpice A/D steps the power supplies up, it has to step carefully through the turn on transition for each device. In the diode example above, for the setting N=1e-6, the knee of the I-V curve would be too sharp for PSpice A/D to maintain its continuity within the power supply step size limit of 1e-6.

■ Behavioral modeling expressions

Range limits: Voltages and currents in PSpice A/D are limited to the range +/- 1e10. Care must be taken that the output of expressions falls within this range. This is especially important when one is building an electrical analog of a mechanical, hydraulic or other type of system.

Source limits: Another consideration is that the controlled sources must turn off when the supplies are almost 0 (.001%). There is special code in PSpice A/D which "squelches" the controlled sources in a continuous way near 0 supplies. However, care should still be taken using expressions that have denominators. Take, for example, a constant power load:

@ 2023 All Rights Reserved.

```
GLOAD 3, 5 VALUE = {2Watts/V(3,5)}
```

The first repeating series starts with V(3,5) = 0 and the current through GLOAD would be infinite (actually, the code in PSpice A/D which does the division clips the result to a finite value). The "squelching" code is required to be a smooth and well-behaved function.

Note: The "squelching" code cannot be "strong" enough to suppress dividing by 0.

The result is that GLOAD does not turn off near 0 power supplies. A better way is described in the application note Modeling Constant Power Loads. The "squelching" code is sufficient for turning off all expressions except those having denominators. In general, though, it is good practice to constrain expressions having the LIMIT function to keep results within physically realistic bounds.

Example: A first approximation to an OPAMP that has an open loop gain of 100,000 is:

```
VOPAMP 3, 5 VALUE = {V(in+,in-)*1e5}
```

This has the undesirable property that there is no limit on the output.

A better expression is:

```
VOPAMP 3, 5 VALUE = + {LIMIT(V(in+,in-)*1e5,15v,-15v}
```

where the output is limited to +/- 15 volts.

- Unguarded p-n junctions

  A second consideration is to avoid "unguarded" p-n junctions (no series resistance).

- No leakage resistance

  A third consideration is to avoid situations, which could have an ideal current source pushing current into a reverse-biased p-n junction without a shunt resistance. p-n junctions in PSpice A/D have (almost) no leakage resistance and would cause the junction's voltage to go beyond 1e10 volts.

- Switches

  PSpice A/D switches have gain in their transition region. If several are cascaded then the cumulative gain can easily exceed the derivative limit of 1e14. This can happen when modeling simple logic gates using totem-pole switches and there are several gates cascaded in series. Usually a cascade of two switches works but three or more can cause trouble.

PSpice A/D Options

- Increase ITL1 to 400 in the .OPTIONS statement.

  ```
  Example: .OPTIONS ITL1=400
  ```


@ 2023 All Rights Reserved.

This increases the number of DC iterations that PSpice A/D will perform before it gives up. In all but the most complex circuits, further increases in ITL1 won't typically aid convergence.

- Add .NODESETs

  `Example: .NODESET V(6)=0`

  Use NODESETs to set node voltages to the nearest reasonable guess at their DC values, particularly at nodes that are isolated by high impedances, and at nodes that are inputs to high gain devices. NODESETs do not "fix" the voltages at these nodes. They hold these voltages at the specified value while the rest of the circuit converges to a reasonably stable point, and then "releases" these voltages for a few more iterations to find the final, complete solution. Removing these voltages from the initial iterations, when voltages and currents are varying widely helps PSpice A/D achieve convergence.

- STEPGMIN

  Specifying the circuit analysis option STEPGMIN enables this (either using .OPTION STEPGMIN in the netlist, or by making the appropriate choice from the PSpice/Edit Simulation Profile… menu command, Options tab). When enabled, the GMIN stepping algorithm is applied after the circuit fails to converge with the power supplies at 100 percent, and if GMIN stepping also fails, the supplies are then cut back to almost zero and then stepped up.

  GMIN stepping attempts to find a solution by starting the repeating cycle with a large value of GMIN, initially 1.0e10 times the nominal value. If a solution is found at this setting it then reduces GMIN by a factor of 10, and tries again. This continues until either GMIN is back to the nominal value, or until PSpice A/D fails to converge at one of the GMIN values on the way. In the latter case, GMIN is restored to the nominal value and the power supplies are stepped.

- Power supply stepping

  As previously discussed, PSpice A/D uses a proprietary algorithm which finds a continuous path from zero power supplies levels to 100%. It starts at almost zero (.001%) power supplies levels and works its way back up to the 100% levels. The minimum step size is 1e-6 (.0001%). The first repeating series of the first step starts at zero for all voltages. So modeling expressions, especially those having denominators that include voltages should be checked carefully.

- Set PREORDER in Simulation Profiles options

  This is important while editing schematic for marginally convergent circuits. Setting PREORDER reduces dependency on the netlisting order thereby ensuring that the non convergence error does not occur because of the change in the netlisting order.

@ 2023 All Rights Reserved.

DC Sweep Convergence

The DC sweep uses a hybrid approach. It uses the bias point algorithm (varying the power supplies) to get started. For subsequent steps it uses the previous solution as the initial approximation. The sweep step is not variable, however. If a solution cannot be found at a step then the bias point algorithm is used for that step.

The whole process relies heavily on continuity. It also requires that the circuit be linear when the supplies are turned off.

Circuit topology and connectivity

This is same as in DC analysis. See Checking circuit topology and connectivity on page 626.

Modeling Checks

This is same as in DC analysis. Modeling Circuit Components on page 627.

PSpice A/D Options

- Set ITL2=100 in the .OPTIONS statement.

  `Example: .OPTIONS ITL2=100`

  This increases the number of DC iterations that PSpice A/D will attempt before it gives up.

- Increase or decrease the step values, which are used in the .DC sweep.

  Example:

  `.DC VCC 0 1 .1` becomes `.DC VCC 0 1 .01`

  Discontinuities in the PSpice models can cause convergence problems. The use of larger steps may help to bypass the discontinuities, while the use of smaller steps may help PSpice A/D find the intermediate answers, which will be used to find the point, which doesn't converge. In some cases, smaller steps can improve convergence, because they help PSpice A/D find a "path" from the valid DC solution at one point to the valid solution at the next.

- Do not use the DC sweep analysis.

  Example:

```
.DC VCC 0 5 .1
VCC 1 0
```


@ 2023 All Rights Reserved.

becomes

```
.TRAN .01 1
VCC 1 0 PULSE 0 5 0 1
```

In many cases, it is preferable to use the transient analysis to ramp the appropriate voltage and/or current sources. The transient analysis tends to be more robust, and is sometimes faster.

Transient Convergence

The transient analysis starts using a known solution - the bias point. It then uses the most recent solution as the first guess for each new time point. If necessary, the time step is cut back to keep the new time point close enough that the first guess allows the Newton-Raphson repeating series to converge. The time step is also adjusted to keep the integration of charges and fluxes accurate enough.

In theory the same considerations which were noted for the bias point calculation apply to the transient analysis. However, in practice they show up during the bias point calculation first and, hence, are corrected before a transient analysis is run.

The transient analysis can fail to complete if the time step gets too small. This can have two different effects:

1. The Newton-Raphson iterations would not converge even for the smallest time step size, or
2. Something in the circuit is moving faster than can be accommodated by the minimum step size.

The message PSpice A/D puts into the output file specifies which condition occurred.

Circuit topology and connectivity

- Avoid using digital components, unless really necessary. Initialize the nodes with valid digital value to ensure no ambiguous state. These can cause time-step issues (time-step may unnecessary go too small) and hence transient convergence issue.
- Use RC snubbers around diodes.
- Add Capacitance for all semiconductor junctions (if no specific value is known: CJO=3pF for diodes, CJC & CJE=5pF for BJTs, CGS and CGD=5pF for JFETs and GaAsFETs, CGDO & CGSO=5pF for MOSFETs if no specific value is known).
- Add realistic circuit and element parasitics.

@ 2023 All Rights Reserved.

- Look for waveforms that transition vertically (up or down) at the point during which the analysis halts. These are the key nodes, which should be examined for problems.
- Increase the rise/fall times of the PULSE sources; e.g. from 1f to 1u.

  Example:

  ```
  VCC 1 0 PULSE 0 1 0 1f 1f
  ```

  becomes

  ```
  VCC 1 0 PULSE 0 1 0 1U 1U
  ```

  An effort should be made to smooth strong non-linearities. The pulse times should be realistic, not ideal. If no rise or fall time values are given, or if 0 is specified, the rise and fall times will be set to the TSTEP value in the .TRAN statement (set in the Output File Options of the Time Domain (Transient) analysis settings in the simulation profile.

- Ensure that there is no unreasonably large capacitor or inductor

  If the transient analysis fails at the first time point then usually there is an unreasonably large capacitor or inductor. Usually this is due to a typographical error. Consider the following capacitor:

  ```
  C 1 3, 0 1Ouf
  ```

"`1O`" (has the letter O) should have been "`10`." This capacitor has a value of one farad, not 10 microfarads. An easy way to catch these is to use the LIST option (on the .OPTIONS command).

LIST

The LIST option can echo back all the devices into the output file *that have their values in scientific notation*.

That makes it easy to spot any unusual values. This kind of problem does not show up during the bias point calculation because capacitors and inductors do not participate in the bias point.

Similar comments apply to the parasitic capacitance parameters in transistor (and diode) models. These are normally echoed to the output file (the NOMOD option suppresses the echo but the default is to echo). As in the LIST output, the model parameters are echoed in scientific notation making it easy to spot unusual values. A further diagnostic is to ask for the detailed operating bias point (.TRAN/OP) information.

@ 2023 All Rights Reserved.

.TRAN/OP

This lists the small-signal parameters for each semiconductor device including the calculated parasitic capacitances.

Realistically Model Circuit; add parasitics, especially stray/junction capacitance

The idea here is to smooth any strong non-linearties or discontinuities. This may be accomplished via the addition of capacitance to various nodes and verifying that all semiconductor junctions have capacitance. Other tips include:

- Bipolar transistors substrate junction

  The UC Berkeley SPICE contains an unfortunate convention for the substrate node of bipolar transistors. The collector-substrate p-n junction has no DC component. If the capacitance model parameters are specified (e.g., CJS) then the junction has (voltage-dependent) capacitance but no DC current. This can lead to a sneaky problem: if the junction is inadvertently forward-biased it can create a very large capacitance. The capacitance goes as a power of the junction voltage. Normal junctions cannot sustain much forward voltage because a large current flows. The collector-substrate junction is an exception because it has no DC current. If this happens it usually shows up at the first time step. It can be spotted turning on the detailed operating point information (.TRAN/OP) and looking at the calculated value of CJS for bipolar transistors. The whole problem can be prevented by using the PSpice model parameter ISS. This parameter "turns on" the DC current for the substrate junction.

- Parasitic capacitances

  It is important that switching times be nonzero. This is assured if devices have parasitic capacitances. The semiconductor model libraries in PSpice A/D have such capacitances. If switches and/or controlled sources are used, then care should be taken to assure that no sections of circuitry can try to switch in zero time. In practice this means that if any positive feedback loops exist (such as a Schmidt trigger built out of switches) then such loops should include capacitances.

  Another way of saying all this is that during transient analysis the circuit equations must be continuous over time (just as during the bias point calculation the equations must be continuous with the power supply level).

- Inductors and transformers

  While the impedance of capacitors gets lower at high frequencies (and small time steps) the impedance of inductors gets higher.

  Note: The inductors in PSpice A/D have an infinite bandwidth.

@ 2023 All Rights Reserved.

Real inductors have a finite bandwidth due to eddy current losses and/or skin effect. At high frequencies the effective inductance drops.

Another way to say this is that physical inductors have a frequency at which their Q begins to roll off. The inductors in PSpice A/D have no such limit. This can lead to very fast spikes as transistors (and diodes) connected to inductors turn on and off. The fast spikes, in turn, can force PSpice A/D to take unrealistically small time steps.

- It is recommended that all inductors have a parallel resistor (series resistance is good for modeling DC effects but does not limit the inductor's bandwidth). The parallel resistor gives a good model for eddy current loss and limits the bandwidth of the inductor. The size of resistor should be set to be equal to the inductor's impedance at the frequency at which its Q begins to roll off. The value of this resistor can be calculated using the following formula:

  $$R = 2 \times \Pi \times f \times L$$

  where f is the roll-off frequency.

  Adding parallel resistors limits the inductor impedance at high frequencies.

  Example:

  A common one milli-henry iron core inductor begins to roll off at no less than 100KHz. A good resistor value to use in parallel is then R = 2*p*100e3*.001 = 628 ohms. Below the roll-off frequency the inductor dominates; above it the resistor does. This keeps the width of spikes from becoming unreasonably narrow.

PSpice A/D options

TIME, the simulation time during transient analysis, is a double precision variable which gives it about 15 digits of accuracy. The dynamic range is set to be 15 digits minus the number of digits of accuracy required by RELTOL. For a default value of RELTOL = .001 (.1% or 3 digits) this gives 15-3 = 12 digits. This means that the minimum time step is the overall run time (TSTOP) divided by 1e12. The dynamic range is large but finite.

It is possible to exceed this dynamic range in some circuits. Consider, for example, a timer circuit which charges up a 100uF capacitor to provide a delay of 100 seconds. At a certain threshold a comparator turns on a power MOSFET. The overall simulation time is 100 seconds. For default RELTOL this gives us a minimum time step of 100 picoseconds. If the comparator and other circuitry has portions that switch in a nanosecond then PSpice A/D needs steps of less than 100 picoseconds to calculate the transition accurately.

- Set RELTOL=.01 in the .OPTIONS statement.

  Example:

  ```
  .OPTIONS RELTOL=.01
  ```


@ 2023 All Rights Reserved.

This option is encouraged for most simulations, since the reduction of Reltol can increase the simulation speed by 10 to 50%. Only a minor loss in accuracy usually results. A useful recommendation is to set Reltol to .01 for initial simulations, and then reset it to its default value of .001 when you have the simulation running the way you like it and a more accurate answer is required. Setting Reltol to a value less than .001 is generally not required.

- Reduce the accuracy of ABSTOL/VNTOL if current/voltage levels allow it.

  Example:

  ```
  .OPTION ABSTOL=1N VNTOL=1M
  ```

  Abstol and Vntol should be set to about 8 orders of magnitude below the level of the maximum voltage and current. The default values are Abstol=1pA and Vntol=1uV. These values are generally associated with IC designs.

- Increase ITL4, but to no more than 100, in the .OPTIONS statement.

  Example:

  ```
  .OPTIONS ITL4=40
  ```

  This increases the number of transient iterations that PSpice A/D will attempt at each time point before it gives up. This is particularly effective at solving convergence problems when the simulation needs to cover a long time period, and fast transitions occur within the circuit during that time. Values greater than 100 won't usually bring convergence; unnecessarily large values can cause.

- Skipping the bias point

  The SKIPBP option for the transient analysis skips the bias point calculation. In this case the transient analysis has no known solution to start from and, therefore, is not assured of converging at the first time point. Because of this, its use is not recommended. Its inclusion in PSpice A/D is to maintain compatibility with UC Berkeley SPICE. SKIPBP has the same meaning as UIC in Berkeley SPICE. UIC is not needed in order to specify initial conditions.

  It should be used as a last resort if there is trouble getting the transient analysis to start because the DC operating point can't be calculated. The initial guess for dc could be made from results of such transient analysis; and then transient analysis could be re-run with operating point. You should add any applicable .IC and IC= initial conditions statements to assist in the initial stages of the transient analysis. Be careful when you set initial conditions, for a poor setting may cause convergence difficulties.

- Increasing the ABSTOL and CHGTOL

  While modeling a mechanical system with an RC circuit, where capacitors are in the order of Farad and current impulses is around 10x A, increase CHGTOL and ABSTOL

@ 2023 All Rights Reserved.

by six order of magnitude. For example, change CHGTOL from `0.01e-012` to`0.01e-006`. Simulate the circuit and then start tightening the ABSTOL and CHGTOL values until a convergence error is generated. Once a convergence error is generated, you can revert one step to get the solution.

- Set the DIGSTEPBACK option

  ```
  .OPTIONS DIGSTEPBACK
  ```

  Setting this option might prove useful in cases where your have convergence problems in a circuits with digital components and you are trying to converge using Solver 1.

  Though setting DIGSTEPBACK option might work, it is recommended that you should use solver 0 simulation algorithm to obtain a solution.

Tip

Solver 1 and Solver 0 are two matrix solving algorithms used by PSpice A/D. By default, Solver 1 that has better convergence property, is used. But at times, for a circuit with convergence problems changing the simulation algorithm to Solver 0 helps. To change the simulation algorithm from Solver 1 to Solver 0, open the circuit in the schematic editor. From the *PSpice* menu choose *Edit Simulation Profile*. In the *Simulation Settings* dialog box, select the *Options* tab. Select the General node of the *Analog Advanced tree* in the *Options* button. Change the *Simulation algorithm (SOLVER)* from `default` to `0`.

Note: For more information on all PSpice A/D options, refer to the .OPTIONS(analysis options) section in *PSpice A/D Reference Guide*.

@ 2023 All Rights Reserved.

C

Importing Spice Models

This appendix covers the process to be followed for importing Spice models downloaded from a web site, into PSpice A/D and making them ready to be used in a circuit. The sections covered in this appendix are:

- Tasks for Vendor-Supplied Models on page 638
- Importing Text Models on page 638
- Generating Part Symbols on page 639
- Configuring New Model Library on page 644
- Editing Model Editor Created Symbols on page 646

@ 2023 All Rights Reserved.

Tasks for Vendor-Supplied Models

Usually, the Spice models downloaded from a Vendor's web site cannot be used directly in PSpice A/D. You are required to follow some steps before you can successfully use the downloaded models for designing your circuits.

Before you can use the simulation models downloaded from a web site in your design, you need to perform following steps:

- Importing Text Models
- Generating Part Symbols
- Configuring New Model Library

Importing Text Models

To import the downloaded Spice models into PSpice A/D, you need to perform the following steps.

1. Rename the downloaded model to have the `.MOD` extension.

   Note: Renaming is required only if the downloaded model does not have a `.MOD` extension. For example, renaming will be required if the download model has a .txt extension.

2. Launch the Model Editor.
3. Open a new or your custom model library.
4. From the Model menu, choose Import.
5. In the Open dialog box, select the downloaded model with the `.MOD` extension and select Open.

   Note: Only the first model in the `.MOD` file is imported. Therefore, it is recommended that the `.MOD` file should not have more than one model.

6. From the file Menu, choose Save As. In the Save As dialog box, specify the name and location of new model library as `<installation_directory>\tools\Pspice\library\userlib`.

   Note: It is preferable to create you own USERLIB library folder to store all of your custom part and model libraries for better library management. It is important that you back up your custom libraries and projects on a regular basis to avoid loss of work.

@ 2023 All Rights Reserved.

Generating Part Symbols

After you have imported the downloaded model into PSpice A/D, you need to generate part symbols for the model. You can associate a model to a symbol either by Creating New Symbols or by Using symbols from an existing symbol library or by Using Model Import wizard.

Creating New Symbols

You can create Capture symbols for the imported/downloaded models. Using Model Editor you can either create parts for all the models in a library or you can enable the auto part generation feature in Model Editor, such that part is created every time you save a model.

Creating symbol for the complete library

1. Open the Model Editor.
2. From the *Tools* menu, choose *Options*.
3. Select Capture as the schematic editor and close the *Options* dialog box.
4. From the *File* menu, choose *Export to Capture Part Library*.

5. Specify the location of the model library (`.LIB`) for which you want the symbols to be created.
6. Specify the location of the part library (`.OLB`) to be created and select OK.
7. A message box appears displaying status of part creation process. Select OK to close the message box.

   Note: Any errors or warning messages that are generated during the part creation, are

@ 2023 All Rights Reserved.

saved in a log file named `<library_name>.err`. Referring to the contents of the `.err` file might be helpful, in cases where part creating fails.

Creating symbol for a model

1. From the *Tools* menu, choose *Options*.
2. To enable part creation every time you save a model, select the *Always Create Part when Saving Model* check box.
3. Select *Capture* as the schematic editor.
4. Using the *Save Part To* group box, specify the library in which the new part should be saved and close the *Options* dialog box.

After making the modifications in the *Options* dialog box, a symbol will be generated for the part every time you save the changes in your custom model. The generated symbol will have the same name as that of the simulation model. The name and location of the part library (`.OLB`) will be same as that of the model library (`.LIB`).

Important

If the downloaded Spice model is of `.SUBCKT` type, the Model Editor generates a rectangular symbol. You can edit the Model Editor generated symbol shapes. For more information, see Editing Model Editor Created Symbols.

Using symbols from an existing symbol library

Instead of creating a new symbol from scratch, you can use a symbol from an existing part library, and associate it with the downloaded model. Using symbols from an existing library involves following steps:

- Copying the symbol
  - Copying parts within the same library
  - Copying a part to another library
- Modifying the IMPLEMENTATION property

Before you can use a part symbol from an existing OLB, you need to know the type of simulation model attached with the source part. If the original part symbol is attached to a device characteristic curves-based PSpice model, you only need to modify the implementation property. In case the original part symbol is attached to a template-based

@ 2023 All Rights Reserved.

PSpice model, you will need to add the IMPLEMENTATION and the PSPICETEMPLATE property to the copied model.

Caution

It is recommended that unless you are very comfortable with different types of simulation models supported by PSpice A/D, you should avoid situations where PSPICETEMPLATE property needs to be changed. You can either create symbols using the Model Editor or if you want to copy a symbol, select a symbol of same type and change the IMPLEMENTATION property.

Copying the symbol

Before you copy the part symbol, it is recommended that you create a custom Userlib folder to store your custom symbol and model libraries. Create a sub folder in the pspice folder called Userlib to store your custom libraries.

➡ To create a new library in Capture, from the File menu, choose New and then from the submenu, choose Library.

Copying parts within the same library

Capture does not allow direct copying and pasting of a part in the same library. Therefore, you need to complete the following steps:

1. In Capture, open the symbol library from which you want to copy the symbol. From the File menu, choose Open and then choose Library.
2. Click on the part to be copied so that it becomes highlighted.
3. From the *Edit* menu choose *Copy*.
4. Right-click on the part you just copied and select *Rename*.
5. Type in the name of the new part and click OK.
6. From the *Edit* menu, choose *Paste* to paste the original part back into the library.

Copying a part to another library

1. Open two part libraries in Capture. First, the source library from which the part is to be copied and the second, the destination library to which the model is to be copied. From the *File* menu, choose *Open* | *Library*.

@ 2023 All Rights Reserved.

2. In the Project Manager for the symbol library, position the libraries in a way to enable dragging the part from one library to another.
3. While holding down the `Ctrl` key, drag and drop the required part from the source library to the destination library.

   Note: Alternatively, you can copy the desired part with '*Edit | Copy*' and '*Edit | Paste*' commands.
4. Right click on the part you just copied, select *Rename*, and give the part the desired part name.

Modifying the IMPLEMENTATION property

If you have used existing part symbols, you must ensure that the symbols you have copied and renamed point to the correct model. The part to model referencing is done using the Implementation property.

1. To edit the value of the IMPLEMENTATION property, open the property editor by double clicking on the part. Alternatively, right-click on the part and from the popup menu choose *Edit Properties*.
2. In the *Property Editor* dialog box, ensure that the *Implementation Type* is set to `PSpice Model`.
3. Change the value of the *Implementation* property to the name specified in the model library (`.LIB`) file.

   Note: In case IMPLEMENTATION property is not already present, click *New Row*. In the *Add New Row* dialog box, specify *Name* as `Implementation` and *Value* as the name of the simulation model in the `.LIB` file.

Important

You need not specify any value in the IMPLEMENTATION PATH field, because PSpice A/D will search the model only in the libraries that are configured for the project. Model libraries will be searched in the same sequence as listed in the Library Files list box in the Libraries tab of the simulation setting dialog box.

Adding PSPICETEMPLATE property

The PSPICE TEMPLATE property is required if you want to simulate the part. This property defines the PSpice syntax required for the netlisting the part. This property is not required for parts based on PSpice A/D provided templates. For detailed information on PSPICETEMPLATE property, see PSPICETEMPLATE on page 150.

@ 2023 All Rights Reserved.

Using Model Import wizard

You can use the Model Import wizard either to generate a symbol for the imported model or to associate an existing model to the imported symbol.

Launching Model Import wizard

You can invoke Model Import wizard, using one of the methods listed below.

- Using File menu
  - **a.** From the File menu in Model Editor, choose Model Import Wizard [Capture].
- Using Tools menu
  - **a.** From the Tools menu, choose Options.
  - **b.** Select the *Always Create Part When Saving Model* check box.
  - **c.** Select the *Pick symbols manually* check box.
  - **d.** Click OK.

    Model Import wizard is launched whenever you save the model.

Associating Model

If you launch Model Import wizard from the File menu, in the first page of the wizard, you need to specify the path to the input simulation library as well as the location of the destination symbol library and click Next.

Model Import wizard starts the process of associate a symbol to the downloaded simulation model.

In the Associate/Replace Symbol page of the wizard, you can view the symbol associated with the downloaded model and if required, replace it with the symbol of your own choice.

1. Select the Replace Symbol button.

   Note: If no symbol was associated to the model by the Model Import wizard, use the Associate Symbol button that is available instead of the Replace Symbol button.

2. In the Select Matching page of the wizard, specify the path to the symbol library containing the symbol to be associated with the downloaded model.

@ 2023 All Rights Reserved.

3. From the Matching Symbols list, select the symbol that you want to associate with the downloaded model and click the Save Symbol button.
4. In the Associate/Replace Symbol page, the selected symbol name appears against the downloaded model name. Click Finish to save your changes to the symbol library.

When you use the Model Import wizard to generate or associate symbols to a downloaded model, all the required properties, such as IMPLEMENTATION TYPE, IMPLEMENTATION, and PSPICETEMPLATE, are also updated. Therefore, you need not modify these properties manually.

Configuring New Model Library

After you have generated the part library for a new/customized model library, you need to make the model library available to the design. To ensure this you need to add the model library containing your custom simulation models to the project simulation profile.

1. In Capture, open your Analog or Mixed-Circuit project.
2. From the *PSpice* menu, choose *Edit Simulation Profile*.
3. Select the *Configuration Files* tab.
4. In the *Category* list box, select *Library*.

5. In the *Filename* text box, specify the location of the model library.

@ 2023
All Rights Reserved.

6. To make the library available to all designs, click *Add as Global*. If you want the library to be used only in the current design, select *Add to Design* and close the *Simulation Settings* dialog box.

Note: Instead of editing a simulation profile, you can also create a new simulation profile. To do this, choose *New Simulation Profile* from the *PSpice* menu in Capture.

@ 2023 All Rights Reserved.

Editing Model Editor Created Symbols

Depending on the model definition, different symbol shapes are generated by the Model Editor. Regular symbol shapes are generated for the standard PSpice A/D primitive models that are defined using the `.MODEL` statement. For devices based on a more complicated subcircuit model definition, `.SUBCKT`, a generic rectangle is created that interfaces with the subcircuit model.

For example, if the downloaded simulation model is an OPAMP model defined using a `.SUBCKT` statement, the symbol generated by the Model Editor will be a generic rectangular graphic with pins attached.

In such cases, you can edit the symbol created by the Model Editor. This section demonstrates the steps for editing the symbol generated using the Model Editor for an OPAMP simulation model `LF442A.MOD`, downloaded from the National Semiconductor's web site. `LF442A` is a Dual Low Power JFET Input Operational Amplifier. After you download the simulation model, use the Model Editor to generate the part symbol, as explained in the Generating Part Symbols section.

1. After the symbol generation is complete, open the Model Editor created symbol library in Capture.

   a. Launch Capture. From the Start menu choose *Programs > Installed OrCAD release > Capture*.

   b. From the *File* menu in Capture, choose *Open > Library*.

2. Double click on the part for which the symbol is to be modified.

   The part symbol appears as shown below.

   Instead of a regular triangular graphic, a rectangular graphic is generated. The numbers inside the rectangle are the pin names and the numbers outside the rectangle represent pin numbers.

@ 2023 All Rights Reserved.

You will now edit the Model Editor generated symbol to have a triangular shape.

3. Delete the shape within the dotted line.

4. Redraw the required figure.
For an OPAMP the required figure is a triangle.

a. From the *Place* menu, choose *Line*.

b. Draw a triangle as shown below.

Note: For detailed procedure see the *Editing part graphics* section in *Chapter 5, Creating parts for models* of the PSpice A/D User Guide.

5. Reposition the pins such that the inverting and the non inverting inputs are on the top left and bottom left of the modified symbol. The positive power supply should be on the top and the negative power supply at the bottom. The Output pin should be placed to the right of the modified symbol.

For repositioning pins you need to refer to the pin names as well as the model definition. This is because the pin names are used for model definition.

The relevant section from the model definition for LF442A relating pin names is shown below:

```
* LF442A Dual Low Power JFET Input Operational Amplifier
* ////////////////////////////////////////////////////////
*
* Connections:     Non-inverting input
*                  |   Inverting input
*                  |   |   Positive power supply
*                  |   |   |   Negative power supply
*                  |   |   |   |   Output
*                  |   |   |   |   |
*                  |   |   |   |   |
.SUBCKT LF442A/NS 1   2   99  50  28
*
* Features:
```


@ 2023
All Rights Reserved.

Reposition the OPAMP pins as shown in the following figure:

Using the line tool, draw lines to join pin numbers 3 and 4 to the modified symbol.

6. You can also change the pin numbers and the pin type using the *Pin Properties* dialog box.

Important

Do not change the pin name because pin names are used in the model library (`.lib`) file for model definition.

7. After modifying the symbol as per your specifications, save the symbol and the part library.

You can now use the modified symbol in your design.

For more details on creating custom parts see, Editing Part Graphics (Capture only) and Basing New Parts on a Custom Set of Parts in Chapter 5.

@ 2023 All Rights Reserved.