MicroSim PSpice & Basics

Circuit Analysis Software

User's Guide

MicroSim Corporation
20 Fairbanks
Irvine, California 92618
(714) 770-3022

Version 7.1, October, 1996.

MicroSim Trademarks

Referenced herein are the registered trademarks used by MicroSim Corporation to identify its products. MicroSim is the exclusive owner of: “MicroSim,” “PSpice,” “PLogic,” “PLSyn” and “Polaris.”

Trademarks of MicroSim include: “DesignLab” and “Setting the Standard for Desktop EDA.”

Additional marks of MicroSim include: “StmEd,” “Stimulus Editor,” “Probe,” “Parts,” “Monte Carlo,” “Analog Behavioral Modeling,” “Device Equations,” “Digital Simulation,” “Digital Files,” “Filter Designer,” “Schematics,” ”MicroSim PCBoards,” “PSpice Optimizer,” and variations thereon (collectively the “Trademarks”) are used in connection with computer programs. MicroSim owns various trademark registrations for these marks in the United States and other countries.

All Other Trademarks

MicroSoft, MS-DOS, Windows, Windows NT and the Windows logo are either registered trademarks or trademarks of Microsoft Corporation.

Adobe, the Adobe logo, Acrobat, the Acrobat logo, Exchange and PostScript are trademarks of Adobe Systems Incorporated or its subsidiaries and may be registered in certain jurisdictions.

ShapeBased is a trademark and SPECCTRA and CCT are registered trademarks of Cooper & Chyan Technologies Inc. (CCT). Materials related to the CCT SPECCTRA Autorouter have been reprinted by permission of Cooper & Chyan Technology, Inc.

Xilinx is a registered trademark of Xilinx Inc. All, X- and XC- prefix product designations are trademarks of Xilinx, Inc.

EENET is a trademark of Eckert Enterprises.

All other company/product names are trademarks/registered trademarks of their respective holders.

Copyright Notice

Technical Support

Internet	Tech.Support@MicroSim.com
Phone	(714) 837-0790
FAX	(714) 455-0554
AUTOFAX	(714) 454-3296

Sales Department

Internet	Sales@MicroSim.com
Phone	800-245-3022

Contents

Before You Begin

Part One Simulation Primer

Chapter 1 Simulation Overview

Chapter 2 Simulation Examples

Part Two Design Entry

Chapter 3 Preparing a Schematic for Simulation

Chapter 4 Creating Symbols

Chapter 5 Creating Models

Chapter 6 Analog Behavioral Modeling

Part Three Setting Up and Running Analyses

Chapter 7 Setting Up Analyses and Starting Simulation

Chapter 8 DC Analyses

Chapter 9 AC Sweep Analysis

Chapter 10 Transient Analysis

Chapter 11 Parametric and Temperature Analysis

Chapter 12 Monte Carlo and Sensitivity/Worst-Case Analyses

Part Four Viewing Results

Chapter 13 Waveform Analysis

Chapter 14 Output Options

Appendix ASetting Initial State

Index

Figures

Tables

Before You Begin

Welcome to MicroSim

Welcome to the MicroSim family of products. Whichever programs you have purchased, we are confident that you will find that they meet your circuit design needs. They provide an easy-to-use, integrated environment for creating, simulating and analyzing your circuit designs from start to finish.

MicroSim PSpice Overview

MicroSim PSpice can simulate analog circuits. A schematic is prepared for simulation from which MicroSim Schematics generates a circuit file set. The circuit file set, containing the circuit netlist and analysis commands, is read by MicroSim PSpice for simulation. The results are formulated into meaningful graphical traces in Probe which can be marked for display directly from your schematic.

How to Use this Guide

This guide is designed so you can quickly find the information you need to use PSpice.

This guide assumes that you are familiar with MicroSoft Windows (3.1, NT or 95), including how to use icons, menus and dialog boxes. It also assumes you have a basic understanding about how Windows manages applications and files to perform routine tasks, such as starting applications and opening and saving your work. If you are new to Windows, please review your *MicroSoft Windows User's Guide*.

Typographical Conventions

Before using PSpice, it is important to understand the terms and typographical conventions used in this documentation.

This guide generally follows the conventions used in the *MicroSoft Windows User's Guide*. Procedures for performing an operation are generally numbered with the following typographical conventions.

Notation	Examples	Description
ALL CAPS	ANALOG.SLB or CLIPPER.SCH	Library files and file names.
Ctrl + r	Press Ctrl + r	A specific key or key stroke on the keyboard.
`monospace font`	Type `VAC...`	Commands/text entered from the keyboard.

Mouse Conventions

- If you have a multiple-button mouse, the left mouse button is the primary mouse button, unless you have configured it differently.
- “Point” means to position the mouse pointer until the tip of the pointer rests on whatever you want to point to on the screen.
- “Click” means to press and then immediately release the mouse button without moving the mouse.
- “Right-click means to press the right mouse button and then immediately release the mouse button without moving the mouse.
- “Drag” means to point and then hold down the mouse button as you move the mouse.

Related Documentation

The documentation for all MicroSim products is available in both hard-copy and on-line.

Manual Name[*]	**Description**
MicroSim Schematics User's Guide	Provides information about how to use MicroSim Schematics, which is a schematic capture front-end program with a direct interface to other MicroSim programs and options.
MicroSim PCBoards User's Guide	Provides information about MicroSim PCBoards, which is a PCB layout editor that allows you to specify printed circuit board structure, as well as the components, metal and graphics required for fabrication.
MicroSim PCBoards Autorouter User's Guide	Provides information on the integrated interface to Cooper & Chyan Technology's (CCT) SPECCTRA autorouter in MicroSim PCBoards.
MicroSim PSpice A/D & Basics+ User's Guide	Describes the capabilities of PSpice A/D, Probe, Stimulus Editor, and Parts utility. It provides examples for demonstrating the process of specifying simulation parameters, analyzing simulation data results, editing device stimuli, and creating models.
MicroSim PSpice A/D Reference Manual	Provides reference material for PSpice A/D. Also included: detailed descriptions of the simulation controls and analysis specifications, start-up option definitions, and a list of device types in the analog and digital model libraries. User interface commands are provided to instruct you on each of the screen commands.
MicroSim Application Notes [**]	Provides a variety of articles that show you how a particular task can be accomplished using MicroSim's products, and examples that demonstrate a new or different approach to solving an engineering problem.
MicroSim PSpice Optimizer User's Guide	Provides information for using the PSpice Optimizer for analog performance optimization.
MicroSim PLSyn Programmable Logic Synthesis	Provides information for using programmable logic synthesis.
MicroSim/AMD PLD Design System User's Guide	Provides information about the implementation of a PLD design targeted for using one or more of AMD's devices.
MicroSim Filter Designer User's Guide	Provides information about designing electronic frequency selective filters.
Library Reference Manual[**]	Provides a complete list of the analog and digital parts in the model and symbol libraries.

---

[*] On-line documentation is available only to those users who install MicroSim products by CD-ROM.

[**] This manual is provided in on-line format *only*.

On-Line Help

Pressing `F1` or selecting Contents from the Help menu brings up an extensive on-line help system.

The on-line help includes:

- Step-by-step instructions on how to use PSpice features.
- Reference information about PSpice.
- Technical Support information.

If you are not familiar with Windows (3.1, NT or 95) Help System, select How to Use Help from the Help menu.

If You Don't Have the Standard PSpice Package

If You Have PSpice Basics

PSpice Basics provides the basic functionality needed for analog and mixed-signal design without the advanced features in the full PSpice package that some users do not need. Since this user guide is for both PSpice and PSpice Basics users, however, there are some features described in this user guide that are not available for PSpice Basics users.

The Basics icon next to this paragraph is used throughout the user guide to mark each section or paragraph which describes a feature ***not available*** for PSpice Basics users. If an entire section describes a "non-Basics" feature, the icon is placed next to the section title. If an individual paragraph describes a "non-Basics" feature, the icon is placed next to the paragraph.

not included in: Basics+

The following table identifies which features are included with PSpice and PSpice Basics.

Feature	PSpice	PSpice Basics
Benefits of integration with MicroSim Schematics		
graphical design entry (schematic capture)	yes	yes
simulation setup using dialog boxes	yes	yes
cross-probing	yes	yes
multi-window analysis of Probe data sets	yes	yes
marching waveforms in Probe	yes	yes
board layout package interfaces	yes	yes

Feature	**PSpice**	**PSpice Basics**
Notable PSpice analysis and simulation features		
DC sweep, AC sweep, transient analysis	yes	yes
noise, Fourier, temperature analysis	yes	yes
parametric analysis	yes	no
Monte Carlo, sensitivity/worst-case analysis	yes	no
analog behavioral modeling (ABM)	yes	yes
Stimulus Editor	yes	no
Parts utility	yes	no
performance analysis (goal functions)	yes	no
save/load bias point	yes	no
Notable PSpice devices and library models		
GaAsFETs: Curtice, Statz, TriQuint, Parker-Skellern	all	Statz
MOSFETs: SPICE3 (1-3) with charge conservation, BSIM1, BSIM3 (version 2)	yes	yes
IGBTs	yes	no
JFETs, BJTs	yes	yes
resistor, capacitor, and inductor .MODEL support	yes	yes
ideal, non-ideal lossy transmission lines	all	ideal
coupled transmission lines	yes	no
nonlinear magnetics	yes	no
voltage- and current-controlled switches	yes	yes
analog model library	8500+	8500+ *
Purchase options		
MicroSim PCBoards	yes	yes
MicroSim PSpice Optimizer	yes	no
Device Equations	yes	no
network licensing	yes	no

Feature	PSpice	PSpice Basics
Miscellaneous specifications		
unlimited circuit size	yes	no[**]

[*] PSpice Basics package includes all libraries except SCRs, thyristors, PWMs, magnetic cores, and transmission lines.

[**] Unlimited circuit size means you can have as many components in your design as can be accommodated by your computer’s memory—rule-of-thumb is one Mb of RAM for every 400 transistors or 4,000 gates in the circuit. The PSpice Basics package is limited by the total device count: 100 large devices (all transistors and ABM devices) or 500 small devices

If You Have the Evaluation CD-ROM

MicroSim’s evaluation CD-ROM has the following limitations:

- Schematic capture limited to one schematic page (A-size)
- Maximum of 25 symbols can be placed on a schematic
- Maximum of nine symbol libraries can be configured
- Maximum of 20 symbols in a user-created symbol library
- Circuit simulation limited to circuits with up to 64 nodes, 10 transistors, two operational amplifiers, or a combination thereof, and 10 ideal transmission lines with not more than four non-ideal lines (lossy lines using RLGC parameters) and four coupled lines
- Device characterization limited to diodes
- Sample library includes 22 analog parts

Part One

Simulation Primer

Part One provides basic information about circuit simulation and includes examples of common analyses.

Chapter 1, Simulation Overview, provides an overview of the circuit simulation process and describes standard simulation analysis types.

Chapter 2, Simulation Examples, includes examples of common analyses which provide an introduction to the methods and tools for entering circuit designs, running simulations with PSpice, and analyzing simulation results using Probe.

Simulation Overview

1

Chapter Overview

This chapter provides an overview of the circuit simulation process and describes standard simulation analysis types.

Creating and Analyzing Circuits on page 1-2 describes the process of circuit simulation. The simulation process consists of creating and preparing a circuit for simulation, simulating a circuit with PSpice, and displaying simulation results.

Analysis Types on page 1-6 describes the analysis types for PSpice.

Creating and Analyzing Circuits

Figure 1-1 illustrates the process of creating a circuit, simulating the circuit, and displaying the simulation results. This process consists of the following:

1. The circuit schematic is created and prepared for simulation using MicroSim Schematics.
2. Circuit simulation and analysis are performed by PSpice.
3. Probe is used to display simulation results.

The term “simulator” refers to PSpice, which simulates analog circuits.

Each of these procedures is described in the following pages.

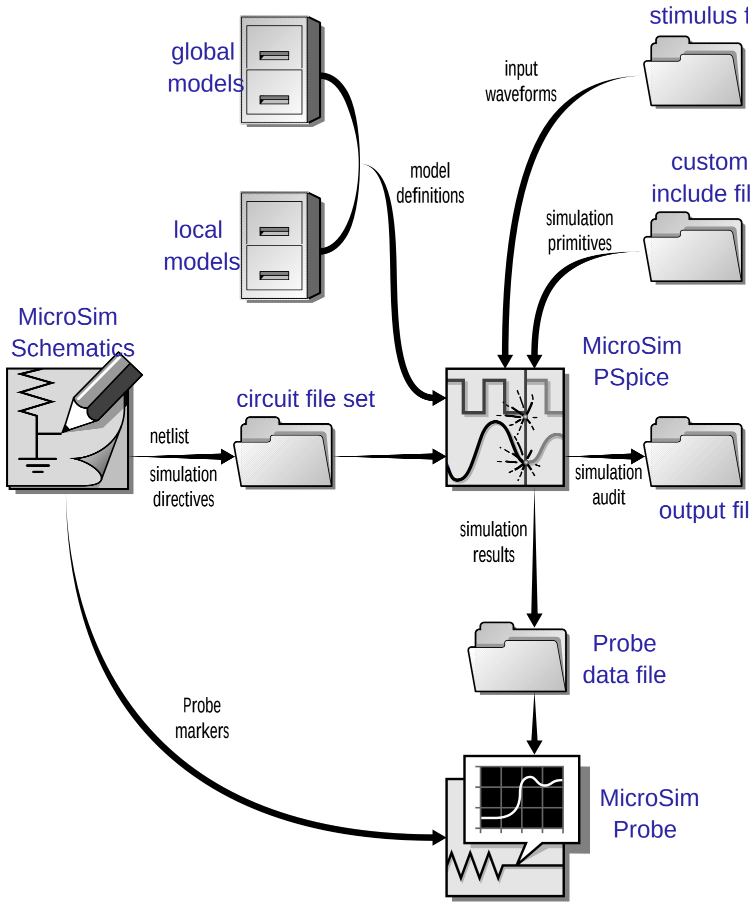


Figure 1-1 *Circuit Analysis File Interactions*

Creating and Preparing a Circuit for Simulation

The following files are used for circuit simulation:

- Circuit file set
- Model files
- Stimulus files
- Custom include files

Circuit file set

The circuit file set includes the following files:

- Primary circuit file, which contains analysis commands, simulation control commands, and references to the netlist, alias, model, and other files required by the circuit to be simulated.
- Netlist file, which describes the components and connections of the circuit.
- Alias file, which maps the Schematics part and pin names to the simulator device and node names.

The filename of the primary circuit file is *schematic*`.cir`, where *schematic* is the name of the schematic.

The filename of the netlist file is *schematic*`.net`, where *schematic* is the name of the schematic.

The filename of the alias file is *schematic*`.als`, where *schematic* is the name of the schematic.

Model files

Model files contain the electrical definition of the components of a schematic. Each model is comprised of an individual device or a subcircuit.

Device models describe a primitive part by a set of parameters. Subcircuits are functional groupings of components to which elements in external circuits can be connected.

The most commonly used models are available in the MicroSim model library. If needed, however, you can create models. See Chapter 5 Creating Models for more

Stimulus files

Stimulus files are user-defined files containing analog stimulus waveform specifications. These can be automatically created using the Stimulus Editor utility or created manually. The creation of these files is transparent when specifying VSTIM, ISTIM, or DIGSTIM stimulus part instances in your schematic.

See The Stimulus Editor Utility on page 10-5 for a description of how to use the Stimulus Editor.

Note *PSpice Basics does not include the Stimulus Editor.*

Custom include files

Custom include files are user-defined files containing simulator command and device instructions referenced in the circuit file set. These files can include user-defined functions that are used in numeric expressions.

Simulating a Circuit with PSpice

See **Part Three,** *Setting Up and Running Analyses*, for a description of how to set up and perform simulation.

Simulation and analysis are performed by PSpice. The simulator interprets the information in the Schematics files for the circuit, runs the simulation, and produces the Probe data file and simulation output file.

To simulate a circuit, the wire connections to the pins of parts shown in a schematic are transformed into electrical node connections to device terminals that are understood by the simulator.

PSpice is a analog electrical circuit simulator that can calculate the behavior of analog circuits with speed and accuracy. PSpice simulates analog circuits and calculates voltages and currents of the devices and nodes.

PSpice uses the files generated by Schematics to run the simulation, and produces a Probe data file and simulation output file.

Displaying Simulation Results

Simulation results are written to the Probe data file and simulation output file.

See **Part Four,** *Viewing Results*, for a description of how to use Probe to display simulation results.

Probe data file

The Probe data file contains simulation results to be viewed and manipulated interactively with the graphical waveform analyzer program, Probe. As the simulation proceeds, the simulator updates the Probe data file with simulation results.

Probe reads the data stored in the Probe data file and displays waveforms reflecting circuit response at marked nets, pins, and devices in your schematic or for output variables specified in Probe. The results can be further manipulated through expressions and search functions, and families of curves can be displayed on a single plot.

Analyzing data in Probe can help you to determine the validity of your design. Both your schematic and simulation setup parameters can be refined based upon earlier results, thus creating an iterative process of running simulations and Probe analyses.

Simulation output file

The simulation output file is an ASCII text file containing lists and tables describing the input circuit, the analysis directives, and the results of the specified simulations—in essence, an audit trail of simulation events and results.

The content of the simulation output file is determined by the types of analyses run, the options selected for running PSpice, and the pseudocomponents placed and connected to nets in your schematic. For example, each instance of a VPRINT1 pseudocomponent placed in your schematic and connected to a different net causes PSpice to save the voltage values at those nets to the simulation output file. Any warning or error messages encountered during read-in or simulation are also written to the output file.

Analysis Types

PSpice circuit analyses can be classified as follows:

- Standard
- Simple multi-run
- Statistical

These analyses types are described in the following pages.

See **Chapter 2,**Simulation Examples, for introductory examples of how to run each type of analysis.

See **Part Three,** *Setting Up and Running Analyses*, for a more detailed description of how to run each type of analysis.

Standard analyses

PSpice standard analyses include the following:

Table 1-1 *PSpice Standard Analysis Types*

DC sweep	Voltages and currents of the circuit's steady-state response are calculated where a source, model parameter, or temperature is swept over a range of values.
Bias point detail	Additional bias point data is calculated and reported (bias point is computed automatically by PSpice).
DC sensitivity	Calculates the DC sensitivity of a node or component's voltage as a function of bias point.
Small-signal DC transfer	Calculates small-signal DC gain, input resistance, and output resistance as a function of bias point.
Frequency response (AC sweep)	Calculates the small-signal response of the circuit, linearized around the bias point, to a combination of inputs. One or more sources are swept over a range of frequencies. The voltages and currents of the circuit are calculated, including magnitude and phase. This can be used to obtain Bode plots.

Table 1-1 *PSpice Standard Analysis Types (continued)*

Noise analysis	A noise analysis can be performed in conjunction with the frequency response analysis. The contribution of each noise generator in the circuit is propagated to an output node for every frequency specified in the AC analysis. These propagated contributions, as well as their RMS sum, are calculated for each frequency.
Transient response	Circuit behavior is tracked over time in response to time-varying sources. The voltages and currents are computed.
Fourier components	A Fourier analysis can also be performed in conjunction with a transient analysis. Calculates the DC and Fourier components of the transient analysis results.

Simple multi-run analyses

The following PSpice analyses allow the circuit-wide temperature, a global parameter, or a characteristic of a particular component to be changed, resulting in each specified DC sweep, AC sweep, or transient analysis being rerun with the new value. The simple multi-run analyses provided are:

Table 1-2 *PSpice Simple Multi-Run Analysis Types*

Parametric	Repeats the specified standard analyses as a global parameter, model parameter, component value, or operational temperature is stepped through a series of values.
Temperature	Repeats the specified standard analyses as the operational temperature is stepped through a list of values.

not included in: Basics+

not included in: Basics+

Statistical analyses

Two PSpice analyses are provided that allow device and lot tolerances to be applied to device models, resulting in multiple runs of the specified DC sweep, AC sweep, or transient analysis. The statistical analyses provided are:

Table 1-3 *PSpice Statistical Analysis Types*

Monte Carlo	Computes circuit response to changes in component values by randomly varying all device model parameters for which a tolerance has been specified.
Sensitivity/ worst-case	Computes circuit response to changes in component values by varying one device model parameter (where a tolerance is specified) at a time on a device by device basis, culminating in a single run where all model parameters for all devices are set to their worst-case values.

Simulation Examples

2

Chapter Overview

The examples in this chapter provide an introduction to the methods and tools for entering circuit designs, running simulations with PSpice, and analyzing simulation results using Probe. All analyses are performed on the same example circuit. This allows for a clearer illustration of analysis setup, simulation, and result analysis procedures for each analysis type.

This chapter includes the following sections:

Example Circuit Design Entry

This section describes how to use MicroSim Schematics to enter the simple diode clipper circuit shown in Figure 2-1.

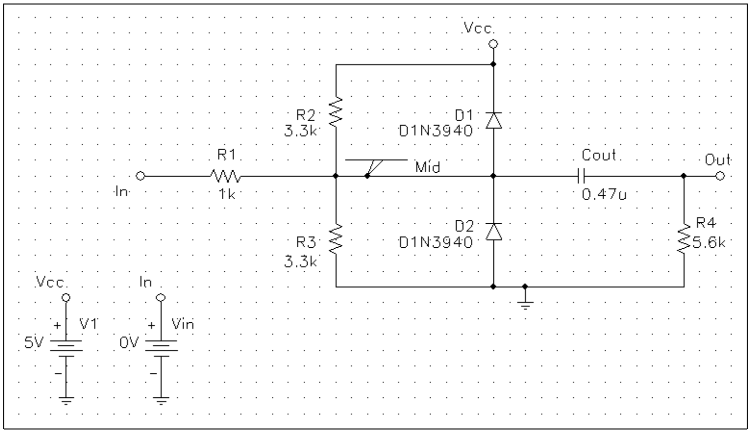


Figure 2-1 *Diode Clipper Circuit*

To open a new schematic window

1. Start Schematics. If Schematics is already running, be sure you are in the schematic editor.
2. If you are in a blank schematic window (indicated by "new" in the title bar at the top of the window), you can begin entering the circuit.

   If you need to open a new schematic window, click the New icon or select New from the File menu.

New icon:

To place the voltage sources

Shortcut: Click on or press

1. Select Get New Part from the Draw menu to display the Part Browser dialog box.
2. Type `VDC` in the Part Name text box.
3. Click on Place & Close.
4. Move the cursor to position the source at the desired location on the schematic page.

If you have enough room on your screen, click on Place to leave the Part Browser dialog box open.

5. Click to place the first source.
6. Move the cursor and click again to place the second source.
7. Right-click to end placement mode.

To place the diodes

1. Go to the Part Browser dialog box.
2. Type `D1N39*` in the Part name text box.
3. Press Enter to display a list of diodes.
4. Click on D1N3940.
5. Click on Place (to leave the dialog box open) or Place & Close (to close the dialog box).
6. Press Ctrl + r to rotate the diode outline to the desired orientation.
7. Click to place the first diode (D1), and click again to place the second diode (D2).
8. Right-click to end placement mode.

If needed, click on to redisplay the Part Browser dialog box.

When placing components:

- Leave space to connect the components with wires.
- If device names and values don't match the names shown in Figure 2-1, they can be changed later.

To move the text associated with the diodes (or any other object)

1. Click once on the text to select it.
2. Drag the text to the desired location.

Select Redraw from the View menu or press Ctrl + l to redraw the circuit as necessary.

To place the other components

Follow similar steps as described for the diodes to place the components listed below. The symbol names you need to type in the Part name text box of the Part Browser dialog box are shown in parentheses.

1. resistors (`R`)
2. capacitor (`C`)
3. ground symbols (`EGND`)
4. bubble symbols (`BUBBLE`)

Shortcut: Click on ✎ or press Ctrl + w .

You can right-click at any time to stop the wiring mode. The cursor changes to the default arrow.

If necessary, double-right click or press Spacebar to resume wiring mode. The cursor changes back to a pencil.

Clicking on any valid connection point terminates a wire. A valid connection point is shown as an “x” (see Figure 2-2).

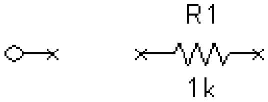


Figure 2-2 *Connection Points*

If you make a mistake when placing or connecting components:

1 Click on the wire or component.
2 Press Del.

Bubbles serve as “wireless” connections where connectivity is implied by identical labels.

To connect the components

1 Select Wire from the Draw menu to enter wiring mode. The cursor changes to a pencil.

2 Click on the connection point (the very end) of the pin on the bubble at the input of the circuit.

3 Click on the nearest connection point of the input resistor R1.

4 Connect the other end of R1 to the output capacitor.

5 Connect the diodes to each other and to the wire between them:

   a Click on the connection point of the anode for the lower diode.

   b Move the cursor straight up and click on the wire between the diodes. The wire terminates and the junction of the wire segments is made visible.

   c Click again on the junction to continue wiring

   d Click on the end of the upper diode’s cathode pin.

6 Continue connecting components until the circuit is wired as shown in Figure 2-1 on page 2-2.

To assign names (labels) to the nets and bubbles

1 Double-click on any segment of the wire which connects R1, R2, R3, the diodes, and the capacitor.

2 Type `Mid` in the Label text box.

3 Click on OK.

4 Double-click on each bubble to label it as shown in Figure 2-1 on page 2-2.

To place the viewpoint symbol

1. Type `VIEWPOINT` in the Part Name text box of the Part Browser dialog box.
2. Place the viewpoint so that its connection point (pin end) touches the wire labeled Mid.

To assign a name to a device

A particular name can be assigned to a device in the schematic (like “Vin” for a VDC, or Cout for the capacitor) as follows:

1. Double-click on the reference designator for the device.
2. Type the new name in the Edit Reference Designator dialog box.
3. Click on OK.

Example: assigning the name “Vin” for the second VDC.

1. Double-click on the reference designator of the VDC symbol, V2.
2. Type `Vin` in the Edit Reference Designator dialog box.
3. Click on OK.

To change the value of a device

1. Double-click on the value for the device.
2. Type the new value in the Edit Value dialog box.
3. Click on OK.

To save your schematic

1. Select Save from the File menu.
2. Type `CLIPPER` as the schematic file name.
3. Click on OK to save the file as `clipper.sch`.

Shortcut: Click on 💾 or press Ctrl + s .

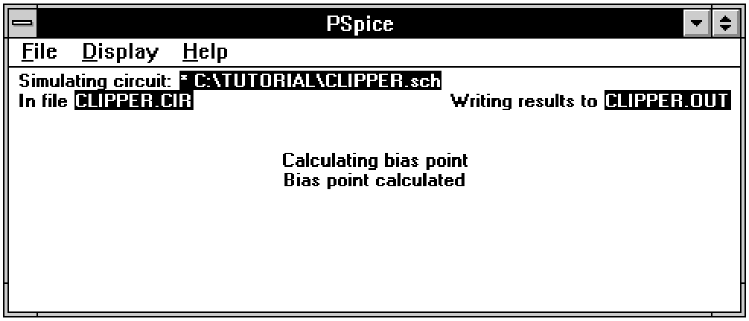
Bias Point Analysis

Running PSpice

Shortcut: Click on or press F11.

After creating the schematic `clipper.sch`, you can run PSpice by selecting Simulate from the Analysis menu. PSpice performs the simulation and generates the output file (`clipper.out`).

While PSpice is running, the progress of the simulation is displayed in the PSpice simulation status window (see Figure 2-3).



```
Simulating circuit: * C:\TUTORIAL\CLIPPER.sch
In file CLIPPER.CIR                              Writing results to CLIPPER.OUT

                         Calculating bias point
                         Bias point calculated
```

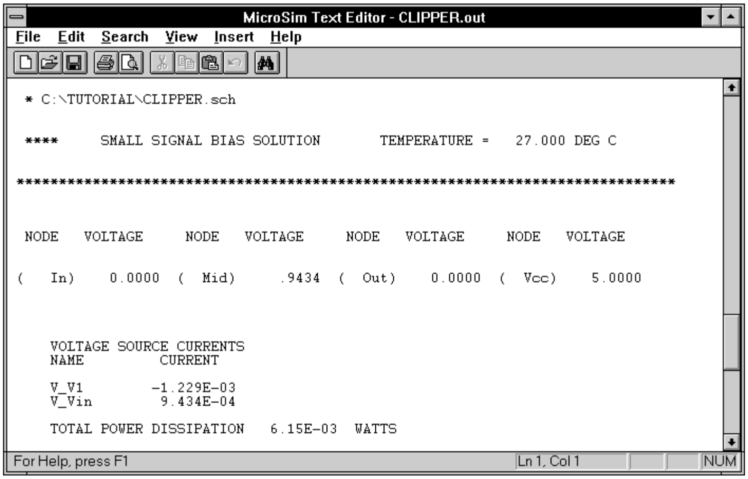
Figure 2-3 *PSpice Simulation Status Window*

Simulation Output File

The simulation output file acts as an audit trail of the simulation. This file optionally echoes the contents of the circuit file as well as the results of the bias point calculation. If there are any syntax errors in the netlist declarations or simulation directives or anomalies while performing the calculation, error and/or warning messages are written to the output file.

To view the simulation output file

1. Close the PSpice window.
2. In Schematics, select Examine Output from the Analysis menu to display the output file in the MicroSim text editor window. Figure 2-4 shows the results of the bias point calculation as written in the simulation output file (`clipper.out`).
3. When finished, close the MicroSim text editor window.


MicroSim Text Editor - CLIPPER.out
File Edit Search View Insert Help
For Help, press F1
Ln 1, Col 1
NUM

```
 * C:\TUTORIAL\CLIPPER.sch

 ****     SMALL SIGNAL BIAS SOLUTION       TEMPERATURE =   27.000 DEG C

*******************************************************************************


 NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE

(   In)    0.0000  (  Mid)     .9434  (  Out)    0.0000  (  Vcc)    5.0000



    VOLTAGE SOURCE CURRENTS
    NAME         CURRENT

    V_V1        -1.229E-03
    V_Vin        9.434E-04

    TOTAL POWER DISSIPATION   6.15E-03  WATTS
```

Figure 2-4 *Simulation Output File*

Since the diodes are both reverse biased (off), and the input source Vin is 0V (a short circuit to ground), the bias point is dependent only on the values of VCC, R1, R2, and R3.

The voltage at net Mid is in agreement with manual calculation:

$$V(MID) = \frac{Req}{R2 + Req} \times VCC$$

where:

$$Req = \frac{R1 \times R3}{R1 + Req}$$

Correct, expected bias point analysis results provide assurance of proper circuit connectivity.

Note that the current through VIN is negative. By convention, PSpice measures the current through a two terminal device into the first terminal and out of the second terminal. For voltage sources, current is measured from the positive terminal to the negative terminal; this is opposite to the positive current flow convention and results in a negative value.

Voltage Viewpoint

The VIEWPOINT symbol was initially a blank line. When the simulation is finished and bias point voltages are available, the viewpoint reflects the voltage at the net to which it was connected (in this case, the voltage at the node Mid is 0.9434).

DC Sweep Analysis

You can visually verify the DC response of the clipper by performing a DC sweep of the input voltage source and displaying the waveform results in Probe. This example sets up DC sweep analysis parameters to sweep Vin from -10 to 15 volts in 1 volt increments.

Setting Up and Running a DC Sweep Analysis

To set up and run a DC sweep analysis

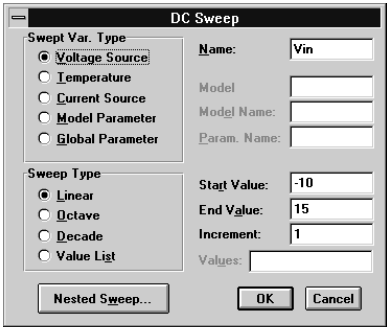
1 Select Setup from the Analysis menu.

Shortcut: Click on .

2 Click on the DC Sweep button in the Analysis Setup dialog box.

3 Set up the DC Sweep dialog box as shown in Figure 2-5.

Note *The default settings for the DC Sweep dialog box are Voltage Source as the swept variable type and Linear as the sweep type. To select a different swept variable type or sweep type, click on the appropriate radio button.*


DC Sweep

Swept Var. Type
● Voltage Source
○ Temperature
○ Current Source
○ Model Parameter
○ Global Parameter

Name: Vin
Model
Model Name:
Param. Name:

Sweep Type
● Linear
○ Octave
○ Decade
○ Value List

Start Value: -10
End Value: 15
Increment: 1
Values:

Figure 2-5 *DC Sweep Dialog Box*

4 Click on OK to exit the DC Sweep dialog box.

5 If needed, click on the DC Sweep check box in the Analysis Setup dialog box so that it is checked on (enabled).

6 Click on Close to exit the Analysis Setup dialog box.

7 Click on the Save icon.

Save icon:

You can also save the circuit under a different name. To save the circuit as `clipperd.sch`, for example, select Save As from the File menu in the schematic editor and enter `clipperd` as the file name.

8 To run the analysis as specified, select Simulate from the Analysis menu.

Shortcut: Click on or press F11.

Displaying DC Analysis Results in Probe

To set up Probe to automatically run after simulation, select Probe Setup from the Analysis menu and click on the Automatically Run Probe After Simulation box so that it is checked on.

Shortcut: Click on 📈 or press Insert.

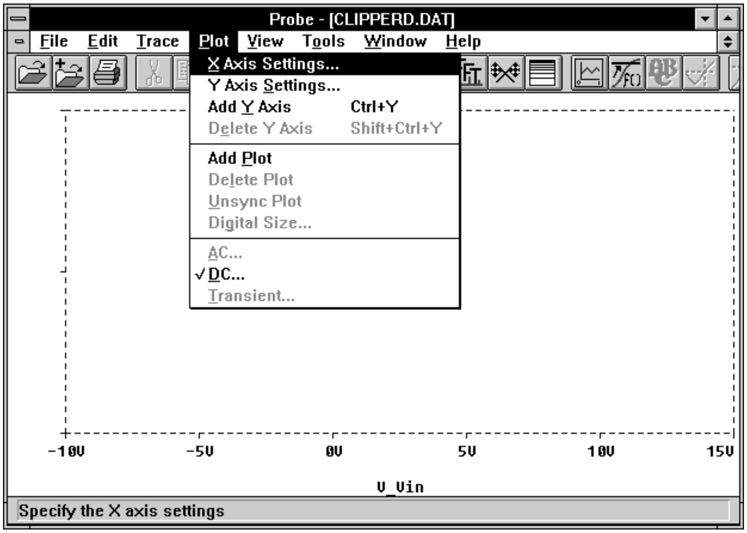
If Probe is set up to automatically run upon successful completion of a simulation (the default setting), then the Probe window is displayed when the simulation is finished. The Probe window displays a plot screen like the one shown in Figure 2-6.

To plot voltages at nets 1 and 2

1. Select Add from the Trace menu.
2. Click on V(In) and V(Mid) in the Add Traces dialog box.
3. Click on OK.

To display traces using markers

1. Delete the two traces you plotted above:
   - **a** Click on the text V(In) located under the X axis of the plot to select trace V(In).
   - **b** Press and hold the Shift key and click on V(Mid) to also select trace V(Mid).
   - **c** Press Del to remove both traces.



Specify the X axis settings

Figure 2-6 *Probe Plot*

2 In Schematics, select Mark Voltage/Level from the Markers menu.

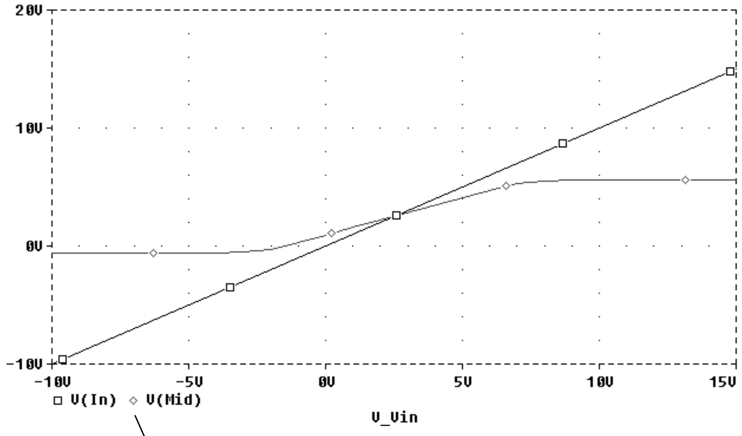
Shortcut: press Ctrl+M.

3 Click to place the first marker on net In.

4 Click to place the second marker on net Mid.

5 Right-click to end marker mode. Waveform traces are displayed as shown in Figure 2-7.

You can click on 💾 in Schematics to save the schematic with markers.



trace legend

Figure 2-7 *Voltage at In and Mid*

Labels are included in Figure 2-7 for informational purposes only. Refer to the *Schematics User's Guide* for a description of how to add labels.

This example uses the cursors feature to view the numeric values for two traces and the difference between them by placing a cursor on each trace.

To place a cursor on each trace

1 In Probe, select Cursor from the Tools menu, then select Display to activate cursors.

Two cursors are activated for the first trace defined in the legend below the X axis—V(In) in this example. The Probe Cursor window is also displayed.

Table 2-1 *Association of Probe Cursors with Mouse Buttons*

Cursor 1	left mouse button
Cursor 2	right mouse button

2 To display the cursor cross-hairs:

- **a** Position the mouse cursor anywhere inside the plot window.
- **b** Click to display the cross-hairs for the first cursor.
- **c** Right-click to display the cross-hairs for the second cursor.

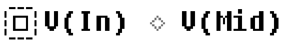


Figure 2-8 *Trace Legend with Cursors Activated*

In the trace legend, the symbol for V(In) is outlined in the crosshair pattern for each cursor, resulting in a dashed line as shown in Figure 2-8.

3 Place the first cursor on the V(In) waveform:

- **a** Click on the portion of the V(In) trace in the proximity of 4 volts on the X axis. The cursor cross-hair is displayed, and the current X and Y values for the first cursor are displayed in the Probe Cursor window.

Your ability to get as close to 4.0 as possible depends on screen resolution and window size.

- **b** To fine-tune the cursor location to 4 volts on the X axis, drag the cross-hairs until the corresponding entry in the Probe Cursor window is approximately 4.0 for the X axis value of the A1 cursor. You can also press → and ← for tighter control.

4 Place the second cursor on the Mid waveform:

- **a** Associate the second cursor with the Mid waveform by right-clicking on the trace legend symbol for V(Mid) (on the diamond). The crosshair pattern for the second cursor outlines the V(Mid) trace symbol as shown in Figure 2-9.

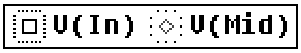


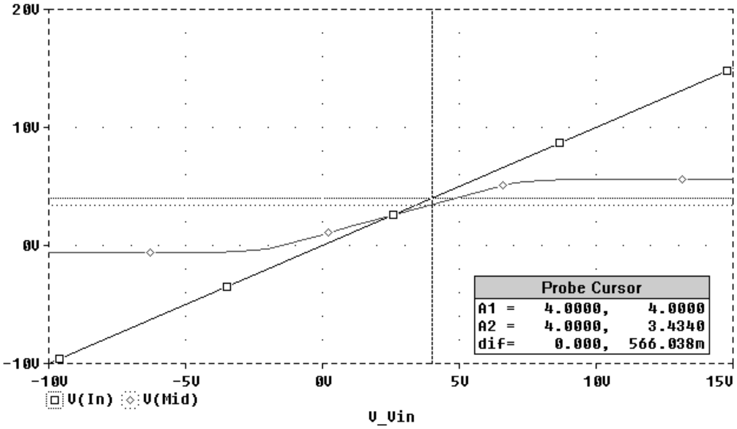
Figure 2-9 *Trace Legend with V(Mid) Symbol Outlined*

- **b** Right-click on the portion on the V(Mid) trace that is in the proximity of 4 volts on the X axis. The X and Y values for cursor 2 are displayed in the Probe Cursor window along with the difference between the two cursors' X and Y values.
- **c** To fine-tune the location of the second cursor to 4 volts on the X axis, drag the cross-hairs until the corresponding entry in the Probe Cursor window is approximately 4.0 for the X axis value of the C2 cursor. You can also press Shift+→ and Shift+← for tighter control.

Figure 2-10 shows the Probe window when both cursors are placed.

There are other ways to display the difference between two voltages at a trace:

- By plotting "V(In)-V(Mid)" in Probe
- By selecting Mark Voltage Differential from the Markers menu in Schematics



Probe Cursor

```
A1 =     4.0000,     4.0000
A2 =     4.0000,     3.4340
dif=      0.000,   566.038m
```

Figure 2-10 *Voltage Difference at V(In) = 4 Volts*

At this point, the schematic has been saved. If needed, you can close Schematics and Probe and begin the remaining analysis exercises later using the saved schematic.

Transient Analysis

This example shows how to run a transient analysis on the clipper circuit. This requires adding a time-domain voltage stimulus as shown in Figure 2-11.

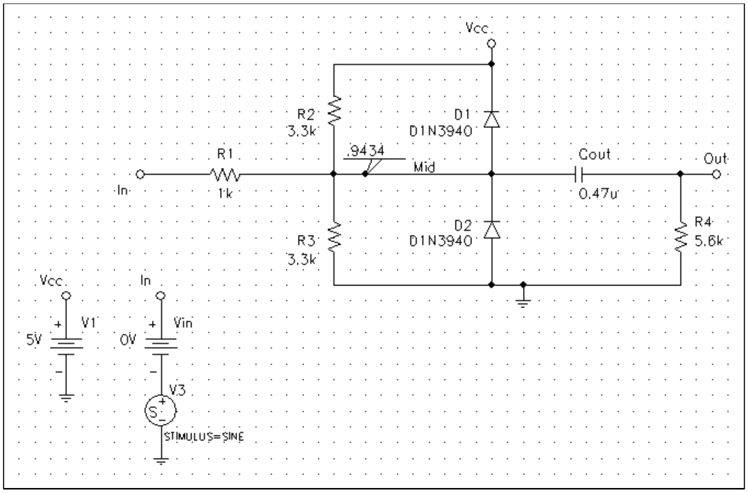


Figure 2-11 *Diode Clipper Circuit with a Voltage Stimulus*

To add a time-domain voltage stimulus

1. In Schematics, select Clear All from the Markers menu.
2. Select the ground symbol beneath the VIN source.
3. Select Cut from the Edit menu.
4. Scroll down or select Out from the View menu (or click on the Zoom Out icon).
5. Place a VSTIM symbol as shown in Figure 2-11.
6. Select Paste from the Edit menu.
7. Place the ground symbols as shown in Figure 2-10.
8. Select Fit from the View menu.
9. Select Save As from the File menu, and save the file as `clippert.sch`.
10. Double-click on the VSTIM symbol to start the Stimulus Editor.

Shortcut: press Ctrl + x .

If your version of Schematics does not include the Stimulus Editor (Basics users):

1. Place a VSIN symbol instead of VSTIM.
2. Double click on it.
3. Double-click on the appropriate attributes to set their values as VOFF, VAMPL, and FREQ. Click on Save Attr or press Enter after entering each attribute's value to accept the changes.
4. Click on OK.

11 When prompted to name the stimulus, type `SINE` and click on OK.

12 In the New Stimulus dialog box of the Stimulus Editor, click on SIN, and click on OK.

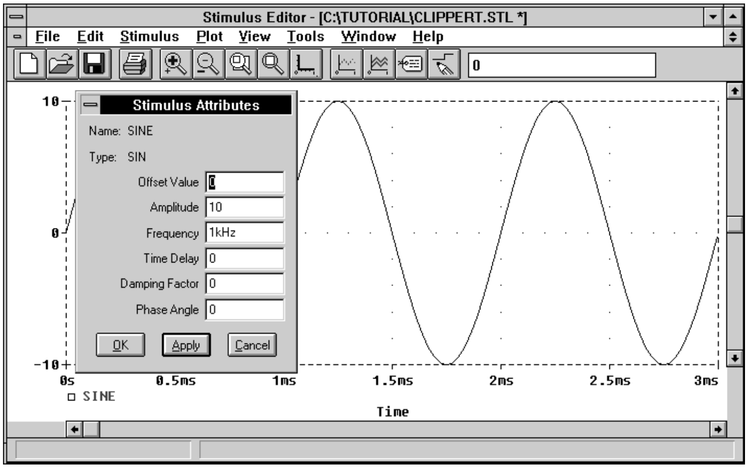
13 In the Stimulus Attributes dialog box, set the first three parameters as follows:

Offset Voltage = 0

Amplitude = 10

Frequency = 1kHz

14 Click on Apply to view the waveform. The Stimulus Editor window should look like Figure 2-12.



Stimulus Editor - [C:\TUTORIAL\CLIPPERT.STL *]

Stimulus Attributes

Name: SINE

Type: SIN

Offset Value: 0

Amplitude: 10

Frequency: 1kHz

Time Delay: 0

Damping Factor: 0

Phase Angle: 0

Figure 2-12 *Stimulus Editor Window*

15 Click on OK.

16 Click on the Save icon or press Shift+F12 to save the stimulus information.

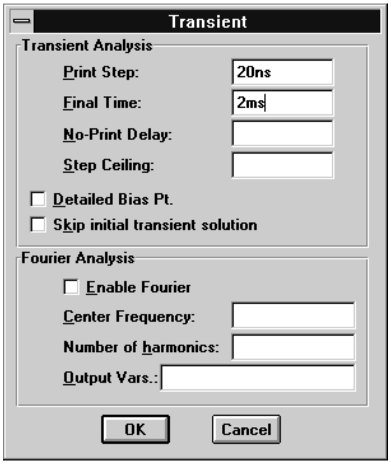
Save icon:

17 Select Exit from the File menu.

To set up and run the transient analysis

Analysis Setup icon:

1. In Schematics, click on the Analysis Setup icon or select Setup from the Analysis menu to display the Analysis Setup dialog box.
2. Click on Transient to display the Transient Analysis dialog box.
3. Set up the Transient dialog box as shown in Figure 2-13.



Transient

Transient Analysis

Print Step: 20ns

Final Time: 2ms

No-Print Delay:

Step Ceiling:

☐ Detailed Bias Pt.

☐ Skip initial transient solution

Fourier Analysis

☐ Enable Fourier

Center Frequency:

Number of harmonics:

Output Vars.:

Figure 2-13 *Transient Analysis Dialog Box*

4. Click on OK.
5. If needed, click on Transient check box in the Analysis Setup dialog box so that it is checked on (enabled).
6. If needed, disable DC sweep from the previous example by clicking on the DC Sweep check box so that it is *not* checked.

   DC Sweep is disabled here so that you can see the results of a transient analysis run by itself. PSpice can run multiple analyses during simulation, and you could run both DC sweep and transient analyses.

7. Click on Close to exit the Analysis Setup dialog box.

Save icon:

8. Click on the Save icon.

Simulate icon:

9. Click on the Simulate icon or press `F11` to start the simulation.

PSpice uses its own internal time steps for computation. The internal time step is adjusted according to the requirements of the transient analysis as it proceeds. Data is saved to the Probe data file for each internal time step.

Note *The internal time step is different from the Print Step value. Print Step controls how often optional text format data is written to the simulation output file (.*OUT*).*

To display the output variable list with aliases and display desired traces in Probe

This portion of the example describes how to view the input sine wave and the clipped wave at Out. The output variable list is used as an alternative to markers to select the traces to be displayed.

1. Select Add from the Trace menu. (You can also click on the Add Trace icon or press Insert.)
2. Select V(Out) and V(In) by clicking on them in the trace list.
3. Click on OK. The traces are displayed.
4. Place the symbols shown in the trace legend on the traces themselves as shown in Figure 2-14:
   - **a** Select Options from the Tools menu.
   - **b** Click on Always in the Use Symbols portion of the dialog box.
   - **c** Click on OK.
5. Click on the Save icon.

Save icon:

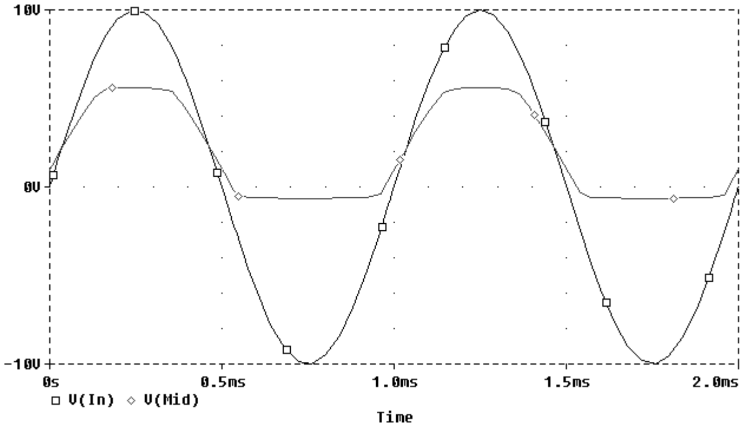


Figure 2-14 *Sinusoidal Input and Clipped Output Waveforms*

These waveforms illustrate the clipping of the input signal.

AC Sweep Analysis

The AC sweep analysis in PSpice is a linear (or small signal) frequency domain analysis which can be used to observe the frequency response of any circuit at its bias point.

Setting Up and Running an AC Sweep Analysis

This example sets up the clipper circuit for AC analysis by adding an AC voltage source for a stimulus signal as shown in Figure 2-15, and by setting up AC sweep parameters.

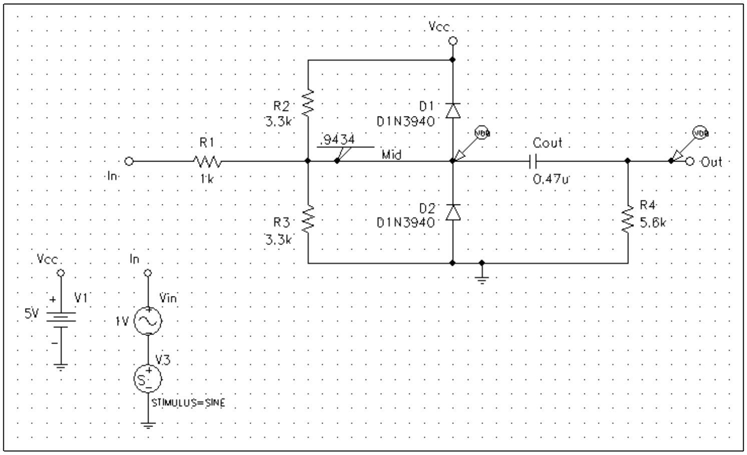


Figure 2-15 *Clipper Circuit with AC Stimulus*

To change Vin to include the AC stimulus signal

1. Click on the DC input source, Vin, to select it.
2. Select Replace from the Edit menu.
3. In the Replace Part dialog box, type `VAC`.
4. Click on the Keep Attribute Values check box so that it is checked on.
5. Click on OK. The input voltage source changes to an AC voltage source.

Time-domain and AC stimuli are independent of one another. For example, the SINE stimulus is ignored (0V) during AC analysis.

The new Vin still has a DC attribute which can be used to include a bias with the AC source. Double-click the AC source to see the DC attribute value.

6 Double-click on the displayed (AC) value of the new Vin. In the Set Attr Value dialog box, set the value to `1V`.

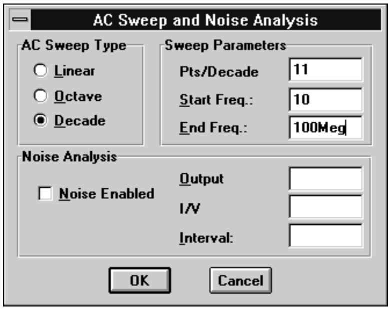
To set up the AC sweep and start simulation

1 Click on the Analysis Setup icon or select Setup from the Analysis menu to display the Analysis Setup dialog box.

Analysis Setup icon:

2 Click on AC Sweep.

3 Set up the AC Sweep and Noise Analysis dialog box as shown in Figure 2-16.



AC Sweep and Noise Analysis

AC Sweep Type
○ Linear
○ Octave
● Decade

Sweep Parameters
Pts/Decade 11
Start Freq.: 10
End Freq.: 100Meg

Noise Analysis
☐ Noise Enabled
Output
I/V
Interval:

Figure 2-16 *AC Sweep and Noise Analysis Dialog Box*

Note *PSpice is not case sensitive, so both M and m can be used as “milli,” and MEG, Meg, and meg can all be used for “mega.”*

4 Click on OK to close the AC Sweep dialog box.

5 Click on Close to close the Analysis Setup dialog box.

6 Select Mark Advanced from the Markers menu.

7 Double-click on vdb.

8 Place one Vdb marker on the output node, and place another on the Mid node.

9 Select Save As from the File menu, and save the schematic as `clippera.sch`.

10 Click on the Simulate icon or press F11 to start the simulation.

Simulate icon:

Since the transient analysis was not deactivated, PSpice performs both the transient and AC analyses.

AC Sweep Analysis Results

Probe displays the dB magnitude (20log10) of the voltage at the marked nodes, Out and Mid, as shown in Figure 2-17. VDB(Mid) has a lowpass response due to the diode capacitances to ground. The output capacitance and load resistor act as a highpass filter, so the overall response, illustrated by VDB(out), is a bandpass response. Since AC is a linear analysis and the input voltage was set to 1V, the output voltage is the same as the gain (or attenuation) of the circuit.

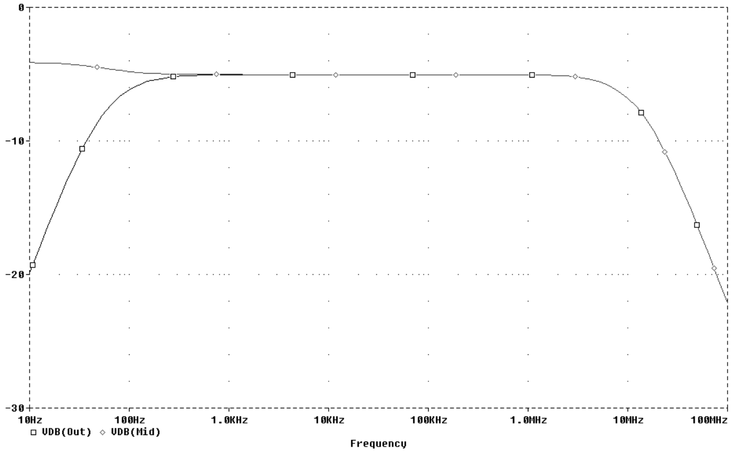


Figure 2-17 *dB Magnitude Curves for "Gain" at Mid and Out*

To display a Bode plot of the output voltage, including phase

1. Switch back to the Schematics window.
2. Select Mark Advanced from the Markers menu.
3. Place a vphase marker on the output next to the Vdb.
4. Delete the Vdb marker on Mid.
5. Switch to Probe. The gain and phase plots are both displayed on the same graph with the same scale.
6. Click on the trace name, VP(Out), to select it. The text should turn red.

Note *Depending upon where the vphase marker was placed, the trace name may be different, such as VP(Cout:2), VP(R4:1), or VP(R4:2).*

7 Select Cut from the Edit menu.

Shortcut: Click on ✂ or press Ctrl + x .

8 Select Add Y Axis from the Plot menu.

9 Select Paste from the Edit menu. The Bode plot is displayed as shown in Figure 2-18.

Shortcut: Click on 📋 or press Ctrl + v .

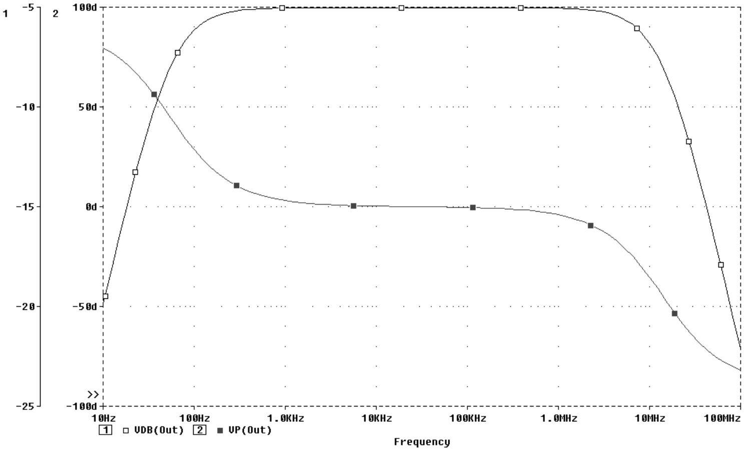


Figure 2-18 *Bode Plot of Clipper's Frequency Response*

not
included
in:

Parametric Analysis

This example shows the effect of varying input resistance on the bandwidth and gain of the clipper circuit. This is done by:

- Changing the value of R1 to the expression {Rval}
- Adding a PARAM symbol to declare the parameter Rval
- Specifying a parametric analysis to step the value of R1 via Rval

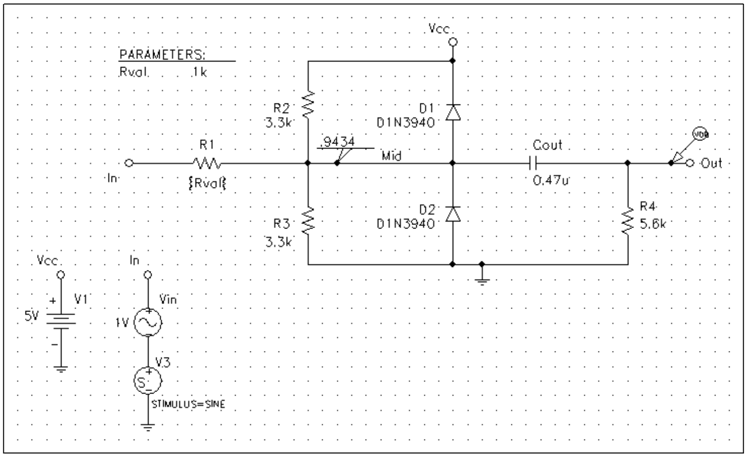


Figure 2-19 *Clipper Circuit with Global Parameter Rval*

The example results in 21 analysis runs, each with a different value of R1. Once the analysis is complete, you can analyze curve families for the analysis runs in Probe.

Setting Up and Running the Parametric Analysis

To change the value of R1 to the expression {Rval}

1. Double-click on the value of R1.
2. Type `{Rval}` in the Set Attribute Value dialog box.
3. Click on OK.

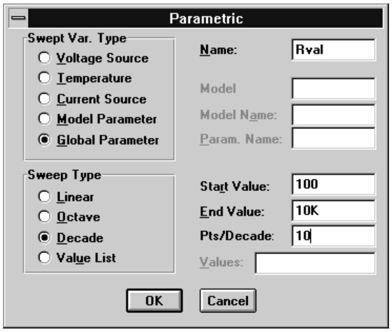
PSpice interprets text in curly braces as an expression which evaluates to a numerical value. This example uses the simplest form of an expression—a constant. The value of R1 will take on the value of the Rval parameter, whatever it may be.

To add a PARAM symbol to declare the parameter Rval

1. Click on the Part Browser icon or press Ctrl + g .
2. Type `PARAM` in the Part text box, and click on OK.
3. Place one PARAM symbol on the schematic.
4. Double-click on the PARAM symbol to display the attributes list.
5. Double-click on NAME1 and type `Rval` (no curly braces) in the Value text box.
6. Click on Save Attr to accept the changes.
7. Double-click on VALUE1, type `1k`, and click on Save Attr.
8. Click on OK. Rval 1k is displayed in the PARAMETERS list on the schematic.

Part Browser icon:

Analysis Setup icon:



Swept Var. Type
- ○ Voltage Source
- ○ Temperature
- ○ Current Source
- ○ Model Parameter
- ● Global Parameter

Name: Rval

Model:

Model Name:

Param. Name:

Sweep Type
- ○ Linear
- ○ Octave
- ● Decade
- ○ Value List

Start Value: 100

End Value: 10K

Pts/Decade: 10

Values:

Figure 2-20 *Parametric Dialog Box*

This setup specifies that the parameter Rval is to be stepped from 100 to 10k logarithmically with a resolution of 10 points per decade.

The analysis is run for each value of Rval. Since the value of R1 is defined as {Rval}, the analysis is run for each value of R1 as it logarithmically increases from 100Ω to 10 kΩ in 20 steps, resulting in a total of 21 runs.

To set up and run a parametric analysis to step the value of R1 via Rval

1. Click on the Analysis Setup icon or select Setup from the Analysis menu to display the Analysis Setup dialog box.
2. Click on Parametric.
3. Set up the Parametric dialog box as shown in Figure 2-20.
4. Click on OK.
5. If needed, click on the Parametric check box in the Analysis Setup dialog box so that it is checked on (enabled).
6. Click on the Transient check box so that it is *not* checked.
7. Click on Close to close the Analysis Setup dialog box.
8. Select Save As from the File menu, and save the schematic as `clipperp.sch`.
9. Delete the VP marker. (For this example, we are only interested in the magnitude of the response.)
10. To run the analysis as specified, click on the Simulate icon or press F11.

Analyzing Waveform Families in Probe

To display all 21 traces in Probe

1 There are 21 analysis runs, each with a different value of R1. When Probe starts, it displays the Available Sections dialog box which lists all 21 runs and the Rval parameter value for each. You have the option to select one or more runs.

Click on OK to accept the default of all runs. All 21 traces (the entire family of curves) for VDB(Out) are displayed in Probe as shown in Figure 2-21.

To select individual runs, click on each one separately.

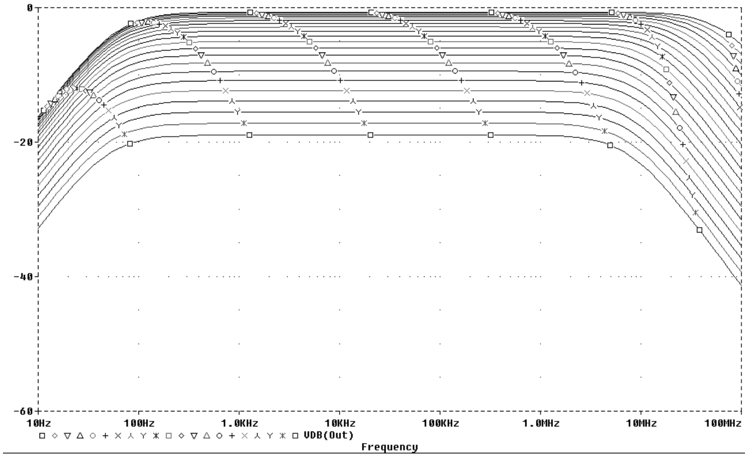


Figure 2-21 *Small Signal Response as R1 is Varied from 100Ω to 10 kΩ*

To see more information about the section which produced a specific trace, double-click on the corresponding symbol in the legend below the X axis.

2 To remove the traces shown, click on the trace name to select it and then press Del.

You can also remove the traces by deleting the VDB marker in Schematics.

To compare the last run to the first run

1 Select Add from the Trace menu.

Shortcut: Click on or press Insert.

2 Type the following text in the Trace Command text box:

```
Vdb(Out)@1 Vdb(Out)@21
```

You can avoid some of the typing for the Trace Command text box by clicking on the V(OUT) selection in the trace list and editing the resulting Trace Command to insert text where appropriate.

Note *The difference in gain is apparent, you can also plot the difference of the waveforms for runs 21 and 1 and then use the search commands feature to find certain characteristics of the difference.*

3 Delete the two traces.

4 Plot the new trace by specifying a waveform expression:

Shortcut: Click on or press Insert.

- **a** Select Add from the Trace menu.
- **b** Type the following waveform expression in the Trace Command text box:

```
Vdb(Out)@1-Vdb(OUT)@21
```

- **c** Click on OK.

5 Use the search commands feature to find the value of the difference trace at its maximum and at a specific frequency:

Shortcut: Click on .

- **a** Select Cursor from the Tools menu, and then select Display.

Shortcut: Click on .

- **b** Select Cursor from the Tools menu, and then select Max.

Shortcut: Click on .

- **c** Select Cursor from the Tools menu, and then select Search Commands.
- **d** Enter the following search command:

```
search forward x value (100)
```

This command instructs Probe to search for the point on the trace where the X axis value is 100.

- **e** Select cursor 2 as the cursor to move.
- **f** Click on OK.

Figure 2-22 shows the Probe window when the cursors are placed.

Note that the Y value for cursor 2 shown in the cursor box at the lower right is about 17.87. This indicates that when R1 is set to 10 kΩ, the small signal attenuation of the circuit at 100Hz is 17.87dB greater than when R1 is 100Ω.

6 Deactivate the cursors and delete the trace to prepare for the next exercise.

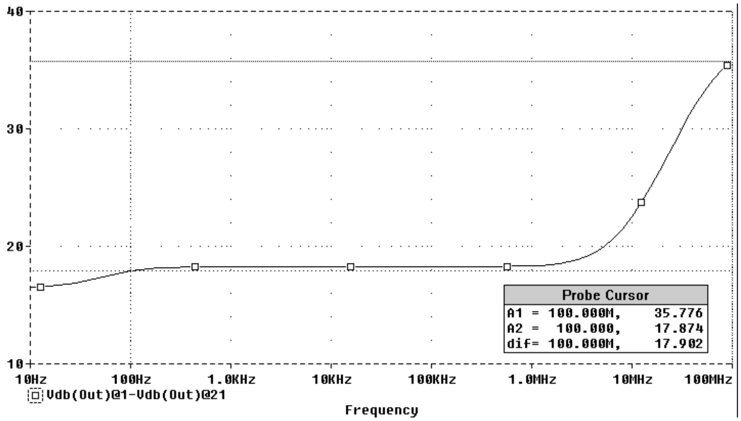


Figure 2-22 *Comparison of Small Signal Frequency Response at 100 and 10 kΩ Input Resistance*

Probe Performance Analysis

not
included
in: Basics +

Probe's performance analysis is an advanced feature that can be used to compare the characteristics of a family of waveforms. Performance analysis uses the principle of search commands introduced earlier in this chapter to define functions which detect points on each curve in the family.

Once these functions have been defined, they can be applied to a family of waveforms using Add on the Trace menu and produce traces that are a function of the variable that changed within the family.

This example shows how to use the performance analysis feature of Probe to view the dependence of circuit characteristics on a swept parameter. In this case, the small signal bandwidth and gain of the clipper circuit are plotted against the swept input resistance value.

To plot bandwidth vs. Rval using the performance analysis wizard

1 In Probe, select Performance Analysis from the Trace menu. The Performance Analysis dialog box is displayed with information on the currently loaded data and performance analysis in general.

At each step, the wizard provides information and guidelines to follow.

2 Click on Wizard.

3 Click on Next>.

4 Click on Bandwidth in the Choose a Goal Function list, and then click on Next>.

Shortcut: Click on 📈, then double-click on V(Out).

5 Click in the Name of Trace text box, and type `V(Out)`.

6 Click in the db Level Down text box and type `3`.

7 Click on Next>. The wizard displays the gain trace for the first run (R=100) and shows how the bandwidth is measured. This is done to test the goal function.

8 Click on Next> or Finish. Probe displays a plot of the 3dB bandwidth vs. Rval.

9 Change the X-axis to log scale.

  a Select X Axis Settings from the Plot menu.

  b Select Log in the Scale list box.

  c Click on OK.

Shortcut: Double-click on the X axis.

To plot gain vs. Rval manually

1 Select Add Y Axis from the Plot menu.

2 Select Add from the Trace menu.

3 Click on the Max(1) goal function. Use the scroll bar to move to the right in the Trace list.

4 Scroll to the left again and click on V(out).

5 Edit the Trace Command text box to be `Max(Vdb(out))`, and click on OK. Probe displays gain on the second Y axis, vs. Rval.

Shortcut: Click on 📈 or press Insert.

The Trace list includes goal functions only in performance analysis mode when the X-axis variable is the swept parameter.

Figure 2-23 shows the final performance analysis plot of 3dB bandwidth and gain in dB vs. the swept input resistance value.

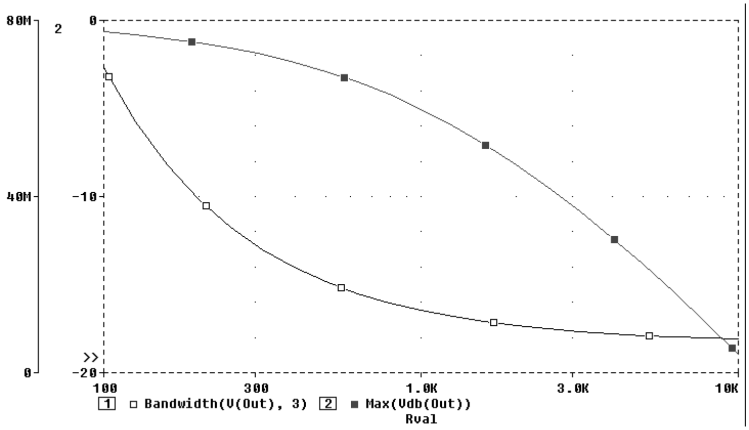


Figure 2-23 *Performance Analysis Plots of Bandwidth and Gain vs. Rval*

For more information about performance analysis, see Example: RLC Filter on page 11-3.

Part Two
Design Entry

Part Two provides information about entering circuit designs in MicroSim Schematics.

Chapter 3,Preparing a Schematic for Simulation, provides information to help you enter circuit designs that simulate properly.

Chapter 4,Creating Symbols, provides information about creating symbols. Symbols are the graphical representation of circuit components.

Chapter 5,Creating Models, provides information about creating models. Models define the electrical behavior of schematic components for purposes of simulation.

Chapter 6,Analog Behavioral Modeling, describes how to use Analog Behavioral Modeling (ABM) feature provided in PSpice.

Preparing a Schematic for Simulation

3

Chapter Overview

This chapter provides information to help you enter circuit designs that simulate properly.

Passive Parts on page 3-2 describes setup information for a variety of analog parts. For a listing of standard library components, refer to the on-line *Library Reference Manual*.

Parameters and Expressions on page 3-9 describes how to use parameters and expressions in place of literal values.

Analog Power Supplies on page 3-13 describes setup information for DC sources.

The information in this chapter supplements the information in the *Schematics User's Guide*. The *Schematics User's Guide* describes how to use Schematics to create circuit schematics, and this chapter includes additional information to help you set up components to ensure proper simulation of the schematic.

Passive Parts

This section describes setup information for a variety of analog parts. For a listing of standard library components, refer to the on-line *Library Reference Manual*.

Resistors, Capacitors, and Inductors

Also see Resistors, capacitors, and inductors on page 5-26.

Table 3-1 *Resistor, Capacitor, and Inductor Attributes*

Part Type	Symbol Name	Symbol Library File	Attribute	Description
capacitor (PSpice ‘C’ device)	C	analog.slb analog_p.slb[*]	VALUE	capacitance
			IC	initial voltage across the capacitor during bias point calculation
	C_VAR		VALUE	base capacitance
			SET	multiplier
inductor (PSpice ‘L’ device)	L	analog.slb analog_p.slb	VALUE	inductance
			IC	initial current through the inductor during bias point calculation
resistor (PSpice ‘R’ device)	R	analog.slb analog_p.slb	VALUE	resistance
			TC	linear and quadratic temperature coefficients
	R_VAR		VALUE	base resistance
			SET	multiplier

[*] The Symbol Library file ANALOG_P.SLB contains R, L, C, R_VAR, and C_VAR symbols with visible pin numbers. To use the symbols in this file instead of those in ANALOG.SLB, use Editor Configuration on the Options menu to add ANALOG_P.SLB to the list of configured Symbol Library files before the ANALOG.SLB file entry. Refer to the *Schematics User’s Guide* for a description of how to use Editor Configuration.

For the standard R, C, and L parts, the effective value of the part is set directly by the VALUE attribute. For the variable resistors and capacitors, R_VAR and C_VAR, the effective value is the product of the base value (VALUE) and multiplier (SET).

In general, resistors, capacitors, and inductors should have positive component values (VALUE attribute). **In all cases, components must not be given a value of zero.**

However, there are cases when negative component values are desired. This occurs most often in filter designs that analyze an RLC circuit equivalent to a real circuit. When transforming from the real to the RLC equivalent, it is possible to end up with negative component values.

PSpice allows negative component values for bias point, DC sweep, AC, and noise analyses. In the case of resistors, the noise contribution from negative component values come from the absolute value of the component (components are not allowed to generate negative noise). A transient analysis may fail for a circuit with negative components. The negative components, especially negative-valued capacitors and inductors, may create instabilities in time which the analysis cannot handle.

Transformers

Table 3-2 lists attributes that can be set per instance of a linear transformer part.

For additional information:

- See Transformers on page 5-29.
- See *Using the Inductor Coupling Symbols* in the *Application Notes Manual* for a description of techniques for creating additional transformer configurations.
- See Using the KBREAK symbol in a schematic on page 5-27 for information about using nonlinear magnetic cores with transformers.

Table 3-2 *Transformers*

Part Type	Symbol Name	Symbol Library File	Attribute	Description
transformer (PSpice ‘K’ and ‘L’ devices)	XFRM_LINEAR	analog.slb	L1_VALUE L2_VALUE	winding inductances in Henries
			COUPLING	coefficient of mutual coupling (must lie between 0 and 1)
	K_LINEAR	analog.slb	Ln	inductor reference designator

Transmission Lines

Ideal and lossy transmission lines

Table 3-3 lists the attributes that can be set per instance of an ideal (T) or lossy (TLOSSY) transmission line. The parts contained in the `tline.slb` symbol file contain a variety of transmission line types. Their part attributes vary.

Table 3-3 *Transmission Lines*

Part Type	Symbol Name	Symbol Library File	Attribute	Description
transmission line (PSpice 'T' device)	T	analog.slb	Z0	characteristic impedance
			TD	transmission delay
			F	frequency for NL
			NL	number of wavelengths or wave number
	TLOSSY[*]	analog.slb	LEN	electrical length
			R	per unit length resistance
			L	per unit length inductance
			G	per unit length conductance
			C	per unit length capacitance

*. Not available for Basics users.

PSpice uses a distributed model to represent the properties of a lossy transmission line. That is, the line resistance, inductance, conductance, and capacitance are all continuously apportioned along the line's length.

A common approach to simulating lossy lines is to model these characteristics using discreet passive elements to represent small sections of the line. This is the lumped model approach, and it involves connecting a set of many small subcircuits in series as shown in Figure 3-1. This method requires that enough lumps exist to adequately represent the distributed characteristic of the line. This often results in the need for a large netlist and correspondingly long simulation time. The method also produces spurious oscillations near the natural frequencies of the lumped elements.

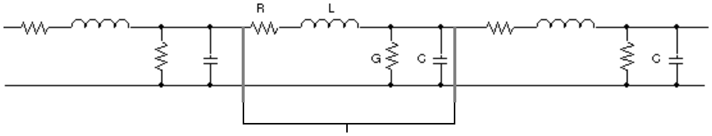


Figure 3-1 *Lossy Line Comprised of Lumped Line Segments*

The distributed model used in PSpice frees you from having to determine how many lumps are sufficient, and eliminates the spurious oscillations. It also allows lossy lines to be simulated with the same accuracy in a fraction of the time required by the lumped approach.

In addition, you can make R and G general Laplace expressions. This allows frequency dependent effects to be modeled, such as skin effect and dielectric loss.

not included in: Basics +

Coupled transmission lines

Table 3-4 lists the attributes that can be set per instance of a coupled transmission line symbol. The symbol library provides symbols that can accommodate up to five coupled transmission lines. You can also create new symbols that have up to ten coupled lines.

Table 3-4 *Coupled Transmission Lines*

Part Type	Symbol Name	Symbol Library File	Attribute	Description
coupled transmission line—symmetric (PSpice ‘K’ and ‘T’ devices)	T2COUPLED T3COUPLED T4COUPLED T5COUPLED	tline.slb	LEN	electrical length
			R	per unit length resistance
			L	per unit length inductance
			G	per unit length conductance
			C	per unit length capacitance
			LM	per unit length mutual inductance
			CM	per unit length mutual capacitance
coupled transmission line—asymmetric (PSpice ‘K’ and ‘T’ devices)	T2COUPLEDX[*] T3COUPLEDX T4COUPLEDX T5COUPLEDX	tline.slb	LEN	electrical length
			R	per unit length resistance
			L	per unit length inductance
			G	per unit length conductance
			C	per unit length capacitance
			LM	per unit length mutual inductance
			CM	per unit length mutual capacitance
transmission line coupling matrix (PSpice ‘K’ device)	KCOUPLE2	tline.slb	T1	name of first coupled line
			T2	name of second coupled line
			LM	per unit length mutual inductance
			CM	per unit length mutual capacitance
	KCOUPLE3 KCOUPLE4 KCOUPLE5		T1	name of first coupled line
			T2	name of second coupled line
			T3	name of third coupled line
			LMij	per unit length mutual inductance between line Ti and line Tj
			CMij	per unit length mutual capacitance between line Ti and line Tj

[*] T2COUPLEDX is functionally identical to T2COUPLED. However, the T2COUPLEDX implementation uses the expansion of the subcircuit referenced by T2COUPLED.

not
included
in: Basics+

Simulating coupled lines

Coupling between transmission lines can be simulated using the PSpice K device. Each of the coupled transmission line parts provided in the standard symbol library translate to K device and T device declarations in the netlist. PSpice compiles a system of coupled lines by assembling capacitive and inductive coupling matrices from all of the K devices involving transmission lines. Though the maximum order for any one system is ten lines, there is no explicit limitation on the number of separate systems that may appear in one simulation.

The simulation model is accurate for:

- ideal lines
- low-loss lossy lines
- systems of homogeneous, equally spaced high-loss lines

For more information about the K device, refer to the *PSpice Reference Manual*.

Simulation considerations

When simulating, transmission lines with short delays can create performance bottlenecks by setting the time step ceiling to a very small value.

If one transmission line is setting the time step ceiling frequently, PSpice reports the three lines with the shortest time step. The percentage attenuation, step ceiling, and step ceiling as percentage of transmission line delay are displayed in the PSpice status window.

If your simulation is running reasonably fast, you can ignore this information and let the simulation proceed. If the simulation is slowed significantly, you may want to abort the simulation and modify your design. If the line is lossy and showing negligible attenuation, model the line as ideal instead.

Parameters and Expressions

Parameters

In many applications, it is convenient to use a parameter instead of a numeric value. A parameter is like a programming variable used to indirectly represent a value. Parameters are useful for assigning like values to multiple part instances as well as setting up analyses involving swept variables (parametric, for example).

A parameter is defined using the PARAM pseudocomponent. Up to three parameters can be defined per PARAM instance by assigning each parameter's name and value to the corresponding NAME*n* and VALUE*n* attributes. These are global to the entire schematic regardless of hierarchical level.

The OPTPARAM pseudocomponent can also be used to define global parameters, though its primary use is to define design parameters for use with the PSpice Optimizer. If you choose to use OPTPARAM, up to eight parameters can be defined by specifying values for Name and Initial Value.

For instance, an independent voltage source, VCC, can have its VALUE attribute set to the value of a parameter called VSUPPLY using the notation {VSUPPLY}. The curly braces instruct the simulator to evaluate VSUPPLY rather than use the text representation itself.

VSUPPLY must also be defined by placing a PARAM pseudocomponent in your schematic, and defining one of the NAME*n*/VALUE*n* attribute pairs. To set VSUPPLY to 12 volts, the first pair of PARAM attributes can be defined as:

```
NAME1 = VSUPPLY
VALUE1 = 12V
```

Note *The system variables in* Table 3-7 on page 3-13 *have reserved parameter names. User-defined parameters should not use these parameter names.*

By using parameters, multiple devices can respond to a change in that parameter. For instance, if a second independent source, VEE, also has its value set to {VSUPPLY} as well, then both sources can be changed to 14 volts by simply reassigning the VALUE1 attribute of the PARAM pseudocomponent to 14V.

Expressions

Additional flexibility can be provided by using parameters in expressions. See Parameters on page 3-9 for more information about parameters.

In many applications, it is helpful to have more flexibility in setting up values by using expressions in place of literal values.

For example, an independent source, VEE, could have its VALUE attribute defined as:

```
VALUE = {-10*FACTOR}
```

where FACTOR is assigned the value of 1.2 using a PARAM pseudocomponent. The value for VEE resolves to (-10 * 1.2) or -12.0 volts.

Expressions can contain the standard operators listed in **Table 3-5**, the functions in **Table 3-6**, and the system variables listed in **Table 3-7**.

The simulator accepts expressions of any length, but MicroSim Schematics imposes some length limits. Part attribute definitions, for example, are limited to 1,024 characters in Schematics. In such cases, it is appropriate to define custom functions for expressions, thus decreasing length and improving readability.

The simulator evaluates expressions upon reading in the entire circuit file set. Parameter values that change as an analysis proceeds (a DC sweep or parametric analysis, for example) cause the simulator to reevaluate the expression at that time.

Table 3-5 *Operators in Expressions*

Type	Operators	Meaning
arithmetic	+	addition (or string concatenation)
	-	subtraction
	*	multiplication
	/	division
	**	exponentiation

Table 3-5 *Operators in Expressions (continued)*

Type	Operators	Meaning
relational: within IF() functions	==	equality test
	!=	non-equality test
	>	greater than test
	>=	greater than or equal to test
	<	less than test
	<=	less than or equal to test

Table 3-6 *Functions in Arithmetic Expressions*

Function	Meaning	Comment
ABS(x)	$\|x\|$	
SQRT(x)	$x^{1/2}$	
EXP(x)	e^x	
LOG(x)	$\ln(x)$	log base e
LOG10(x)	$\log(x)$	log base 10
PWR(x,y)	$\|x\|^y$	
PWRS(x,y)	$+\|x\|^y$ (if $x > 0$) $-\|x\|^y$ (if $x < 0$)	
SIN(x)	$\sin(x)$	x in radians
ASIN(x)	$\sin^{-1}(x)$	result in radians
SINH(x)	$\sinh(x)$	x in radians
COS(x)	$\cos(x)$	x in radians
ACOS(x)	$\cos^{-1}(x)$	result in radians
COSH(x)	$\cosh(x)$	x in radians
TAN(x)	$\tan(x)$	x in radians
ATAN(x)	$\tan^{-1}(x)$	result in radians
ARCTAN(x)		
ATAN2(y,x)	$\tan^{-1}(y/x)$	result in radians
TANH(x)	$\tanh(x)$	x in radians

Table 3-6 *Functions in Arithmetic Expressions (continued)*

Function	Meaning	Comment
M(x)	magnitude of x	same result as ABS(x)
P(x)	phase of x	in degrees; returns 0.0 for real numbers
R(x)	real part of x	
IMG(x)	imaginary part of x	applicable to AC analysis only
DDT(x)	time derivative of x	applicable to transient analysis only
SDT(x)	time integral of x	applicable to transient analysis only
TABLE(x,x_1,y_1,...)	y value as a function of x	x_n,y_n point pairs are plotted and connected by straight lines
MIN(x,y)	minimum of x and y	
MAX(x,y)	maximum of x and y	
LIMIT(x,min,max)	min if x < min max if x > max else x	
SGN(x)	+1 if x > 0 0 if x = 0 -1 if x < 0	
STP(x)	1 if x > 0 0 otherwise	used to suppress a value until a given amount of time has passed; for example, {v(1)*STP(TIME-10ns) gives a value of 0.0 until 10 nsec has elapsed, then gives v(1)
IF(t,x,y)	x if t is true y otherwise	t is a relational expression using the relational operators shown in **Table 3-5**

Table 3-7 *System Variables*

Variable	Applicability
TIME	Time values resulting from a transient analysis. If no transient analysis is run, this variable is undefined. TIME can only be used in analog behavioral modeling expressions.

Analog Power Supplies

This section describes setup information for DC sources.

The symbols in **Table 3-8** have a DC attribute which can be set to the supply level. See To place the voltage sources on page 2-2 for an example of how to place a power supply.

Table 3-8 *Analog Power Supplies*

Symbol	Symbol Library	Description
VDC	source.slb	voltage source
VSRC	source.slb	voltage source
IDC	source.slb	current source
ISRC	source.slb	current source

Creating Symbols

4

Chapter Overview

This chapter provides information about creating symbols. Symbols are the graphical representation of circuit components.

When you create a new symbol in the symbol editor, you need to define the graphics and pins of the symbol and the symbol attributes. You also must define a device model or subcircuit if the symbol will be simulated and a package definition if it will be used with a board layout package.

Creating a New Symbol on page 4-2 describes how to create a symbol and define graphics and attributes for the symbol.

Linking a Symbol to a Model or Subcircuit Definition on page 4-12 describes how to link a symbol to the model or subcircuit which defines the simulation behavior of the part.

Adding New Symbols from a Vendor on page 4-14 describes how to set up symbols created by a vendor.

Creating a New Symbol

The process of creating a new symbol includes the following:

- Creating the symbol
- Defining symbol graphics
- Defining attributes for the symbol

Each of these are described in the following pages.

Creating a Symbol

There are two ways to create a new symbol:

The symbol library contains graphical descriptions of all devices found in the model library.

- Copy an existing symbol from the symbol library and modify it as needed
- Create a new symbol from scratch

If there is an existing symbol similar to the one you want to create, it is normally quicker and easier to create the new symbol by copying and modifying the existing symbol.

It is recommended that you create a separate symbol library file for new symbols rather than modify any of the original symbol library files.

To copy and modify an existing symbol

1 Select Edit Library from the File menu to access the symbol editor.

It is recommended that changes to symbols be saved in a user library rather than the supplied MicroSim library.

2 The currently selected symbol library is indicated in the title bar of the symbol editor window. If you want the new symbol to be in an existing user library:

  a Select Open from the File menu.

  b Select the library file to open and click on OK.

3 In the symbol editor, select Copy from the Part Menu and click on the Select Lib button.

4 In the Open dialog box, click on the symbol library file which contains the symbol you wish to copy, then click on OK.

5 At the Copy Part dialog box, scroll through the part list in the lower right corner and click on the symbol you want to copy. Make sure that you copy a "base" part rather than an "AKO" part.

An AKO part cannot exist in a symbol library file without the base part that it references. If you want to copy an AKO part, first copy the base, then the AKO.

6 When you click on the desired part, the part name appears in both the Existing Part Name and New Part Name fields. Modify the New Part Name field as desired, then click on OK. The symbol is displayed.

7 Select Attributes from the Part menu. Click on the PART attribute so that it is selected. In the VALUE text box, type the name to display on the symbol. If the symbol has a MODEL attribute, click on it and set the value to the exact name of the model (.MODEL or .SUBCKT name). Make sure attributes have been changed.

Be sure to configure the model library using Library and Include Files on the Analysis menu in Schematics.

8 Select Save As from the File menu (for a new file) or Save (if you opened an existing file in step 2).

To create a new symbol from scratch

1 Select Symbol from the Edit menu to access the symbol editor.

2 The currently selected symbol library is indicated in the title bar of the symbol editor window. If you want the new part to be in a different library:

  a Select Open from the File menu.

  b Select the library file to open and click on OK.

3 Select New from the Part menu to define the part name, description, and other information for the symbol. Click on OK when finished.

Understanding Symbol Graphics

Graphics checklist:

1. Is the origin at the hot-spot of the upper leftmost pin of the symbol?
2. Are all visible pins and graphics contained within the bounding box?
3. Is the bounding box no larger than necessary?
4. Are all pins on grid?

The following graphic information is defined for a symbol:

- Origin
- Bounding box
- Grid spacing

Origin

The origin is used when placing a part and is the center point used when rotating a part. The symbol origin is defined using Origin on the Graphics menu in the symbol editor.

By convention, the origin of each symbol in the symbol library is placed at the hot-spot of the upper leftmost pin on the device. The hot-spot is the point of connection to a wire or to another pin.

The origin is maintained as a point of reference on the schematic. If you change the origin of a symbol, thus changing the location of the symbol graphics and pins, relative to that point, then the symbol graphics and pins are relocated accordingly when a schematic is edited.

Bounding box

The bounding box determines the selection area for a part. When you click on a part in MicroSim Schematics, the area in which you can click and have that part be selected is defined by the bounding box of the symbol. The symbol bounding box is defined using Bbox on the Graphics menu in the symbol editor.

By convention, the bounding box contains the complete symbol (graphics and pins). You will get a warning if you try to save a symbol which has any pins outside the bounding box. All visible pins **must** be contained within the bounding box in order to make proper connections.

The bounding box should be no larger than necessary. If a part’s bounding box is larger than necessary, then part selection in the schematic editor can be difficult for parts placed close together.

The bounding box should only contain the symbol graphics and visible pins.

Grid spacing

The grid spacing in the symbol editor can be changed when higher resolution is desired for creating graphics in a tighter grid. Grid spacing is defined using Display Options on the Options menu in the symbol editor.

The default grid spacing is set at 0.1", and the minimum grid spacing is 0.01". If you change the grid spacing, be sure to set the grid spacing back to the default before placing pins.

Pins must be placed on the grid at 0.1" intervals from the origin of the symbol and at least 0.1" from any adjacent pins. Pins that are not at 0.1" intervals from the origin are considered “off-grid,” and a warning is displayed if you try to save a symbol with one or more pins off-grid.

Defining Part Symbol Attributes

When you use New on the Part menu in the symbol editor to create a new symbol, the PART, MODEL, REFDES, and TEMPLATE attributes are provided as a default set. You can provide any other attributes as needed.

Attributes checklist

1. Does the value of the MODEL attribute match the PSpice .MODEL/.SUBCKT name?
2. Does the TEMPLATE specify the correct number of pins/ nodes?
3. Are the pins/nodes in the TEMPLATE specified in the proper order?
4. Do the pin/node names in the TEMPLATE match the pin names on the symbol?

This section describes the following attributes:

MODEL, below

SIMULATION ONLY, below

TEMPLATE on page 4-7

MODEL

This attribute indicates the name of the model referenced for simulation. The MODEL name should match the name of the .MODEL or .SUBCKT definition of the simulation model as it appears in the PSpice model library file (`.lib`).

For example, your design could include a 2N2222 bipolar transistor with a .MODEL name of Q2N2222. The MODEL attribute on the symbol for that part should be Q2N2222. This MODEL attribute can then be referenced in the TEMPLATE attribute.

In the schematic editor Edit Attributes dialog box, the MODEL attribute is marked with an asterisk and cannot be changed. You can change the MODEL attribute the following ways:

- Use Model on the Edit menu item in the schematic editor to either change the model reference or to create an instance model.
- To edit the underlying model definition of a part, use Model on the Edit menu in the symbol editor.

SIMULATION ONLY

If present, this attribute indicates that the part only has meaning for simulation. The SIMULATION ONLY attribute identifies parts such as voltage and current sources, breakout devices (found in `breakout.slb`), and special symbols (found in `special.slb`).

TEMPLATE

This attribute defines how to create a netlist entry for PSpice. The pin names specified in the TEMPLATE attribute must match the pin names on the symbol. The number and order of the pins listed in the TEMPLATE attribute must match those appropriate for the associated .MODEL/.SUBCKT definition referenced for simulation.

In the schematic editor Edit Attributes dialog box, the TEMPLATE attribute is marked with an asterisk and cannot be changed. To change the TEMPLATE attribute, use Model on the Edit menu in the symbol editor.

The TEMPLATE contains regular characters that are copied to the netlist, and special items such as attributes, that are translated before being copied to the netlist. The translation process is repeated until there are no more attributes left in the template.

Regular characters include alphanumerics, $ and _, white space, and some punctuation marks. An “identifier” is taken to mean an alphabetic character followed by zero or more alphanumeric characters, $ or _.

Attribute names are denoted by an identifier preceded by one of the following special characters:

```
@ ? ~ # &
```

Attribute processing is determined by the character preceding the attribute name:

`@<id>`	replaced by the value of <*id*>. Error if no <*id*> attribute or if no value assigned.
`&<id>`	replaced by the value of <*id*> if <*id*> defined.
`?<id>s...s`	replaced by text between s...s if <*id*> defined.
`~<id>s...s`	replaced by text between s...s if <*id*> undefined.
`#<id>s...s`	like ? but delete rest of template if <*id*> undefined.

Connectors

For board layout purposes, connectors are treated as parts (and may be packaged, in the case of a multi-pin connector). For simulation purposes, connectors do not play any part other than to provide one or more pins where Probe markers might be placed.

Connectors are indicated by having a PKGREF attribute but no TEMPLATE attribute. If you create a connector symbol, be sure to delete the automatically provided TEMPLATE attribute from the attribute list in the symbol editor.

The ? and ~ forms can also take an “else” clause:

```
?<id> s...ss...s
~<id> s...ss...s
```

In the above, “s” can be any “non-regular” character. These conditional attribute constructs can be nested by using different separator characters.

The ^ character is also special. It is replaced by the complete hierarchical path to the device being netlisted.

The sequence `\n` is replaced by a new line. This allows a multi-line netlist entry to be generated from a one-line template.

Pin names are indicated by a % character followed by one or more regular characters. A pin name in the template is replaced by the name of the node connected to that pin. The pin name is everything after the % to a separator (white space or comma). Some characters in pin names are translated to avoid problems with Probe:

<	is replaced by l (L)
>	is replaced by g
=	is replaced by e
\XXX\	is replaced by XXXbar

To enter a % character into the netlist output, enter `%%` in the template.

Any devices generated from a netlist template must start with a letter acceptable to PSpice for that particular kind of device (such as `Q` for a bipolar transistor). To ensure this, templates for devices in the Symbol Library start with a prefix letter, followed by the hierarchical path, and then the reference designator (REFDES) attribute.

For example:

`R^@REFDES` .... for a resistor

We strongly recommend that users adopt this scheme when generating their own netlist templates.

TEMPLATE example: simple resistor (R)

The symbol for a simple resistor has attributes for REFDES and VALUE, both of which are required. It has two pins (1 and 2). A suitable template would be:

```
R^@REFDES %1 %2 @VALUE
```

A device with REFDES = R23 and VALUE = 1k connected to nodes *abc* and *def* would generate:

```
R_R23 abc def 1k
```

TEMPLATE example: voltage source with optional AC and DC specifications (VSRC)

The VSRC symbol has two attributes (AC and DC) and has two pins (+ and -). A suitable TEMPLATE would be:

```
V^@REFDES %+ %- ?DC|DC=@DC| ?AC|AC=@AC|
```

A device with REFDES = V6 connected to nodes *vp* and *vm*, with DC set to 5v and AC undefined would generate:

```
V_V6 vp vm DC=5v
```

or with AC set to 1v would generate:

```
V_V6 vp vm DC=5v AC=1v
```

TEMPLATE example: parameterized subcircuit call (X)

Suppose you have a subcircuit Z that has two pins (a and b), and a subcircuit parameter G. If G is undefined, the value 1000 is to be supplied. You use an attribute, G, to allow the parameter to be changed on the schematic.

A suitable TEMPLATE would be:

```
X^@REFDES %a %b Z PARAMS: ?G|G=@G| ~G|G=1000|
```

A device U33 connected to nodes 101 and 102, and with G set to 1024 would generate:

```
X_U33 101 102 Z PARAMS: G=1024
```

or with G undefined would generate:

```
X_U33 101 102 Z PARAMS: G=1000
```

The TEMPLATE could also be written using the ‘if...else’ form:

```
X^@REFDES %a %b Z PARAMS: ?G|G=@G||G=1000|
```

Pin callout in subcircuit templates

The number and sequence of pins named in a subcircuit's template must agree with the definition of the subcircuit itself. To see how this works, consider the following first line of a (hypothetical) subcircuit definition:

Subcircuit definitions (which start with the `.SUBCKT` statement) are stored in model library files (with the `.LIB` extension).

```
.SUBCKT GIMMICK 10 3 27 2
```

The four numbers following the name `GIMMICK` are *node* numbers. These nodes—and only these—are to be brought out as pins. Thus, this subcircuit has four pins, corresponding to the four nodes shown.

Note that the node numbers need not be in ascending order. (In fact, they need not be numbers; other alphanumeric designations are allowed.)

The remaining lines in the definition include descriptions of *all* the subcircuit's nodes. The first line merely lists those that are brought out as pins.

Now suppose that the symbol definition shows four pins:

```
IN+          OUT+          IN-
          OUT-
```

The *number* of pins on the symbol equals the number of nodes in the subcircuit definition (namely, four).

Now suppose that the correspondence between pin names and nodes is to be as follows:

Node Number	**Pin Name**
10	IN+
3	IN-
27	OUT+
2	OUT-

Then the template might look like this:

```
X^@REFDES %IN+ %IN- %OUT+ %OUT- @MODEL
```

The initial `X` indicates that the template refers to a subcircuit definition.

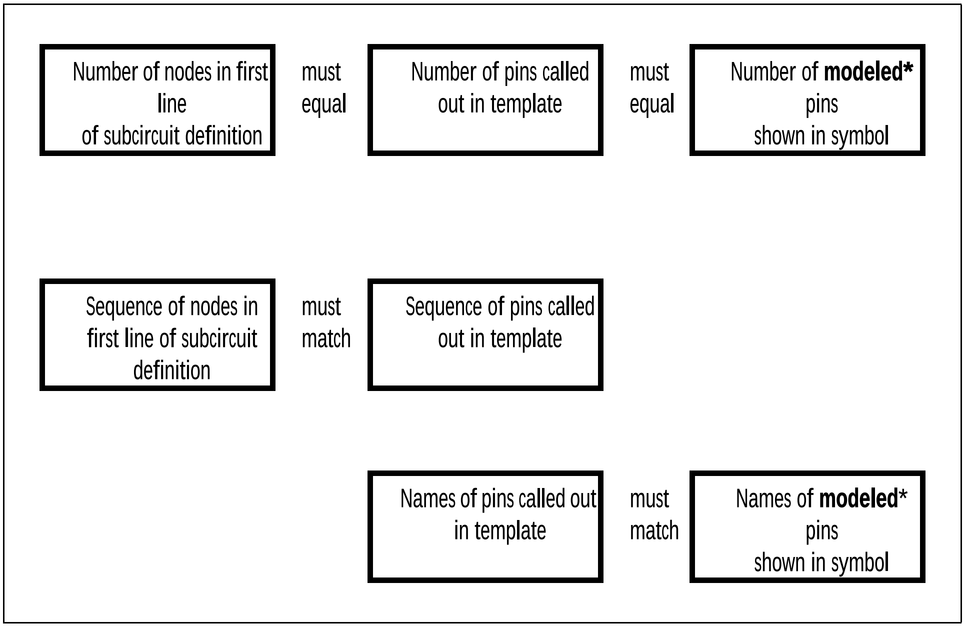
The `^@REFDES` appends the hierarchical path (if any) and the reference designator to the "X" in the netlist.

The final `@MODEL` will be automatically replaced with the value of the MODEL attribute.

Every term beginning with a `%` sign refers to a pin. In fact, each pin entry consists of the pin's name preceded by the `%` sign.

Note that the template contains the correct *number* of pins (four), as well as the correct *sequence* of pin names.

The “rules of agreement” are outlined in Figure 4-1.


Number of nodes in first line of subcircuit definition
must equal
Number of pins called out in template
must equal
Number of **modeled*** pins shown in symbol
Sequence of nodes in first line of subcircuit definition
must match
Sequence of pins called out in template
Names of pins called out in template
must match
Names of **modeled*** pins shown in symbol

* *Unmodeled* pins may appear on a symbol (like the two voltage offset pins on a 741 opamp symbol). Such pins are not netlisted, and *do not appear on the template*.

Figure 4-1 *Rules for Pin Callout in Subcircuit Templates*

Linking a Symbol to a Model or Subcircuit Definition

Each part that you place on your schematic is linked to a model or subcircuit definition which describes the electrical behavior of the part for simulation purposes. This section provides information about how parts are linked to models and subcircuits.

Linking a Symbol to a Model

For a description of the MODEL attribute, see MODEL on page 4-6.

When the electrical behavior of a part is described by a model, the associated model is identified by the MODEL attribute for the part symbol.

To tell PSpice where to find the model library file containing the definition of the simulation model referenced, use Library and Include Files on the Analysis menu in the schematic editor.

In the Library and Include Files dialog box, the files listed must be in a directory on the library search path or be specified with a complete path. You can display or modify the library search path using Editor Configuration on the Options menu in the schematic editor.

Relating Subcircuits to Part Symbols

For a description of the MODEL attribute, see MODEL on page 4-6.

When the electrical behavior of a part is described by a subcircuit, the associated subcircuit is identified by the MODEL attribute for the part symbol.

In the standard symbol and model libraries, the symbol name and corresponding subcircuit name are identical. This convention should be followed when creating user-defined parts with subcircuit definitions.

If the subcircuit has variable input parameters using the PARAMS: construct, the part symbol must have corresponding attributes which can be optionally set on a part instance basis.

For example, the LM7805C subcircuit definition in the model library has two variable parameters: `Av_feedback` and `Value` as defined in the following .SUBCKT statement:

```
.SUBCKT LM7805C Input Output Ground
   x1 Input Output Ground x_LM78XX PARAMS:
+     Av_feedback=1665, R1_Value=1020
```

The equivalent part symbol attributes for this subcircuit are:

```
MODEL = LM7805C
Av_feedback=1665
R1_Value=1020
```

These part symbols also have a TEMPLATE attribute, which defines how Schematics should translate the attributes of a given part instance into an X (subcircuit) device declaration written to the netlist. A LM7805C TEMPLATE is defined in a single line of text as:

For a description of the TEMPLATE attribute, see TEMPLATE on page 4-7.

```
TEMPLATE=X^@REFDES %IN %OUT %COMMON @MODEL
```

X^@REFDES instructs Schematics to substitute the hierarchical (if applicable) reference designator for this part instance prefixed with the letter X, thus producing a simulator-compatible subcircuit device.

Template items preceded by a % character produce simulator compatible node names. The required connecting terminals must be listed in the order specified in the subcircuit definition. In this example, %IN, %OUT, and %COMMON reflect the exact order of the terminals in the LM7805C .SUBCKT definition.

@MODEL is substituted with the subcircuit name LM7805C.

When creating a subcircuit definition for a new part (rather than modifying an existing part symbol or part instance), you need to create a new part symbol (and possibly a package definition), being sure to define all of the appropriate attributes.

Adding New Symbols from a Vendor

This section describes how to add new symbols from a vendor or other source when the user has an existing model library.

For every new device that you want to add to the software, you need to add two and sometimes three distinct items:

- a model or subcircuit definition
- a symbol
- a package definition

The model and symbol are required in all cases, but the package is required only if you are packaging your design to go to a PCB layout program.

Model Library

All .MODEL and .SUBCKT statements are kept in the model library. By convention, this file has a `.lib` extension.

If you receive new models from a vendor, they may be in a single file or in many files that each contain one model or subcircuit. If they are in a single file, simply rename it so that it has a `.lib` extension.

If each model is in its own file, then you can copy them all into a single file using the DOS copy command. For example, if the model files all have a `.mod` extension, enter the following DOS command:

```
copy *.mod mylib.lib
```

This command copies all of the models into the file called `mylib.lib`.

Note *Make sure none of the new model or file names duplicate those already in the MicroSim package.*

Now you need to tell the simulator that this file exists:

1. In Schematics select Library & Include Files from the Analysis menu.
2. Type in the name of your file including the extension in the File Name text box.
3. If you want the models to be global (available for any schematic), select Add Library*. If you want them to be local (only for the current design), select Add Library.

Symbol Library

Now you need to add a symbol for each of the models or subcircuits that you intend to use.

Switch to the symbol editor by selecting Edit Library from the File menu. The title bar at the top of the window should say <new>:<new>. This means that you are editing a new symbol library file and are editing a new symbol.

A quick overview of the structure of a typical symbol library may be helpful. Most symbol libraries are composed of base parts and AKO (a kind of) parts. The base part usually contains the graphical information for the symbol as well as the minimum attributes required to make that symbol functional. The AKO symbol inherits all of the graphics and attributes of the base part but can alter them or add to them. One other important point: an AKO symbol can only reference base parts contained in its own library.

The quickest way to add a new symbol is to copy an existing one and customize its attributes. Let’s assume you have a new set of opamps from a vendor. For this example, we assume that each device has five interface pins. Since all of the symbols will look the same, we want a symbol library that has one base part and an AKO part for each device to be used.

Copying and Modifying a Symbol for a Single Vendor Model

1 Select Edit Library from the File menu to start the symbol editor.

2 If the new symbol is to be placed in an existing library, select Open from the File menu to open the destination library. (Skip this step if the symbol is to be placed in a new symbol library.)

3 Select Copy from the Part menu.

4 Click on the Select Lib button.

5 Scroll through the list of `.slb` files, select `opamp.slb`, and click on OK. A list of symbols is displayed in the parts window. Notice that most of the symbols have an AKO reference after them.

6 To copy a base part:

  a Scroll through the list nearly to the bottom (base parts are always at the bottom of the library).

  b Select a base part (op5, for example). The base part name appears in both the New Part Name and Existing Part Name text boxes.

There are two types of parts in a symbol library: a base part and an AKO part. Only base parts (those without the AKO reference after the part name) can be copied.

  c Enter the name of the new model or subcircuit in the New Part Name text box.

  d Click on OK. The symbol for the part is displayed.

7 Select Attributes from the Part menu and do the following:

  a Type `PART` in the Name text box.

  b Type the name of the model or subcircuit in the Value text box.

  c Click on Save Attr.

  d Type `MODEL` in the Name text box. (The name of the model or subcircuit should still be displayed in the Value text box.)

The TEMPLATE attribute must be defined in accordance with the model or subcircuit definition. See TEMPLATE on page 4-7

  e Click on Save Attr.

8 Select Save As from the File menu (or select Save if an existing library was opened in step 2).

9 Type the name of the new file without the `.slb` extension. Make sure that the new symbol and file names do not duplicate existing file names.

10 When you are asked whether to add this file to the list of Schematics' configured libraries, select Yes. This makes the symbol library available to Schematics.

Copying and Modifying a Symbol for Multiple Vendor Models Using AKO Symbols

Adding a base part

1 Select Edit Library from the File menu to start the symbol editor.

2 If the new symbol is to be placed in an existing library, select Open from the File menu to open the destination library. (Skip this step if the symbol is to be placed in a new symbol library.)

3 Select Copy from the Part menu.

4 Click on the Select Lib button.

5 Scroll through the list of `.slb` files, select `opamp.slb`, and click on OK. A list of symbols is displayed in the parts window. Notice that most of the symbols have an AKO reference after them.

6 To copy a base part:

   a Scroll through the list nearly to the bottom (base parts are always at the bottom of the library).

   b Select a base part (op5, for example). The base part name appears in both the New Part Name and Existing Part Name text boxes.

There are two types of parts in a symbol library: a base part and an AKO part. Only base parts (those without the AKO reference after the part name) can be copied.

   c. Enter the name of the new model or subcircuit in the New Part Name text box.
   d. Click on OK. The symbol for the part is displayed.

7. Select Save As from the File menu (or select Save if an existing library was opened in step TBD2).
8. Type the name of the new file without the `.slb` extension. Make sure that the new symbol and file names do not duplicate existing file names.
9. When you are asked whether to add this file to the list of Schematics' configured libraries, select Yes. This makes the symbol library available to Schematics.

Adding AKO symbols

AKO symbols are useful if a library contains multiple models that can use the same symbol graphics. If only one symbol is needed, you can skip this section.

Your new library now has one base part. Next we will add the first AKO part (the first symbol that can be used for simulation). Assume that the first opamp that you want to add is called LM557.

1. Select New from the Part menu.
2. Enter a description for the part. This can be anything meaningful to you (`inverting OPAMP`, for example).
3. Enter the name of the part. This usually matches the name of the model (`LM557`, for example).
4. Enter `OP5` for the AKO name.
5. Click on OK. The status bar at the top of the screen lists the symbol being edited as LM557.
6. Select Attributes from the Part menu. In the Attributes dialog box, there are four attributes listed for the symbols: REFDES, TEMPLATE, PART, and MODEL. REFDES equals U? by default, which denotes that it is referencing an IC definition. For most cases you do not need to edit TEMPLATE. Both PART and MODEL are blank.
7. Click on PART. The word PART appears in the NAME text box.
8. Enter the name of the symbol (`LM557`) in the VALUE text box.

9 Click on Save Attribute.

10 Repeat the previous two steps for the MODEL attribute.

   MODEL must match exactly the name of the part as defined in the .SUBCKT definition found in your `.lib` file. (Case is not important.)

11 Click on OK.

12 Select Save from the File menu.

You now have one base part and one AKO symbol in your library. At this point, you could return to Schematics and call up the LM557 symbol, place it in your drawing, and simulate with it.

Repeat the procedure for each AKO part that you want to add. If other new devices have different graphics than the op5, add a new base part and proceed as above.

If you need to add symbols over the course of several editing sessions, you need to add one step: after switching to the symbol editor, select Open from the File menu and open the file to which you will be adding new symbols. The status bar at the top of the screen shows which library you are working in at any given time.

Creating Models

5

Chapter Overview

This chapter provides information about creating models. Models define the electrical behavior of schematic components for purposes of simulation.

For a listing of standard library components, refer to the on-line *Library Reference Manual*.

This chapter includes the following sections:

Model Overview

See MODEL on page 4-6 for a description of the MODEL attribute.

Models define the electrical behavior of parts or subcircuits for simulation. A model is linked to a specific part or subcircuit by the MODEL attribute defined for the symbol used by the part/ subcircuit.

The model library shipped with your MicroSim software installation contains descriptions of the electrical behavior for many stock parts using the device model (.MODEL) and subcircuit (.SUBCKT/.ENDS) constructs.

Model definitions describe circuit behavior by a set of variable model parameters which fine-tune the output of the model. The model construct is used for analog devices that are supported by intrinsic simulator device models.

Subcircuit definitions describe the electrical behavior of a part using circuitry. The subcircuit construct supports variable input parameters to fine-tune output. Subcircuits are used to describe analog devices.

Global Models vs. Local Models

Models can be global (available to any schematic) or local (customized for one schematic).

When designing a reusable custom part, it is usually necessary to add a model or subcircuit definition to the model library in conjunction with a symbol and package definition in the symbol and package libraries, respectively. This kind of global definition becomes available to any circuit created in the MicroSim Schematics environment.

It is sometimes necessary to modify the model or subcircuit definition for a part instance in your schematic, thus creating a local version of the definition. An example of this is setting device and lot tolerances on the model parameters for a particular part instance in order to run a Monte Carlo or sensitivity/worst-case analysis.

Creating and customizing device definitions requires knowledge of the model and subcircuit syntax recognized by the simulator, as well as various techniques to facilitate their creation and modification.

See Supported Device Types on page 5-8 for a description of models supported by the Parts utility.

Parts Utility vs. Text (Model) Editor

Models can be defined using the Parts utility or the text editor (sometimes called the model editor).

The Parts utility is useful for characterizing specific models from data sheet curves. The text editor is useful if model parameters are already defined (such as for models from a vendor) or if the model is not supported by the Parts utility.

Note *The Parts utility is not included in*

not
included
in: Basics+

Using the Parts Utility

MicroSim software is shipped with a model library containing device model and subcircuit definitions for thousands of off-the-shelf parts. For most users, the devices in this library are sufficient. However, it is sometimes necessary to define new device models when creating additional parts to be used in your schematic or to test changes to a part’s operational characteristics.

By simply entering data sheet values for the desired semiconductor components, the Parts utility can be used to create new .MODEL and .SUBCKT definitions. These definitions are saved to external model files which can then be used when simulating a schematic.

A difficult areas in using analog circuit simulators is finding accurate models for off-the-shelf parts. The Parts utility is a semiautomatic aid for determining the model parameters for standard devices, such as bipolar transistors, and the subcircuit definitions for more complex models, such as operational amplifiers. The Parts utility helps to convert information from the component manufacturer’s data sheet (without taking measurements of a real device) into parameter values used by PSpice.

You may ask, “Why do I need to model these devices? Won’t the data sheet values work?” Well, yes and no. Yes, for simpler devices such as resistors, which only need the resistance value to have a complete model. No, for more complex devices, especially semiconductor devices. This is because the physical model for predicting how a transistor operates views the transistor from the inside, while the manufacturer provides measurements that show how the transistor operates from the outside. Therefore, a conversion is necessary from data sheet values to physical model parameters.

Data sheet information shows what the manufacturer will guarantee when you buy a part. The device’s operating characteristics will fall within the range specified: a particular part could be near the minimum value of one specification and near the maximum value of another. A typical value is given for some specifications to indicate how most of the devices will operate. Though the Parts utility works with measurements taken from a specific device, it is sufficient to use typical values from the data sheet for most of your simulation work. You may also want to create best/worst case models for checking your design.

For a given device type, the Parts utility displays the data sheet specifications, estimated model parameters, and graphs of the device characteristics. Usually, you will enter or modify data sheet information for each device characteristic. The Parts utility then estimates the model parameters and generates a graph displaying device characteristic behavior. Alternatively, you can edit model parameters directly to investigate their effect on a device characteristic (see Figure 5-2).

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

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

This means that the curve displayed in the Parts utility is not exactly as what is displayed in Probe after a simulation. Models should always be tested and verified using PSpice and fine-tuned if necessary.

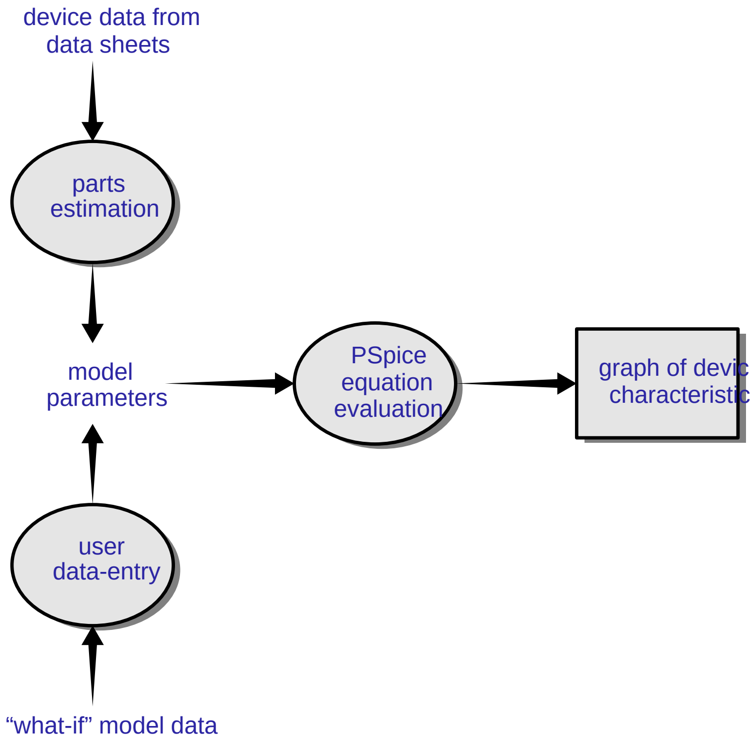

device data from data sheets
parts estimation
model parameters
PSpice equation evaluation
graph of devic characteristic
user data-entry
“what-if” model data

Figure 5-1 *Process and Data Flow for the Parts Utility*

Starting the Parts Utility

The Parts utility can be started from the schematic editor or the symbol editor.

To start the Parts utility from the schematic editor:

1. Select Model from the Edit menu.
2. Click on Edit Instance Model (Parts).

To start the Parts utility from the symbol editor:

1. Select Model from the Edit menu.
2. Click on Edit Model (Parts).

Once it is started, the Parts utility creates instance models or base models, respectively. Models are saved to model files and automatically configured for use with Schematics.

Supported Device Types

The Parts utility produces either a .MODEL definition of the device or a .SUBCKT definition of the device. Devices such as diodes and bipolar transistors are derived from PSpice's intrinsic template models and use the .MODEL statement.

Opamps, on the other hand, are defined as a macromodel or subcircuit whereby a collection of circuit elements are bound together in a single package. Table 5-1 summarizes the device model types supported in the Parts utility.

Note *The model parameter defaults used by the Parts utility are different than those used by the base models intrinsic to PSpice.*

Table 5-1 *Device Models Supported in the Parts Utility*

Device Type	Definition Form	Prefix (PSpice Device Type)
diode	.MODEL	"D"
bipolar transistor	.MODEL	"Q"
IGBT	.MODEL	"Z"
JFET	.MODEL	"J"
power MOSFET	.MODEL	"M"
operational amplifier	.SUBCKT	"X"
voltage comparator	.SUBCKT	"X"
nonlinear magnetic core	.MODEL	"K"
voltage regulator	.SUBCKT	"X"
voltage reference	.SUBCKT	"X"

Parts Model Files (.lib or .mod)

The .MODEL and .SUBCKT definitions generated by the Parts utility are output to one or more model files, depending upon platform. The files are ordinary text files which can be copied or changed with a text editor.

Once a configured model file (usually with the `.lib` extension) is opened, all new and/or modified .MODEL and .SUBCKT definitions are saved to the open model file.

If, from within Schematics, the part instance refers to a base model definition contained in the model library, then a default file named <*schematic name*>`.lib` is opened into which an instance model is created. That is, the base model is copied into the file and its name is changed to a unique name, thus creating the instance model. The instance model name is of the form <*base model name*>-X<*n*> where <*n*> is <*blank* 1 | 2 | ... >. If desired, the currently opened file can be saved to a new file using Save As on the File menu.

Configuring parts model files

Within the schematic editor, Library and Include Files on the Analysis menu allows you to view the list of model files pertaining to your current schematic, or to manually add, delete, or change the model file configuration (see Configuring the Library on page 5-36). The Library Files list box displays all of the currently configured model files. One file is specified per line. Files can be configured as either global to the Schematics environment (using Add Library*) or local to the current schematic (using Add Library). Global files are marked with an asterisk (*) after the file name.

Any model files created by default (<*schematic name*>`.lib`, for example) are automatically configured (added to the list) as local to the current schematic. However, if you create a new model file using Save As on the File menu, the new file must be manually configured into the list.

Note *If you have defined device models with the same names as those in other configured model files and you want to use the new definitions during simulation instead, care must be taken when positioning the file names in the configuration list. If the new files and the existing files are configured with the same scope (either all local or all global), be sure to position the new files before the existing files. Note that PSpice searches locally configured files before the global files. So, if your new model files are configured locally, and the duplicate definitions exist in globally configured files (such as the central model library), your new definitions are still used by PSpice. See* Search order on page 5-38 *for a complete discussion on search order.*

Automatic Model Configuration

1 Start the Parts utility from either the schematic editor or the symbol editor:

From the schematic editor:

a Select the part instance to be modified.

b Select Model from the Edit menu.

c Click on Edit Instance Model (Parts).

Schematics searches the list of configured model files for a match. If a match is found, the model file containing the instance model is opened. If no match is found, it is assumed to be a new device model which is by default saved to <*schematic name*>`.lib`, and the Parts utility is initialized with a *copy* of the base model.

From the symbol editor:

a Edit or create a symbol with the following attributes defined:

PART	Symbol name; it is good practice to have the symbol name match the MODEL name.

MODEL Model name as defined in the Parts utility .

b Select Model from the Edit menu.

c Click on Edit Model (Parts).

2 The main Parts utility window is displayed with a list of the device characteristics and a list of all model parameters. If desired, performance curves can be viewed by selecting a device characteristic then selecting Display from the Plot menu (see Figure 5-3).

To enter device data from the data sheet for any or all of the device characteristics, in any order:

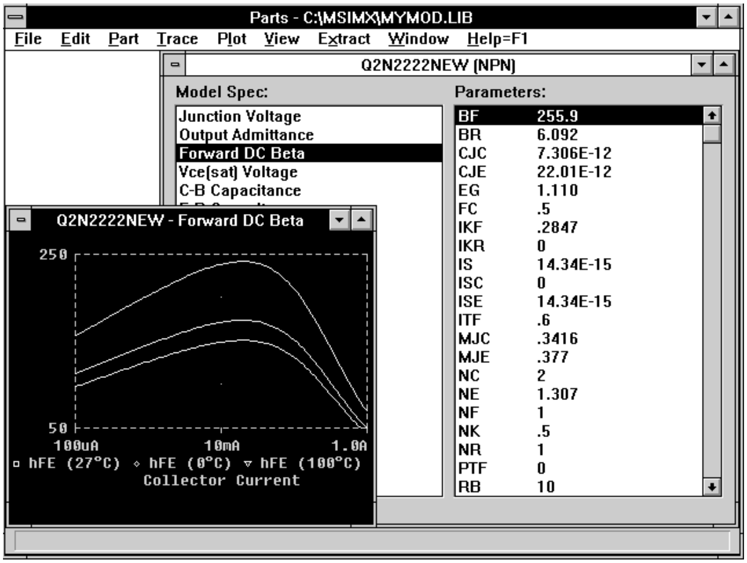
a Select the device characteristic in the Model Spec list box**.**

b Select Spec from the Edit menu.

c Fill in the Edit Model Spec dialog controls and click on Add.

d Click on OK to exit the dialog box.

e Repeat steps a-d for as many characteristics as need specification.



Model Spec:

- Junction Voltage
- Output Admittance
- Forward DC Beta
- Vce(sat) Voltage
- C-B Capacitance

Parameters:

BF	255.9
BR	6.092
CJC	7.306E-12
CJE	22.01E-12
EG	1.110
FC	.5
IKF	.2847
IKR	0
IS	14.34E-15
ISC	0
ISE	14.34E-15
ITF	.6
MJC	.3416
MJE	.377
NC	2
NE	1.307
NF	1
NK	.5
NR	1
PTF	0
RB	10

Figure 5-2 *Parts Utility Window with Data for a Bipolar Transistor*

3. Extract all model parameters for the current specification using Parameters on the Extract menu.
4. Repeat steps 2-3 until the model meets target behaviors.
5. Save the model by selecting Save from the Part menu.
6. Update the model file by selecting Save Library from the File menu.
7. End the Parts utility session by selecting Exit from the File menu.

Manual Model Configuration

Note *Although you can configure models manually as described in this section, it is recommended to use automatic configuration as described in* Automatic Model Configuration on page 5-10 *instead.*

1. Configure Part’s output files into your schematic:
   - a Select Library and Include Files from the Analysis menu.
   - b Enter the file name as <*model file name*>`.lib` (`mycir.lib`, for example).
   - c If the model definitions are for local use in the current schematic, click on the Add Library (or Add Include) button. For global configuration, use Add Library* (or Add Include*) instead.
   - d Click on OK.

If you opened an existing model file to which model definitions were appended, then this step can be skipped.

2. Either part instances or symbols can be modified to reference the new model.

   To change model references locally for a part instance:

   - a Select one or more part instances.
   - b Select Model from the Edit menu.
   - c Click on Change Model Reference and type in the appropriate model name.

To change model references globally for a symbol:

a Enter the symbol editor by selecting Edit Library from the File menu.

b Create or change a symbol definition, making sure to define the following attributes:

See **Chapter 4,**Creating Symbols, for a description of how to create and edit symbols.

PART	Symbol name; it is good practice to have the symbol name match the MODEL name.
MODEL	Model name as defined in the Parts utility .

Creating Model and Subcircuit Definitions by Characterization

This method does not support the variable parameters construct, PARAMS:, the local .PARAM command, or the local .FUNC command. To refine the subcircuit definition for these constructs, use the model editor described in Using the Model Editor (Text Editor) on page 5-21.

The Parts utility is available to facilitate the characterization of new semiconductor devices by simply entering data sheet values for the required stock parts. This utility runs stand-alone, producing complete device model (.MODEL) and subcircuit (.SUBCKT/.ENDS) definitions written to existing or new files which, by default, have the `.lib` extension

Parts Utility Tutorial

In this section, we step through modeling a simple diode device. We first create the schematic for a simple half-wave rectifier. We then run the Parts utility from the schematic editor to create an instance model for the diode in our schematic. The model file containing the new definition will be automatically configured for use with the current schematic.

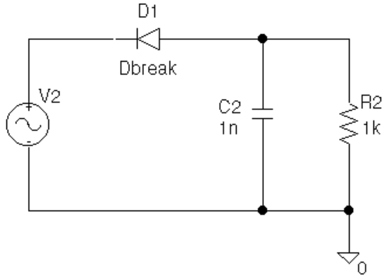


Figure 5-3 *Schematic for a Half-Wave Rectifier*

Creating the half-wave rectifier schematic

1 Start Schematics. From within the schematic editor, draw the simple half-wave rectifier shown in Figure 5-4.

2 Place one each of the following symbols using Get New Part from the Draw menu (reference designator shown in parentheses):

- Dbreak (D1 diode)
- C (C2 capacitor)
- R (R2 resistor)
- VSIN (V2 sine wave source)
- AGND (0 analog ground).

3 Connect the symbols as shown in Figure 5-4 using Wire from the Draw menu or by pressing Ctrl+W.

4 Save the schematic drawing by selecting Save As from the File menu. Type `rectfr` at the prompt. The schematic is saved to `rectfr.sch`.

Note *For simulation, V1 requires a DC, AC, and/or transient specification by appropriately defining its attributes. The Attributes dialog box can be displayed by double-clicking on the V1 symbol. Since we will not be running a simulation in this tutorial, this step is skipped.*

Starting the Parts utility for the D1 diode

1 Click on the D1 symbol. The symbol is highlighted.

2 Select Model from the Edit menu.

3 Click on Edit Instance Model (Parts).

4 Schematics searches the configured set of model library files for a model instance corresponding to this symbol. Since it does not exist, Schematics will ask whether a new default model file, `rectfr.lib` should be created. Click on OK at the prompt to do so. Schematics automatically configures `rectfr.lib` into the set of local model library files.

The parts window is displayed. Notice also that the D1 instance in the schematic references a unique instance model name, Dbreak-X.

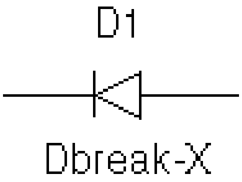
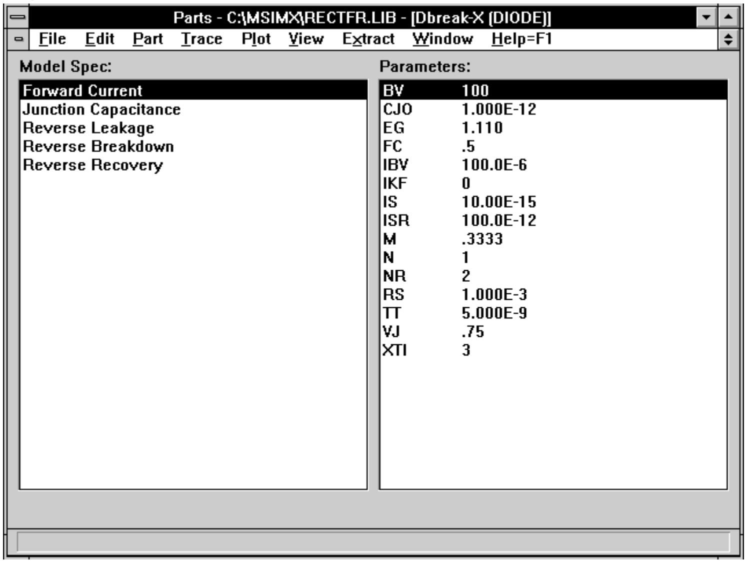


Figure 5-4 *Dbreak-X Instance Model*

Entering data sheet information

The parts window is initialized with the diode model characteristics listed in the Model Spec box and the Dbreak-X model parameter values in the Parameters box as shown in Figure 5-6.



Model Spec:

- Forward Current
- Junction Capacitance
- Reverse Leakage
- Reverse Breakdown
- Reverse Recovery

Parameters:

BV	100
CJO	1.000E-12
EG	1.110
FC	.5
IBV	100.0E-6
IKF	0
IS	10.00E-15
ISR	100.0E-12
M	.3333
N	1
NR	2
RS	1.000E-3
TT	5.000E-9
VJ	.75
XTI	3

Figure 5-5 *Diode Model Characteristics and Parameter Values for the Dbreak-X Instance Model*

Each model characteristic listed in the Model Spec list can be modified with new values from the data sheets. This information is then fit into new model parameter values.

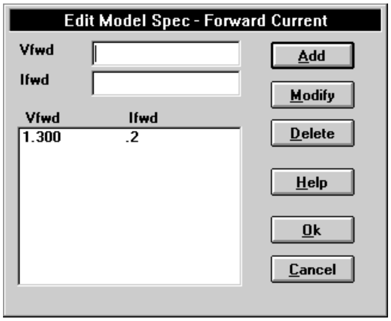
Data sheet information is updated by selecting a model characteristic from the Model Spec list, and entering the values in the displayed edit dialog. There are two types of edit dialogs requiring either device curve data (point pairs) or single-valued data. For the diode, Forward Current, Junction Capacitance, and Reverse Leakage expect device curve data. Reverse Breakdown and Reverse Recovery require single-valued data.

Table 5-2 lists the data sheet information we would like to have incorporated into the Dbreak-X model.

Table 5-2 *Sample Diode Data Sheet Values*

Model Characteristic	Data Sheet Information
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

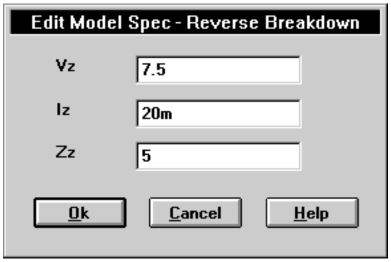


Vfwd	Ifwd
1.300	.2

1. Double-click on the Forward Current entry in the Model Spec box. The Edit Model Spec-Forward Current dialog box is displayed. This dialog box accepts *curve* data.
2. Type `1.3` in the Vfwd text box.
3. Press Tab to move to the Ifwd text box (or click in the Ifwd text box), and type `0.2`.
4. Click on Add or press Enter. The new values will appear in the Vfwd-Ifwd table for the Edit Model Spec-Forward Current dialog box.
5. Click on OK.

To change the values for Junction Capacitance and Reverse Leakage, use the same technique as for Forward Current to invoke the dialog, enter the device curve data, and exit.

To change the Reverse Breakdown characteristic:



Vz 7.5

Iz 20m

Zz 5

1. Double-click on the Reverse Breakdown entry in the Model Spec box. The Edit Model Spec-Reverse Breakdown dialog box is displayed. This dialog accepts single-valued data.
2. Type `7.5` in the Vz text box.
3. Press Tab to move to the Iz text box (or click in the Iz text box), and type `20m`. Note that the Parts utility accepts the same scale factors normally accepted by PSpice.
4. Press Tab to move to the Zz text box (or click in the Zz text box), and type `5`.
5. Click on OK to end this dialog.

Extracting model parameters

To generate new model parameter values from the entered data sheet information, select Parameter from the Extract menu. The new values appear in the Parameters box.

To display the curves corresponding to the five diode characteristics:

1. Select all of the entries in the Model Spec box by clicking on Forward Current and dragging the mouse down to the end of the list.
2. Select Display from the Plot menu. Any one of the plots can be brought to the foreground by clicking on its title bar (or anywhere in the plot window). Any plot can be freely moved (dragged) to a new position in the parts window. Pan and zoom functions are available on the View menu; axes can be scaled using the Plot menu.

   A few of the plots are shown in Figure 5-7, after having moved them to new locations. (Arrange on the Window menu can also help with this.)

By default, device curves are computed at 27°C. For any given characteristic, additional curves can be added to the plot at other temperatures. To do this for Forward Current:

1. Click on the Forward Current plot window to bring it to the foreground.
2. Select Add from the Trace menu or press Ins.
3. Enter the new temperature as `100` (in °C). Click on OK. The Forward Current plot should now appear as in Figure 5-8.

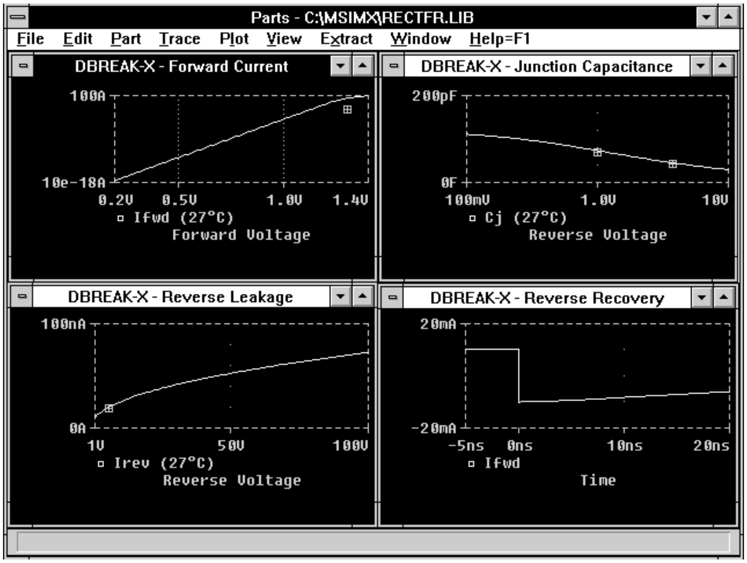


Figure 5-6 *Assorted Device Characteristic Curves for a Diode*

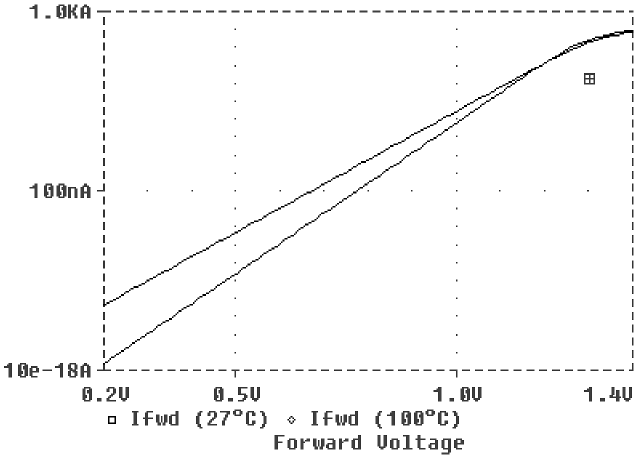
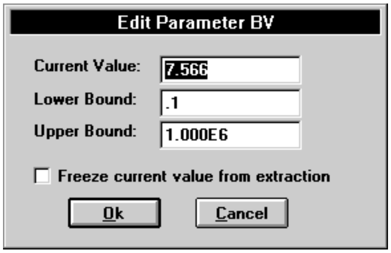


Figure 5-7 *Forward Current Device Curve at Two*

Completing the model definition

The model can be refined by modifying the data sheet information as described before or by editing model parameters directly. Individual model parameters can be selected for update by double-clicking on the corresponding entry in the Parameters box of the main parts window.



Current Value: 7.566
Lower Bound: .1
Upper Bound: 1.000E6
☐ Freeze current value from extraction

For instance, double-clicking on BV brings up the Edit Parameter BV dialog box. This dialog gives you the opportunity to define a valid range for the model parameter and whether it should be excluded from the extraction process. For now, we will leave the model parameters at their current settings.

To save the current model parameter values to our new device model for use in our schematic:

1. Select Save from the Part menu. This saves the current model definition to the list of model definitions contained in `rectfr.lib`.
2. Select Save from the File menu to write the latest model file updates to disk.

We can now return to Schematics and proceed to set up a simulation.

Note *Using Save on the Part menu and Save on the File menu ensure that the model instance name and model file name remain the same as those created by Schematics.* ***This automatically configures the new model definition into our schematic without further intervention.*** *If we had chosen to change the model instance name using Save As on the Part menu, then we would need to manually change the model reference for the D1 symbol by selecting Model from the Edit menu and clicking on Change Model Reference. If we had created a new model file using Save As on the File menu, we would need to configure the new model file using Library and Include Files on the Analysis menu.*

Subcircuits

Create Subcircuit on the Tools menu in the schematic editor creates a subcircuit definition for the current level of hierarchy in your schematic, including all lower levels. This is written to a file named <*schematic name*>`.sub`. Any named interface ports at the current level of hierarchy are mapped into the terminal nodes in the .SUBCKT statement.

To create a subcircuit definition for an existing schematic

1. Place interface ports (IF_IN for input ports and IF_OUT for output ports) in your schematic using Get New Part on the Draw menu.
2. Save the schematic using Save on the File menu.
3. Generate the subcircuit definition using Create Subcircuit on the Tools menu.
4. Using the symbol editor, create a symbol to accompany the subcircuit definition.

   See **Chapter 4,**Creating Symbols, for a description of how

5. Make the model file containing the subcircuit definition available to PSpice by configuring it as either a library or include file using Library and Include Files on the Analysis menu.

   See Configuring the Library on page 5-36, for a description of the Library and Include File

6. If necessary, further refine the subcircuit definition for the new symbol or for a part instance in a schematic using the model editor.

   For example, you could extend the subcircuit definition using the variable parameters construct, PARAMS:, or the local .PARAM or .FUNC commands.

   See Using the Model Editor (Text Editor) on page 5-21, for a description of subcircuit definition with the model editor.

Using the Model Editor (Text Editor)

Schematics supports general model and subcircuit editing by providing the model editor for manipulation of .MODEL and .SUBCKT definitions in the model library, or in model files that are local to a schematic. The model editor allows you to edit definitions, create new definitions, associate definitions with symbols, or derive new definitions from existing ones.

Implementation of Model Editing

Both the schematic editor and the symbol editor have a Model menu item on the Edit menu which allows you to edit the .MODEL or .SUBCKT definition for a part. The behavior of this menu item depends on whether it is run from the schematic editor or from the symbol editor.

The schematic editor modifies local instances of a part for a given schematic, while the symbol editor modifies the global definition of a part in the model library. These definitions are saved to model files and automatically configured as local and global library files, respectively (as shown in the Library and Include Files dialog box).

The model editor selects model or subcircuit definitions by attempting to match the MODEL attribute value of a part to the .MODEL or .SUBCKT name in a library or include file. The local list specified in the Library and Include Files dialog is searched before the global list specified in the same dialog. The first model or subcircuit name that matches the value of the MODEL attribute is the one used by the model editor.

Editing Model Definitions in the Schematic Editor

From within the schematic editor, selecting Model from the Edit menu makes an individual copy of a model or subcircuit representation, allowing you to modify a local copy. By default, this new model is named <*old model name*>-X. Changes only apply to that specific part instance in the schematic or any other part instance in the same schematic that references the new model. By default, these individual instance models or subcircuits are written to a file with the name <*schematic name*>`.lib` which is configured for use with the current schematic only. The default model/subcircuit name and model file name can be modified.

To edit a model or subcircuit definition from within the schematic editor, first select a part or parts, then select Model from the Edit menu. The Edit Model dialog box is displayed if the part has a MODEL attribute.

If a MODEL attribute exists, then the dialog box gives you the choice of changing the name of the model/subcircuit referenced by this part or editing an instance model/subcircuit (copy) of that part. Suppose, for example, that you are in the schematic editor and have placed an instance of a Q2N2222 whose model parameters you want to modify or to which you want to add DEV and/or LOT tolerances for Monte Carlo analysis. When you select this part and then select Model from the Edit menu, the Edit Model dialog box is displayed.

The Change Model Reference button initiates a dialog that allows you to enter the name of another model or subcircuit for the current part instance. The Edit Instance Model (Text) command allows you to edit a copy of the model or subcircuit definition.

The schematic editor does not allow you to edit the underlying base model or subcircuit definition of the symbol; this must be done from the symbol editor. If an instance model/subcircuit (copy) does not already exist, the schematic editor creates a new instance definition that is a copy of the original model/subcircuit. The only difference between the copy and the original is that the schematic editor appends a -X<*n*> to the model/subcircuit name to make the name unique, where <*n*> is

<*blank* | 1 | 2 | ... >. If an instance model/subcircuit already exists, it is displayed in the edit dialog.

Editing Model Definitions in the Symbol Editor

From within the symbol editor, the model or subcircuit representation associated with a symbol maintained in the symbol library is available for edit. Changes are made to the actual model or subcircuit definition, and have global scope. By default, these changes are written to a file called `user.lib` which is configured for global access. Thus, model/subcircuit changes made in the symbol editor affect all instances of the corresponding symbol, no matter what schematic the symbol is used in.

When select Model from the Edit menu in the symbol editor, the Edit Model dialog box is displayed with the base model or subcircuit definition associated with that part, if one exists. Otherwise, you are presented with a blank dialog box in which you can create a new model or subcircuit definition. From within the symbol editor, you are free to change the names of the model/subcircuit and model file.

Model Editor Dialog Box

The model or subcircuit definition is presented in a text-edit work area (a mini text editor). If you are editing a model definition, then the text-edit work area will contain one parameter per line. This makes it easier to add DEV/LOT tolerances to model parameter lines for Monte Carlo or worst-case analysis.

The Expand AKO button will search through any AKOs on which the model is based, adding the new parameters to this model. In essence, this command “flattens” a hierarchy of AKOs into one model in which you can see all the parameters.

Subcircuit definitions are brought into the text-edit work area in the same format as read from the model file. All of the comments immediately before or after the model or subcircuit definition are included for editing. Note that this will sometimes mean that comments intended for the previous or next model definition will be included in the edit dialog.

Model Editing Example

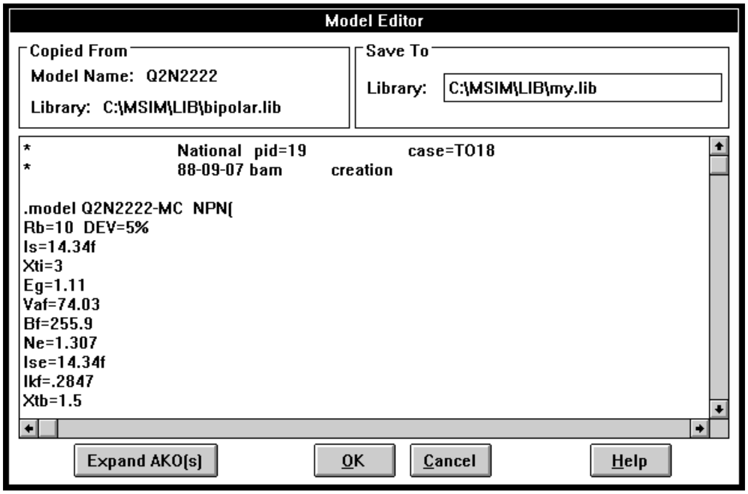
Suppose you have a schematic named `my.sch` which contains several instances of a Q2N2222, and you are interested in the effect of base resistance spread on one specific device—Q6, for example. To investigate this, you must select Q6 in your schematic and Model from the Edit menu from within the schematic editor. The Edit Model dialog box is displayed with the current value of Q6’s MODEL attribute: Q2N2222.

Clicking on the Edit Instance Model (Text) button, the model editor automatically creates a copy of the Q2N2222 model definition with the default name of Q2N2222-X. A dialog box is displayed to allow you to edit the newly copied definition.

You can edit the model definition to specify a tolerance on the parameter(s) on which a Monte Carlo analysis is to be performed. In Figure 5-9, a tolerance of 5% has been added to the Rb parameter. You must also specify a new part name and model file, or use the defaults provided. Figure 5-9 shows that Q2N2222-MC has been entered for the new model name. The model file name `my.lib` remains unchanged.

When you click on OK, the model editor writes the model definition to `my.lib` in the current working directory. If `my.lib` does not already exist, it is created.

If this model file is a new file, it is automatically added to the Library Files list box in the Library and Include Files dialog for the current schematic. If you select Library and Include Files from the Analysis menu, you see the standard `nom.lib` file that is part of the global model library (annotated with an asterisk) and the newly created `my.lib` which is local to the current schematic (no asterisk).



Copied From
Model Name: Q2N2222
Library: C:\MSIM\LIB\bipolar.lib

Save To
Library: C:\MSIM\LIB\my.lib

```
*                      National  pid=19            case=TO18
*                      88-09-07 bam        creation

.model Q2N2222-MC  NPN(
Rb=10  DEV=5%
Is=14.34f
Xti=3
Eg=1.11
Vaf=74.03
Bf=255.9
Ne=1.307
Ise=14.34f
Ikf=.2847
Xtb=1.5
```

Figure 5-8 *Model Editor Dialog Box for Q2N2222 with a DEV Tolerance Set on Rb*

Now you can select Setup from the Analysis menu to set up the parameters as required for a Monte Carlo or worst-case analysis. You are then ready to select Simulate from the Analysis menu to run the simulations.

If you want to change the model reference for this part back to Q2N2222, you can select the part in the schematic editor, select Model from the Edit menu, click on Change Model Reference, and type in the original model name.

Breakout Parts

For non-stock passive and semiconductor devices, Schematics provides a set of “breakout” parts designed for customizing model parameters for simulation. These are useful for setting up Monte Carlo and worst-case analyses with device and/or lot tolerances specified for individual model parameters.

Another approach is to use the model editor to derive an instance model and customize this. For example, you could add device and/or lot tolerances to model parameters.

Basic breakout part names consist of the intrinsic PSpice device letter from the set [BCDJKLMQRSTWZ] plus the suffix BREAK. By default, the model name is the same as the part name and references the appropriate device model with all parameters set at their default. For instance, the DBREAK part references the DBREAK model which is derived from the intrinsic PSpice D model (.MODEL DBREAK D).

Resistors, capacitors, and inductors

Also see Resistors, Capacitors, and Inductors on page 3-2.

For breakout parts RBREAK, CBREAK, and LBREAK, the effective value is computed from a formula that is a function of the specified VALUE attribute.

Table 5-3 *Resistors, Capacitors, and Inductors*

Part Type	Symbol Name	Symbol Library File	Attribute	Description
capacitor (PSpice ‘C’ device)	CBREAK	breakout.slb	VALUE	capacitance
			IC	initial voltage across the capacitor during bias point calculation
			MODEL	CAP model name
inductor (PSpice ‘L’ device)	LBREAK	breakout.slb	VALUE	inductance
			IC	initial current through the inductor during bias point calculation
			MODEL	IND model name
resistor (PSpice ‘R’ device)	RBREAK	breakout.slb	VALUE	resistance
			MODEL	RES model name

Using the KBREAK symbol

The inductor coupling symbol, KBREAK, can be used to couple up to six independent inductors on a schematic. A MODEL attribute is provided for using nonlinear magnetic core (CORE) models, if desired. By default, KBREAK references the KBREAK model contained in `breakout.lib`; this model, in turn, uses the default CORE model parameters.

Table 5-4 *KBREAK Symbol*

Part Type	Symbol Name	Symbol Library File	Attribute	Description
inductor coupling (PSpice ‘K’ device)	KBREAK	breakout.slb	COUPLING	coupling factor
			Li	inductor reference designator

The KBREAK symbol can be used to:

- Effect linear coupling between inductors.
- Reference a CORE model in a configured model library file.
- Define a user-defined CORE model with custom model parameter values.

The dot convention for the coupling is related to the direction in which the inductors are connected. The dot is always next to the first pin to be netlisted. For example, when the inductor symbol, L, is placed without rotation, the dotted pin is the left one. Rotate on the Edit menu (Ctrl+ r), rotates the inductor +90°, making this pin the one at the bottom.

Using the KBREAK symbol in a schematic

1. Draw the schematic and assign the desired reference designator names to all of the part instances. To change default names, double-click on the reference designator label, and enter a new name in the Edit Reference Designator dialog box.
2. Place a coupling symbol on the schematic for each group of coupled inductors. Notice that the symbol has no pins.
3. Define the coupled inductors: Double-click on the coupling symbol’s box with the enclosed “K,” and enter the reference

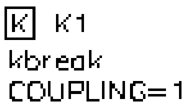


Figure 5-10 *KBREAK Symbol*

designator names for the coupled inductors as values for Li (i=1,2,...,6).

Nonlinear coupling is not included in PSpice Basics

For nonlinear coupling: L1 must have a value; the rest may be left blank.

For linear coupling: L1 and at least one other Li must have values; the rest may be left blank.

4. Change the coupling symbol’s MODEL attribute reference:
    - a click on the symbol’s box with the enclosed “K.”
    - b Select Model from the Edit menu.
    - c Click on Change Model Reference in the Edit Model dialog box.
    - d Change the name as required in the Model Name text box. These rules apply:

      For nonlinear coupling: The model must reference a CORE model such as those contained in `magnetic.lib` or other user-defined models.

      For linear coupling: The model reference must be blank.

5. Set the value of the COUPLING attribute to the value of the coupling factor, K.

6. Set the value (VALUE attribute) for each of the coupled inductors (L symbol instances). These rules apply:

   For nonlinear coupling: VALUE is set to the number of windings.

   For linear coupling: VALUE must be in Henries.

Transformers

not included in: Basics +

Table 5-5 lists attributes that can be set per instance of a nonlinear transformer part. Also see Transformers on page 3-4.

Table 5-5 *Transformers*

Part Type	Symbol Name	Symbol Library File	Attribute	Description
transformer (PSpice ‘K’ and ‘L’ devices)	XFRM_NONLINEAR	breakout.slb	L1_TURNS L2_TURNS	number of turns on each winding
			COUPLING	coefficient of mutual coupling (must lie between 0 and 1)
			MODEL	nonlinear CORE model name

To use XFRM_NONLINEAR, the MODEL attribute must be set appropriately. However, the MODEL attribute of a given instance can be modified to reference a CORE model provided in the `magnetic.lib` model library file or in a user-defined model file.

See *Using the Inductor Coupling Symbols* in the *Application Notes Manual* for a description of techniques for creating additional transformer configurations.

Ideal switches

Table 5-6 summarizes the available switch part types provided in the `breakout.slb` symbol library. The switches may be voltage or current-controlled. To create a time-controlled switch, simply connect the switch control pins to a voltage source with the appropriate voltage vs. time values (transient specification).

Table 5-6 *Ideal Switch Parts in breakout.slb*

Part Type	Symbol Name	Model Type
Voltage-Controlled Switch (PSpice ‘S’ device)	SBREAK	VSWITCH
Current-Controlled Switch (PSpice ‘W’ device)	WBREAK	ISWITCH

The VSWITCH and ISWITCH models define the on/off resistance and the on/off control voltage or current thresholds. These switches have a finite “on” resistance and “off” resistance, and change smoothly between the two as their control voltage (or current) changes. This behavior is important, allowing PSpice to find a continuous set of solutions to the circuit being simulated. In practice, the “on” resistance may be made very small compared to the other circuit impedances, and the “off” resistance may be made very large compared to the other circuit impedances.

As with current-controlled sources (F, FPOLY, H, and HPOLY), WBREAK contains a current-sensing voltage source. When netlisted, WBREAK generates two device declarations to the circuit file set:

- one for the controlled switch
- one for the independent current-sensing voltage source

If you choose to create a new symbol for a current-controlled switch (with, for example, different on/off resistance and current threshold settings in the ISWITCH model), the TEMPLATE attribute must account for the additional current-sensing voltage source.

Semiconductor Parts

Semiconductor parts can be divided into two classifications. One class is described by model definitions derived from intrinsic PSpice device model types: diodes, bipolar transistors, junction field effect transistors (JFETs), metal-oxide-silicon field effect transistors (MOSFETs), gallium arsenide field effect transistors (GaAsFETs), and insulated gate bipolar transistors (IGBT). The other class is described by subcircuit definitions: opamps, silicon controlled rectifiers (SCRs), triacs, unijunction transistors (UJTs), and more.

Intrinsically modeled semiconductors

Table 5-7 lists the semiconductor parts that reference models derived from intrinsic PSpice device model types. The varieties of breakout devices in a given technology vary according to the kind of model referenced and/or the number of pins. Table 5-8 summarizes the classes of semiconductor models.

Operating temperature can be set to one that is different from the global circuit temperature by defining one of the model parameters, T_ABS, T_REL_GLOBAL, or T_REL_LOCAL. Additionally, model parameters can be assigned unique measurement temperatures using the T_MEASURED model parameter. Refer to the *PSpice Reference Manual* for a description of model parameters.

Table 5-9 lists the attributes that can be set for the semiconductor breakout parts. The AREA attribute provided with the GaAsFET, diode, IGBT, JFET, and bipolar transistor parts allows equivalent part instances to be scaled relative to one another. For example, instance Q1 of the QBREAKL part, with AREA set to 1.5, will have 1.5 times the junction capacitance, 1.5 times the base-emitter conductance, and so on, of another QBREAKL instance with AREA left unset.

Table 5-7 *Intrinsically Modeled Semiconductor Parts*

Part Type	Symbol Name	Model Type	Symbol Library File
Bipolar Transistor (PSpice ‘Q’ device)	QBREAKL	LPNP	breakout.slb
	QBREAKN QBREAKN3 QBREAKN4	NPN	
	QBREAKP QBREAKP3 QBREAKP4	PNP	
Diode (PSpice ‘D’ device)	DBREAK DBREAK3 DBREAKCR DBREAKVV DBREAKZ	D, X	breakout.slb
GaAsFET (PSpice ‘B’ device)	BBREAK	GASFET	breakout.slb
IGBT (PSpice ‘Z’ device)	ZBREAKN	IGBT	breakout.slb
JFET (PSpice ‘J’ device)	JBREAKN	NJF	breakout.slb
	JBREAKP	PJF	
MOSFET (PSpice ‘M’ device)	MBREAKN MBREAKN3 MBREAKN4	NMOS	breakout.slb
	MBREAKP MBREAKP3 MBREAKP4	PMOS	

Note *The IGBT model is not supported by PSpice Basics.*

Table 5-8 *Semiconductor Model Types*

Device Type	Model Type	Basis for Enhanced PSpice Model
GaAsFET	GASFET, Level 1	Curtice N-channel GaAsFET
	GASFET, Level 2	Raytheon (or Statz) N-channel GaAsFET
	GASFET, Level 3	Triquint N-channel GaAsFET
	GASFET, Level 4	Parker-Skellern MESFET
	GASFET, Level 5	TOM-2

Note *GASFET levels 1, 3, 4, and 5 are not supported by PSpice Basics.*

Table 5-8 *Semiconductor Model Types (continued)*

Device Type	Model Type	Basis for Enhanced PSpice Model
Diode	D	SPICE2G6 diode
JFET	NJF	Schichman-Hodges N-channel JFET
	PJF	Schichman-Hodges P-channel JFET
MOSFET	NMOS, Level 1	Schichman-Hodges N-channel MOSFET
	NMOS, Level 2	Grove-Frohman N-channel MOSFET
	NMOS, Level 3	Semi-empirical short channel N-channel MOSFET
	NMOS, Level 4	BSIM N-channel MOSFET
	NMOS, Level 6	BSIM3, Berkeley version 2.0, N-channel MOSFET
	PMOS, Level 1	Schichman-Hodges P-channel MOSFET
	PMOS, Level 2	Grove-Frohman P-channel MOSFET
	PMOS, Level 3	Semi-empirical short channel P-channel MOSFET
	PMOS, Level 4	BSIM P-channel MOSFET
	PMOS, Level 6	BSIM3, Berkeley version 2.0, P-channel MOSFET
Bipolar Transistor	NPN	Gummel-Poon/Ebers-Moll NPN transistor
	PNP	Gummel-Poon/Ebers-Moll PNP transistor
	LPNP	Gummel-Poon/Ebers-Moll lateral PNP transistor
IGBT	NIGBT	N-channel IGBT

Note *The IGBT model is not supported by PSpice Basics.*

Table 5-9 *Semiconductor Part Attributes*

Symbol Name	**Attribute**	**Description**
BBREAK	AREA MODEL	area scaling factor GASFET model name
DBREAK DBREAKCR DBREAKVV DBREAKZ	AREA MODEL	area scaling factor D model name
JBREAKN JBREAKP	AREA MODEL	area scaling factor NJF or PJF model name
QBREAKL QBREAKN QBREAKN3 QBREAKN4 QBREAKP QBREAKP3 QBREAKP4	AREA MODEL	area scaling factor NPN, PNP, or LNP model name
DBREAK3	MODEL	D model name
MBREAKN MBREAKN3 MBREAKN4 MBREAKP MBREAKP3 MBREAKP4	L W AD AS PD PS NRD NRS NRG NRB M MODEL	channel length channel width drain diffusion area source diffusion area drain diffusion perimeter source diffusion perimeter relative drain resistivity (in squares) relative source resistivity (in squares) relative gate resistivity (in squares) relative substrate resistivity (in squares) device multiplier (simulating parallel devices) NMOS or PMOS model name

Table 5-9 *Semiconductor Part Attributes (continued)*

Symbol Name	Attribute	Description
ZBREAKN	AGD	gate-drain overlap area
	AREA	area of the device
	KP	MOS transconductance
	TAU	ambipolar recombination lifetime
	WB	Metallurgical base width
	MODEL	NIGBT model name

Note *The ZBREAKN symbol is not supported by PSpice Basics.*

Configuring the Library

The Schematics environment can be set up such that all or any subset of model files are available for reference from any schematic. These files have global scope in the Schematics environment meaning that the simulator will, by default, search these model files for unresolved device definitions. Conceptually, the model files with global status constitute the model library and should be stored in a centralized location. See Library directory paths on page 5-39 for how to specify the directory paths to these files.

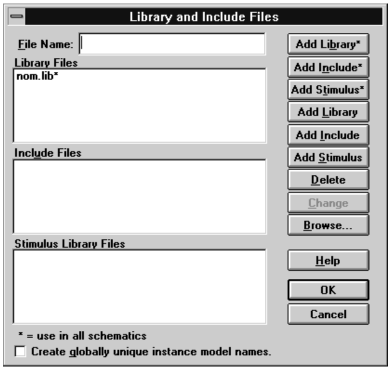
In other cases, it may be desirable to have model files with local impact to one schematic. These model files are not considered part of the model library, and are stored, by default, with the schematic for which they were originally created. The simulator will search the local model files before those in the model library when the schematic referencing them is being simulated.

Global and local model files

The Library and Include Files dialog presents two list boxes to identify model and include files pertaining to your schematic (named Library Files and Include Files, respectively). One file is specified per line. Files are grouped into two distinct classes: global to the Schematics environment and local to a given schematic. Global files are marked with an asterisk (*) after the file name.

A third list box contains stimulus files. See Configuring Stimulus Files on page 10-5 for more information.

At installation, all of the files referenced in `nom.lib` are global to the Schematics environment. The Library and Include Files dialog box (accessed from the Analysis menu) displays the `nom.lib*` entry in the Library Files list box. The asterisk to the right of the file name indicates that `nom.lib` has global status and is available to all schematics.


Library Files: nom.lib*

* = use in all schematics

☐ Create globally unique instance model names.

New model files can be added to either the library or include list by entering file names in the File Name text box and then clicking on the Add Library, Add Include, Add Library*, or Add Include* button. The commands with an asterisk configure the file as a global file. By configuring model files into the Library Files list box, the simulator will automatically generate an index file (if an up-to-date index does not already exist) and use the index to access only the model and/or subcircuit definitions relevant to the simulation. Model files configured into the Include File list box are explicitly written to the circuit file set and read by the simulator in their entirety.

The Add Library* button is used to incorporate new model files into the model library (with global status). For instance, to make the device definitions contained in the user-defined model file `mymodels.lib` available to the simulator at all times, this file name must be added to the list of globally accessible model files. This requires typing its name in the File Name text box and clicking on Add Library*. After doing this, a new global entry will appear in the library files list as `mymodels.lib*`. Note that this technique avoids any additions to the `nom.lib` file.

Additional model files can be added, old files deleted, global/ local status modified, or the library/include status modified through this same dialog. The Delete button deletes the selected file from its list box. The Change button copies the selected file from the list box up to the edit control and deletes it from the list, allowing you to modify it. The Browse button brings up a dialog allowing you to browse the directory structure and select a file name.

To set up model files for local use with the current schematic, you can use the Add Library button (no asterisk). Suppose your current schematic is called `mydesign`, and `mymodels.lib` contains device models specifically for `mydesign`. `Mymodels.lib` should be added to the library list using Add Library. This produces the `mymodels.lib` entry in the library files list without an asterisk, indicating local file status. When simulating `mydesign`, the simulator searches this model file before those with global status.

Note *Care should be taken when configuring model files with the Add Include* (global) and Add Include (local) buttons in this dialog box. This method can have disadvantages when the model file is large. Rather than selecting only the model and subcircuit definitions needed, the simulator will load the entire file into memory. However, when developing device models and/or subcircuits, it can be useful to initially treat the model file as an include file. This avoids the rebuilding of index files every time the model file changes. Then, once the model definitions are stable, the model file can be reconfigured as a library file. The simulator will build the associated index file once. Henceforth, the simulator will read in only those definitions required for the simulation.*

Search order

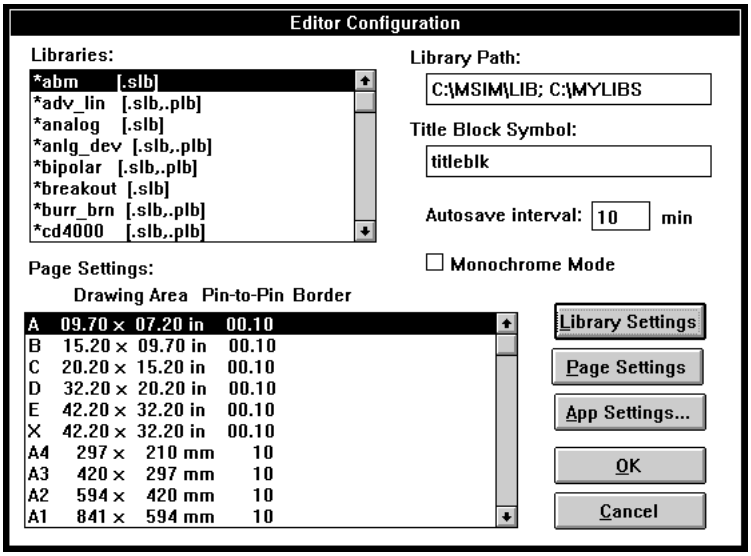
The simulator always searches local model files before global model files in the model library. However, within each class of local or global files, the order in which they are specified in the Library Files list box dictates the order in which the simulator will search the files to resolve references. Optimizing the search order can significantly reduce search time. Where device names are alike across model files, search order will determine which of the devices is selected for simulation. The simulator always uses the first instance.

In the Library and Include Files dialog box (accessed from the Analysis menu), new files are always added after the currently selected item in the Library Files list box. To add a file name in a certain position in the list, select the file name in the list above where you want the new file name to placed, then select the Add_Library* (global) or Add_Library (local) command as required.

If you have configured your model files by listing multiple `.lib` commands within a single file (like `mymodels.lib`), and search order is critical, then you will need to manually edit the file contents with your platform’s text editor. In the standard model library, only `diodes.lib` and `ediodes.lib` (European manufactured diodes) have identically named device definitions. If your schematic uses a device out of one of these files, you must insure that the model file containing the definition of choice is listed first. If your system is configured to use `nom.lib` for the model library, and the file order is not in accordance with your circuit requirements, then you will have to edit this file.

Library directory paths

For model files without explicit path names, the simulator first searches the directory where the current schematic resides, then steps down the list of directories specified in the Library Path text box of the dialog under Editor Configuration (accessed from the Option menu).



Libraries:

*abm	[.slb]
*adv_lin	[.slb,.plb]
*analog	[.slb]
*anlg_dev	[.slb,.plb]
*bipolar	[.slb,.plb]
*breakout	[.slb]
*burr_brn	[.slb,.plb]
*cd4000	[.slb,.plb]

Library Path: C:\MSIM\LIB; C:\MYLIBS

Title Block Symbol: titleblk

Autosave interval: 10 min

☐ Monochrome Mode

Page Settings:

	Drawing Area	Pin-to-Pin Border
A	09.70 × 07.20 in	00.10
B	15.20 × 09.70 in	00.10
C	20.20 × 15.20 in	00.10
D	32.20 × 20.20 in	00.10
E	42.20 × 32.20 in	00.10
X	42.20 × 32.20 in	00.10
A4	297 × 210 mm	10
A3	420 × 297 mm	10
A2	594 × 420 mm	10
A1	841 × 594 mm	10

To ensure that model files added to the model library will be located, it is necessary to do one of the following:

- Place the model files in the standard library directory (that is, where the standard symbol and model files are stored)
- Update the library path definition to include any new directories
- Use the full path name when adding the model file to the Library Files list box in the Library and Include Files dialog box described above

We recommend that model files belonging to the model library (global status) reside in a centralized location. If this location includes more than one directory, the directory paths can be listed in the Library Path text box. Entries must be separated by semicolons.

For example, while the standard model files shipped with the MicroSim simulation software might reside in C:\MSIM\LIB, a second directory, for example C:\MYLIBS, might contain all of your custom model files. To ensure that the simulator searches both C:\MSIM\LIB and C:\MYLIBS for model files, type the new directory path name in the Library Path text box of the Editor Configuration dialog box (accessed from the Options menu) and click on OK.

Schematics and simulator interaction

When generating the circuit file set for simulation, Schematics outputs a .LIB or .INC command for each model file specified in the Library Files or Include Files list box in the Library and Include Files dialog box (accessed from the Analysis menu). The list order is preserved, thus directing the simulator to search the specified model files for unresolved model and subcircuit references in the given order.

When the simulator runs, only those models and subcircuits referenced by the parts in your schematic are actually read from the model files and put into main memory (RAM). This avoids using up main memory for definitions which are not used. It also means that there is no memory penalty for having large model files. Read-in time is also kept to a minimum.

Analog Behavioral Modeling

6

Chapter Overview

This chapter describes how to use Analog Behavioral Modeling (ABM) feature provided in PSpice. This chapter includes the following sections:

Overview of Analog Behavioral Modeling

The Analog Behavioral Modeling (ABM) feature provided in PSpice allows for 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 that the segment need not be designed component by component.

The symbol library contains several ABM parts that can be classified as either control system parts or as PSpice-equivalent parts. See Basic Controlled Sources on page 6-47 for an introduction to these parts, how to use them, and the distinction between those with general-purpose application and those with special purpose application.

Control system parts are defined with the reference voltage preset to ground so that each controlling input and the output are represented by a single pin in the symbol. These are described in Control System Parts on page 6-7.

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. These are described in PSpice-Equivalent Parts on page 6-28.

For compatibility with Berkeley SPICE, the polynomial transfer function is also supported and is discussed in Basic Controlled Sources on page 6-47. However, the mathematical expressions offered with ABM can usually replace this method.

The Device Equations option (described in the *PSpice Reference Manual*) can also be used for modeling of this type, but we recommend using the ABM feature wherever possible. With Device Equations, the PSpice source code is actually modified. While this is more flexible and the result executes faster, it is much more difficult to use and prone to error. In addition, any changes made to source code must be reapplied whenever a PSpice update is installed. Parts built using ABM can be used for most cases of interest, are much easier to use, and are unaffected by PSpice updates.

The abm.slb Symbol Library File

The symbol file `abm.slb` contains the ABM components. This file can logically be thought of as consisting of two sections.

The first section contains symbols that can be quickly connected to form “control system” types of circuits. These components have names like SUM, GAIN, LAPLACE, and HIPASS.

The second section contains symbols that are useful for more traditional “controlled source” forms of schematic parts. These PSpice-equivalent symbols have names like EVALUE and GFREQ and are based on extensions to traditional PSpice “E” and “G” device types.

ABM components are implemented using PSpice primitives; there is no corresponding `abm.lib` file. A small number of components generate multi-line netlist entries, but the majority are implemented as single PSpice “E” or “G” device declarations. See ABM Part Templates on page 6-6 for a discussion of TEMPLATE attributes and their role in generating netlist declarations. See Implementation of PSpice-Equivalent Parts on page 6-29 for more on PSpice “E” and “G” syntax.

Placing and Specifying ABM Parts

ABM parts are placed and connected in the same way as other part symbols. Once an ABM symbol is placed, the instance attributes can be edited, effectively customizing the operational behavior of the part. This is equivalent to defining an ABM expression describing how inputs are transformed into outputs. The following sections discuss some of the rules for specifying ABM expressions.

Net Names and Device Names in ABM Expressions

In ABM expressions, it is natural to 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.

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.

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.

MicroSim Schematics recognizes these constructs in ABM expressions:

`V(`<*net name*>`)`
`V(`<*net name*>`,`<*net name*>`)`
`I(`<*vdevice*>`)`

When one of these is recognized, Schematics 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 Schematics 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.

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 Schematics will find the local definition. You can override this, and force Schematics 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).

ABM Part Templates

For most ABM symbols, a single PSpice "E" or "G" device declaration is output to the netlist per symbol instance. The TEMPLATE attribute in these cases is straightforward. For example the LOG symbol 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 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 symbols produce more than one primitive PSpice device per symbol instance. In this case, the TEMPLATE attribute may be quite complicated. An example is the DIFFER (differentiator) symbol. 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 netlist, and it creates a local node for the connection of the capacitor and its current-sensing V device.

For clarity, the template is shown on three lines although the actual template is a single line.

```
C^@REFDES %in $$U^@REFDES 1\n
V^@REFDES $$U^@REFDES 0 0v\n
E^@REFDES %out 0 VALUE {@GAIN *
I(V^@REFDES)}
```

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.

Control System Parts

Control system parts have single-pin inputs and outputs. The reference for input and output voltages is analog ground (0). An enhancement to PSpice means that these components can be connected together with no need for dummy load or input resistors.

Table 6-1 lists the control system parts, grouped by function. Also listed are characteristic attributes that may be set. In the sections that follow, each part and its attributes are described in more detail.

Table 6-1 *Control System Parts*

Category	Symbol	Description	Attributes
Basic Components	CONST	constant	VALUE
	SUM	adder	
	MULT	multiplier	
	GAIN	gain block	GAIN
	DIFF	subtracter	
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
Integrator and Differentiator	INTEG	integrator	GAIN, IC
	DIFFER	differentiator	GAIN
Table Look-Ups	TABLE	lookup table	ROW1...ROW5
	FTABLE	frequency lookup table	ROW1...ROW5

Table 6-1 *Control System Parts (continued)*

Category	Symbol	Description	Attributes
Laplace Transform	LAPLACE	Laplace expression	NUM, DENOM
Math Functions (where ‘x’ is the input)	ABS	$\lvert x\rvert$	
	SQRT	$x^{1/2}$	
	PWR	$\lvert x\rvert^{EXP}$	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	$tan^{-1}(x)$	
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

Basic Components

The basic components provide fundamental functions and in many cases, do not require specifying attribute values. These parts are described below.

CONST

VALUE	is a constant value

The CONST part outputs the voltage specified by the VALUE attribute. 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	is a constant gain value

The GAIN part multiplies the input by the constant specified by the GAIN attribute, 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.

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	is a value representing the upper limit
LO	is a value representing the lower limit

The LIMIT part constrains the output voltage to a value between an upper limit (set with the HI attribute) and a lower limit (set with the LO attribute). This part takes one input and provides one output.

GLIMIT

HI	is a value representing the upper limit
LO	is a value representing the lower limit
GAIN	is a 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 attributes. This part takes one input and provides one output.

SOFTLIMIT

HI	is a value representing the upper limit
LO	is a value representing the lower limit
GAIN	is a constant gain value
A, B, V, TANH	are internal variables used to define the limiting function

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.

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 done.

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 *PSpice 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.*

Note *To obtain a listing of the filter Laplace coefficients for each stage, select Setup from the Analysis menu, click on Options, and turn on LIST in the Options dialog box.*

Each of the Chebyshev filter parts is described in the following pages.

LOPASS

FS	is the stop band frequency
FP	is the pass band frequency
RIPPLE	is the pass band ripple in dB
STOP	is the 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.

Figure 6-1 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

We recommend looking at one or more of the references cited in Frequency-Domain Device Models on page 6-35, as well as some of the following references on analog filter design:

1. Ghavsi, M.S. & Laker, K.R., *Modern Filter Design*, Prentice-Hall, 1981.
2. Gregorian, R. & Temes, G., *Analog MOS Integrated Circuits*, Wiley-Interscience, 1986.
3. Johnson, David E., *Introduction to Filter Theory*, Prentice-Hall, 1976.
4. Lindquist, Claude S., *Active Network Design with Signal Filtering Applications*, Steward & Sons, 1977.
5. Stephenson, F.W. (ed), *RC Active Filter Design Handbook*, Wiley, 1985.
6. Van Valkenburg, M.E., *Analog Filter Design*, Holt, Rinehart & Winston, 1982.
7. Williams, A.B., *Electronic Filter Design Handbook*, McGraw-Hill, 1981.

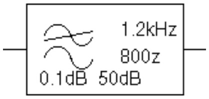


Figure 6-1 *LOPASS Filter*

minimum stop band attenuation is 50 dB. 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 800
1.2K .1dB 50dB
```

HIPASS

FS	is the stop band frequency
FP	is the pass band frequency
RIPPLE	is the pass band ripple in dB
STOP	is the 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.

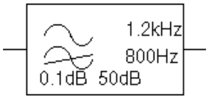


Figure 6-2 *HIPASS Filter Part*

Figure 6-2 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. 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 1.2K
800 .1dB 50dB
```

BANDPASS

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 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.

Figure 6-3 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. 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:

```
EBANDPASS 5 0 CHEBYSHEV {V(10)} = BP 800
1.2K 2K 3K .1dB 50dB
```

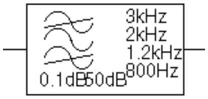


Figure 6-3 *BANDPASS Filter Example Part*

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.

Figure 6-4 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. 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:

```
ENOTCH 5 0 CHEBYSHEV {V(10)} = BR 1.2K 800
3K 2K .1dB 50dB
```

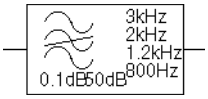


Figure 6-4 *BANDREJ Filter*

Integrator and Differentiator

The integrator and differentiator parts are described below.

INTEG

IC	is the initial condition of the integrator output
GAIN	is the value of the gain

The INTEG part implements a simple integrator. A current source/capacitor implementation is used to provide support for setting the initial condition.

DIFFER

GAIN	is the value of the gain

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 which is used to correlate an input and an output based on a set of data points. These parts are described below and in the following pages.

TABLE

If more than five values are required, the symbol can be customized through the symbol editor. Insert additional row variables into the template using the same form as the first five, and add ROW*n* attributes as needed to the list of attributes.

ROW*n*	is an (input, output) pair; by default, up to five triplets are allowed where *n*=1, 2, 3, 4, or 5

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*	is 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
DELAY	group delay increment; defaults to 0 if left blank
R_I	defines the 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

If more than five values are required, the symbol can be customized through the symbol editor. Insert additional row variables into the template using the same form as the first five, and add ROW*n* attributes as needed to the list of attributes.

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 attribute 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 symbol’s DELAY attribute 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 6-30). 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 or more of the references cited in Frequency-Domain Device Models on page 6-35 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 Figure 6-5 could be used.

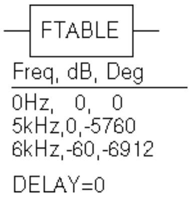


Figure 6-5 *FTABLE Part*

This part is characterized by the following attributes:

```
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)} = (0,0,0) (5kHz,0,-5760)
          (6kHz,-60,-6912)
```

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 attribute. For this example, the group delay is 3.2 msec (6912 / 360 / 6k = 5760 / 360 / 6k = 3.2m). Equivalent attribute 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)} = (0,0,0) (5kHz,0,0) (6kHz,-60,0)
+ DELAY=3.2ms
```

Laplace Transform Part

The LAPLACE part specifies a Laplace transform which is used to determine an output for each input value.

LAPLACE

NUM	is the numerator of the Laplace expression
DENOM	is the 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 attributes for the symbol.

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·ω for s, where ω is 2π·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 1

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·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·ω for s. This gives a flat response out to a corner frequency of 1000/(2π) = 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·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. The LAPLACE part shown in Figure 6-6 could be used for this purpose.

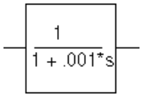


Figure 6-6 *LAPLACE Part*

The transfer function is the Laplace transform (1/[1+.001*s]). This LAPLACE part is characterized by the following attributes:

```
NUM = 1
DENOM = 1 + .001*s
```

The gain and phase characteristics are shown in Figure 6-7.

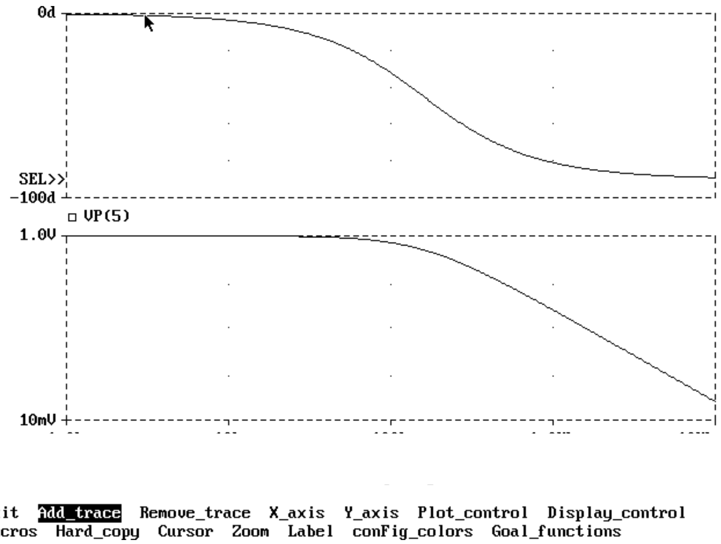


Figure 6-7 *Lossy Integrator Example: Viewing Gain and Phase Characteristics with Probe*

This produces a PSpice netlist declaration like this:

```
ERC    5 0 LAPLACE {V(10)} = {1/(1+.001*s)}
```

Example 2

The input is V(10). The output is a current applied between nets 5 and 0. The Laplace transform describes a lossy transmission

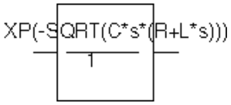


Figure 6-8 *LAPLACE Part*

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 Figure 6-8.

The parameters R, L, and C can be defined in a .PARAM statement contained in a model file. (Refer to the *PSpice Reference Manual* 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.

Math Functions

The ABM math function parts are shown in **Table 6-2**. For each device, the corresponding template is shown, indicating the order in which the inputs are processed, if applicable.

Table 6-2 *ABM Math Function Parts*

Device	**Description**
ABS	Output is the absolute value of the input
SQRT	Output is the square root of the input
PWR	Output is the result of raising the absolute value of the input to the power specified by EXP
PWRS	Output is the result of raising the (signed) input value to the power specified by EXP
LOG	Output is the LOG of the input
LOG10	Output is the LOG_{10} of the input
EXP	Output is the result of e raised to the power specified by the input value (e^{x} where x is the input)
SIN	Output is the *sin* of the input (where the input is in radians)
COS	Output is the *cos* of the input (where the input is in radians)
TAN	Output is the *tan* of the input (where the input is in radians)
ATAN, ARCTAN	Output in radians is the tan^{-1} of the input

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.

ABM Expression Parts

The expression parts are shown in **Table 6-3**. 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* attributes 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 attribute, the second fragment to EXP2, and so on.

Expression attributes can be defined using a combination of arithmetic operators and input designators. You may use any of the standard PSpice arithmetic operators (see **Table 3-5 on page 3-10**) within an expression statement. You may also use the EXP*n* attributes as variables to represent nets or constants.

Table 6-3 *ABM Expression Parts*

Device	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.

Example 1

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 Figure 6-9.

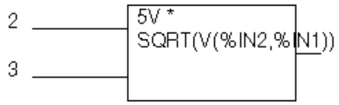


Figure 6-9 *ABM Expression Part Example 1*

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 attribute 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 2

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 internal sweep variable used in transient analyses. 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 Figure 6-10.

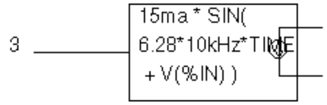


Figure 6-10 *ABM Expression Part Example 2*

This part is characterized by the following attributes:

```
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 3

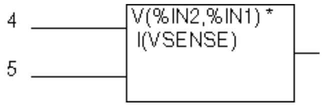


Figure 6-11 *ABM Expression Part Example 3*

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, current output) in Figure 6-11 could represent this.

This part is characterized by the following attributes:

```
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 4

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

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 Figure 6-12.

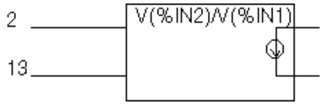


Figure 6-12 *ABM Expression Part Example 4*

This part is characterized by the following attributes:

```
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.*

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 Figure 6-13.

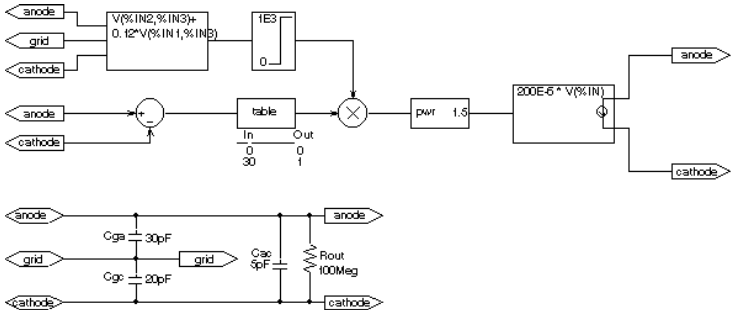


Figure 6-13 *Triode Circuit*

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:

$$\text{ianode} = k_0 * (v_g + k_1 * va)^{1.5}$$

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 attribute EXP1 then represents V(grid, cathode) and the expression attribute 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)

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 attributes 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 attribute 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 attribute 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 in Figure 6-13 shows a possible method for incorporating these effects. To complete the example, one could add a circuit which produces the family of I-V curves (shown in Figure 6-14).

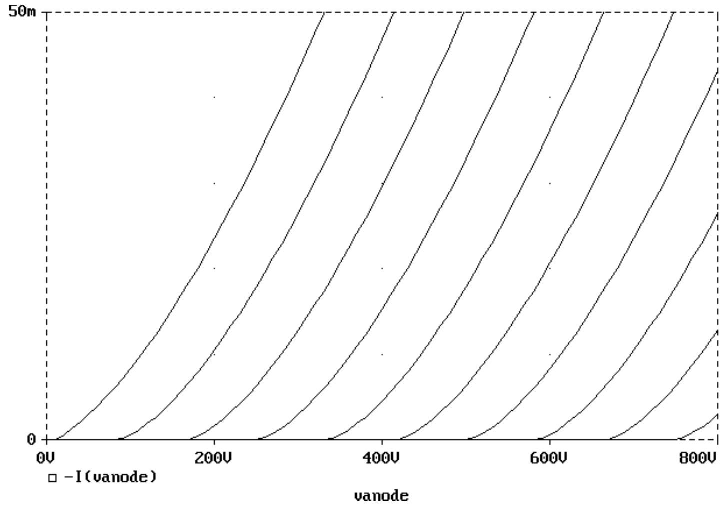


Figure 6-14 *Triode Subcircuit Producing a Family of I-V Curves*

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 symbol. Table 6-4 summarizes the PSpice-equivalent parts available in the symbol library.

Table 6-4 *PSpice-Equivalent Parts*

Category	Symbol	Description	Attributes
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

There are no equivalent "F" or "H" part types in the symbol library since PSpice "F" and "H" devices do not support the ABM extensions.

PSpice-equivalent ABM parts can be classified as either "E" part types or "G" part types. The E part type provides a voltage output, and the G part type provides a current output. The part'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.

Each E or G part type in the `abm.slb` symbol file is defined by a template that provides the specifics of the transfer function.

Other attributes in the model definition can be edited to customize the transfer function. By default, the template cannot be modified directly using Attributes on the Edit menu in Schematics. Rather, the values for other attributes (such as the expressions used in the template) are usually edited, then these values are substituted into the template. However, the symbol editor can be used to modify the template or designate the template as modifiable from within Schematics. This way, custom symbols can be created for special-purpose application.

Implementation of PSpice-Equivalent Parts

Although you generally use Schematics 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 symbols since symbol 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 Reference Manual* for detailed information.

Modeling Mathematical or Instantaneous Relationships

The instantaneous models (using VALUE and TABLE extensions to PSpice "E" and "G" devices in the symbol 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 attribute 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. Available built-in functions are summarized in **Table 3-6 on page 3-11**.

The EVALUE and GVALUE symbols are defined, in part, by the following attributes (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.

Example 1

In the example of an EVALUE device shown in Figure 6-15, the output voltage is set to 5 volts times the square root of the voltage between pins %IN+ and %IN-.

The attribute settings for this device are as follows:

```
EXPR = 5v * SQRT(V(%IN+,%IN-))
```

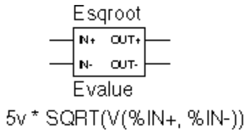


Figure 6-15 *EVALUE Part Example*

Example 2

Consider the device in Figure 6-16. This device could be used as an oscillator for a PSK (Phase Shift Keyed) modulator.

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 attribute settings for this device are shown below:

```
EXPR = 15ma*SIN(6.28*10kHz*TIME+V(%IN+,%IN-))
```

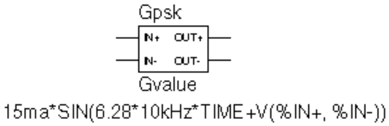


Figure 6-16 *GVALUE Part Example*

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

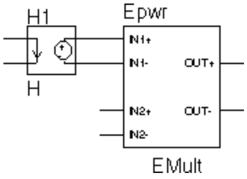


Figure 6-17 *EMULT Part*

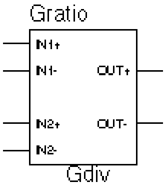
Consider the device in Figure 6-17. This device computes the instantaneous power by multiplying the voltage across pins %IN+ and %IN- by the current through VSENSE. This device's behavior is built-in to the TEMPLATE attribute as follows:

```
TEMPLATE=E^@REFDES %OUT+ %OUT- VALUE {V(%IN1+,%IN1-)
    *V(%IN2+,%IN2-)}
```

You can use the symbol 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.

Example 2

Consider the device in Figure 6-18.



G^@REFDES %OUT+ %OUT- VALUE {V(%IN1+,%IN1-)/V(%IN2+,%IN2-)}

Figure 6-18 *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 symbol can also be used as part of the controlling function.

To create this device, you would first make a new symbol, 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 symbols are defined in part by the following attributes (default values are shown):

ETABLE

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 attribute) 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)
```

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:

1. 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:

2. W. H. Chen, *The Analysis of Linear Systems*, McGraw-Hill (1962)
3. J. A. Aseltine, *Transform Method in Linear System Analysis*, McGraw-Hill (1958)
4. G. R. Cooper and C. D. McGillen, *Methods of Signal and System Analysis*, Holt, Rinehart, and Winston (1967)

Laplace Transforms (LAPLACE)

The ELAPLACE and GLAPLACE parts allow a transfer function to be described by a Laplace transform function. The ELAPLACE and GLAPLACE symbols are defined, in part, by the following attributes (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 6-30). XFORM is an expression in the Laplace variable, s. It follows the rules for standard expressions as described for VALUE expressions with the addition of the s variable.

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 6-35 for more information.

Example

The input to the Laplace transform is the voltage across the input pins, or V(%IN+, %IN-). The EXPR attribute 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 symbol editor, you would define the XFORM and EXPR attributes as follows:

```
XFORM = 1/(1+.001*s)
EXPR = V(%IN+, %IN-)
```

The default template remains:

```
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·ω for s. This gives a flat response out to a corner frequency of 1000/(2π) = 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 6-7 on page 6-19).

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 netlist declaration similar to:

```
ERC 5 0 LAPLACE {V(10)} = {1/(1+.001*s)}
```

Frequency Response Tables (EFREQ and GFREQ)

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.

EFREQ and GFREQ attributes are defined as follows:

EXPR	is the value used for table lookup; defaults to V(%IN+, %IN-) if left blank.
TABLE	is a 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	defines the 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 DELAY attribute 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 symbol’s DELAY attribute 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 6-35 for more information.

Note *The table's frequencies must be in order from lowest to highest.*

Figure 6-19 shows an EFREQ device used as a low pass filter. 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 of .001 (-60 dB) for frequencies above 6 kilohertz. The output is a voltage across the output pins.

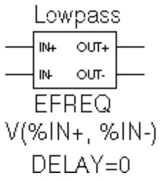


Figure 6-19 *EFREQ Part*

This part is defined by the following attributes:

```
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
```

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 Figure 6-20. 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).

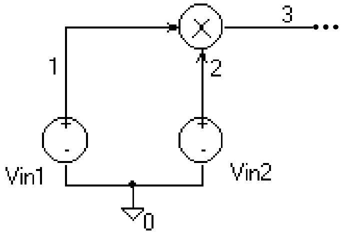


Figure 6-20 *Voltage Multiplier Circuit (Mixer)*

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 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 uses the Transient Analysis Print Step, TSTEP, instead.

Note *TSTOP, TMAX, and TSTEP values are configured using Transient on the Setup menu. The RELTOL attribute is set using Options on the Setup menu.*

PSpice 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/

(2π) = 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 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 does not have a problem calculating the transient (or DC) response. However, you need to be aware that such poles will make the actual device oscillate.

Non-causality and Laplace transforms

PSpice 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 convolution must be applied to both past and future samples of the input in order to properly represent the inverse of the Laplace expression.

A good example of this is the expression {S}, which 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 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 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.

One more point about {S}. You can delay it to keep it causal, but keep in mind that 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 useful 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 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.

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 then checks to see if it can be loosened without inducing sampling errors.

Trading Off Computer Resources For Accuracy

It should be clear from the foregoing discussion that 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, you should first assure yourself based on the relevant criteria described above that 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 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 0.01. This can seriously compromise the accuracy of your simulation.*

Basic Controlled Sources

As with basic SPICE, PSpice has basic controlled sources, including a built-in polynomial capability (POLY keyword) where functions are in the form of a polynomial of any degree and any dimension. POLY is explained in detail under E, F, G, and H device types in the *PSpice Reference Manual*.

The basic controlled source parts available in the symbol library are derived from the standard SPICE E, F, G, and H devices, including their polynomial forms.

The POLY form is no longer necessary since ABM parts provide an easier and more flexible method for specifying expressions.

The EPOLY and GPOLY parts are made obsolete by the Analog Behavioral Modeling extensions provided with PSpice “E” and “G” devices.

This section provides a brief overview of the POLY form, and a comparison of POLY and ABM parts. The standard SPICE implementation of POLY has these limitations:

- many transfer functions are not well represented by polynomials
- the syntax for specifying the polynomial is quite difficult to use
- there is no way to specify frequency-dependent behavior
- you cannot parameterize the coefficients on a POLY statement

The ABM parts eliminate these restrictions.

Table 6-5 summarizes the linear controlled source types provided in the standard symbol library. **Table 6-6** lists their attributes.

Table 6-5 *Basic Controlled Sources in analog.slb*

Part Type	Symbol Name
Controlled Voltage Source (PSpice “E” device)	E EPOLY
Current-Controlled Current Source (PSpice “F” device)	F FPOLY
Controlled Current Source (PSpice “G” device)	G GPOLY
Current-Controlled Voltage Source (PSpice “H” device)	H HPOLY

Basic SPICE Polynomial Expressions (POLY)

PSpice (and SPICE) use the following syntax:

```
<controlled source> <connecting nodes>
+   POLY(<dimension>) <controlling input> <coefficients>
```

where

<controlled source>	is <[E][F][G][H]*device name*>, meaning the device type is one of E, F, G, or H
<connecting nodes>	specifies <(+*node_name*, -*node_name*)> pair between which the device is connected
<dimension>	is the dimension <*value*> of the polynomial describing the controlling function
<controlling input>	specifies <(+*node_name*, -*node_name*)>* pairs used as input to the voltage controlled source (device types E and G), or <*V device name*>* for the current controlled source (device types F and H), and where the number of controlling inputs for either case equals <*dimension*>

<*coefficients*>	specifies the coefficient <*values*>* for the polynomial transfer function

If the source is one-dimensional (there is only one controlling source), “POLY(1)” is required unless the linear form is used. If the source is multidimensional (there is more than one controlling source), the dimension needs to be included in the keyword, for instance “POLY(2).”

Caution must be exercised with the POLY form. For instance,

```
EWRONG 1 0 POLY(1) (1,0) .5 1.0
```

tries to set node 1 to .5 volts greater than node 1. In this case, any analyses which you specify will fail to calculate a result. In particular, PSpice cannot calculate the bias point for a circuit containing EWRONG. This also applies to the VALUE form of EWRONG (EWRONG 1 0 VALUE = {0.5 * V(1)}).

Table 6-6 *Basic Controlled Source Attributes*

Symbol Name	**Attribute**	**Description**
E	GAIN	gain
F		gain
G		transconductance
H		transresistance
EPOLY, FPOLY, GPOLY, HPOLY	COEFF	polynomial coefficient

PSpice has a built-in capability allowing controlled sources to be defined with a polynomial transfer function of any degree and any dimension. Polynomials have associated coefficients for each term. Consider a voltage-controlled source with voltages V_1, V_2, ... V_n. The coefficients are associated with the polynomial according to this convention:

$$
\begin{aligned}
&\mathrm{Vout} = P_0 + \\
&\quad P_1 \cdot V_1 + P_2 \cdot V_2 + \cdots P_n \cdot V_n + \\
&\quad P_{n+1} \cdot V_1 \cdot V_1 + P_{n+2} \cdot V_1 \cdot V_2 + \cdots P_{n+n} \cdot V_1 \cdot V_n + \\
&\quad P_{2n+1} \cdot V_2 \cdot V_2 + P_{2n+2} \cdot V_2 \cdot V_3 + \cdots P_{2n+n-1} \cdot V_2 \cdot V_n + \\
&\qquad\qquad\qquad \vdots \\
&\quad P_{n!/(2(n-2)!)+2n} \cdot V_n \cdot V_n + \\
&\quad P_{n!/(2(n-2)!)+2n+1} \cdot V_1^2 \cdot V_1 + P_{n!/(2(n-2)!)+2n+2} \cdot V_1^2 \cdot V_2 + \cdots \\
&\qquad\qquad\qquad \vdots
\end{aligned}
$$

The above is written for a voltage-controlled voltage source, but the form is similar for the other sources.

The POLY part types shown in **Table 6-5** are defined with a dimension of one, meaning there is only one controlling source. However, similar parts can be defined of any degree and dimension by creating part symbols with appropriate coefficient and TEMPLATE attributes, and the appropriate number of input pins.

The current-controlled parts (F, FPOLY, H, and HPOLY), contain a current-sensing voltage source. When netlisted, they generate two device declarations to the circuit file set: one for the controlled source and one for the independent current-sensing voltage source.

When defining a current-controlled source symbol of higher dimension, the TEMPLATE attribute must account for the same number of current-sensing voltage sources (equal to the dimension value). For example, a two dimensional current-controlled voltage source is described by the following polynomial equation:

$$\text{Vout} = C_0 + C_1I_1 + C_2I_2 + C_{11}I_1^2 + C_{12}I_1I_2 + C_{22}I_2^2$$

To create the two dimensional HPOLY2 symbol, these attributes must be defined:

COEFF0 = 1
COEFF1 = 1
COEFF2 = 1
COEFF11 = 1
COEFF12 = 1
COEFF22 = 1
COEFFS = @COEFF0 @COEFF1 @COEFF2 @COEFF11 @COEFF12 @COEFF22
TEMPLATE = H^@REFDES %5 %6 POLY(2) VH1^@REFDES VH2^@REFDES
\n+ @COEFFS \nVH1^@REFDES %1 %2 0V \nVH2^@REFDES %3 %4 0V

The TEMPLATE definition is actually contained on a single line. The VH1 and VH2 fragments after the \n characters represent the device declarations for the two current-sensing voltage sources required by this part. Also, the symbol graphics must have the appropriate number of pins. When placing an instance of HPOLY2 in your schematic, the COEFF*n* attributes must be appropriately set.

Implementation Examples

Following are some examples of traditional SPICE POLY constructs and equivalent ABM parts which could be used instead.

Example 1: Four-Input Voltage Adder

This is an example of a device which takes four input voltages and sums them to provide a single output voltage.

The representative polynomial expression would be as follows:

$$V_{out} = 0.0 + (1.0)V_1 + (1.0)V_2 + (1.0)V_3 + (1.0)V_4$$

The corresponding SPICE POLY form would be as follows:

```
ESUM 100 101 POLY(4) (1,0) (2,0) (3,0) (4,0) 0.0 1.0 1.0
+  1.0 1.0
```

This could be represented with a single ABM expression device configured with the following expression attributes:

```
EXP1 = V(1,0) +
EXP2 = V(2,0) +
EXP3 = V(3,0) +
EXP4 = V(4,0)
```

Following template substitution for the ABM device, the output becomes:

```
V(OUT) = { V(1,0) + V(2,0) + V(3,0) + V(4,0) }
```

Example 2: Two-Input Voltage Multiplier

This is an example of a device which takes two input voltages and multiplies them together resulting in a single output voltage.

The representative polynomial expression would be as follows:

$$V_{out} = 0.0 + (0.0)V_1 + (0.0)V_2 + (0.0)V_1{}^2 + (1.0)V_1V_2$$

The corresponding SPICE POLY form would be as follows:

```
EMULT 100 101 POLY(2) (1,0) (2,0) 0.0 0.0 0.0 0.0 1.0
```

This could be represented with a single MULT device, which is described in Basic Components on page 6-9. For additional examples of a voltage multiplier device, see Basic Components on page 6-9 and EMULT, GMULT, ESUM, and GSUM on page 6-31.

Example 3: Voltage Squarer

This is an example of a device that outputs the square of the input value.

For the one-dimensional polynomial, the representative polynomial expression reduces to:

$$Vout = P_0 + P_1 \cdot V + P_2 \cdot V^2 + \ldots P_n \cdot V^n$$

The corresponding SPICE POLY form would be as follows:

```
ESQUARE 100 101 POLY(1) (1,0) 0.0 0.0 1.0
```

This could be represented by a single instance of the MULT part, with both inputs from the same net. This results in the following:

$$V_{out} = (V_{in})^2$$

Creating custom ABM parts

When you require a controlled source that is not provided within the special purpose set, or that exhibits more elaborate behavior than is provided with the general purpose parts (with multiple controlling inputs, for example), we recommend building the functionality into a custom symbol similar to the special purpose parts introduced above.

See Creating a New Symbol on page 4-2 for a description of how to create a custom symbol.

The transfer function can be built into the symbol either of the following ways:

- directly in the TEMPLATE definition
- by defining the part's EXPR and related attributes (if any)

Familiarity with the PSpice syntax for declaring "E" and "G" devices will help you form a suitable TEMPLATE definition.

Refer to the *PSpice Reference Manual* for more information about "E" and "G" devices.

Part Three

Setting Up and Running Analyses

Part Three describes how to set up and run analyses and provides setup information specific to each analysis type.

Chapter 7,Setting Up Analyses and Starting Simulation, provides an overview of setting up analyses and starting simulation which applies to any analysis type.

Chapter 8,DC Analyses, describes how to set up DC analyses, including DC sweep, bias point detail, small-signal DC transfer, and DC sensitivity.

Chapter 9,AC Sweep Analysis, describes how to set up AC sweep and noise analyses.

Chapter 10,Transient Analysis, describes how to set up a transient analysis. This chapter includes a variety of topics related to transient analysis, including using the Stimulus Editor.

Chapter 11,Parametric and Temperature Analysis, describes how to set up parametric, temperature, and performance analyses.

Chapter 12,Monte Carlo and Sensitivity/Worst-Case Analyses, describes how to set up Monte Carlo and sensitivity/worst-case analyses.

Setting Up Analyses and Starting Simulation

7

Chapter Overview

This chapter provides an overview of setting up analyses and starting simulation which applies to any analysis type. The other chapters in **Part Three,** *Setting Up and Running Analyses* provide specific analysis setup information for each analysis type.

This chapter includes the following sections:

Analysis Types

Table 7-1 provides a summary of the available PSpice analyses and the corresponding dialog box (accessed using Setup on the Analysis menu) where the analysis parameters are specified.

Table 7-1 *Classes of PSpice Analyses*

Analysis	Analysis Setup Dialog Box	Swept Variable
Standard analyses		
DC sweep	DC sweep	source parameter temperature
bias point	bias point detail	
small-signal DC transfer	transfer function	
DC sensitivity	sensitivity	
frequency response	AC sweep	frequency
noise (requires a frequency response analysis)	AC sweep	frequency
transient response	transient	time
Fourier (requires transient response analysis)	transient	time
Simple multi-run analyses		
parametric	parametric	
temperature	temperature	
Statistical analyses		
Monte Carlo	Monte Carlo/ worst-case	
Sensitivity/worst-case	Monte Carlo/ worst-case	

Note *Parametric analysis is not supported by PSpice Basics.*

Note *Monte Carlo and sensitivity/ worst-case analyses are not supported by PSpice Basics.*

The Probe waveform analyzer is used to display and graphically analyze the results of PSpice simulations for swept analyses. Supplementary analysis information is generated to the simulation output file in the form of lists and tables.

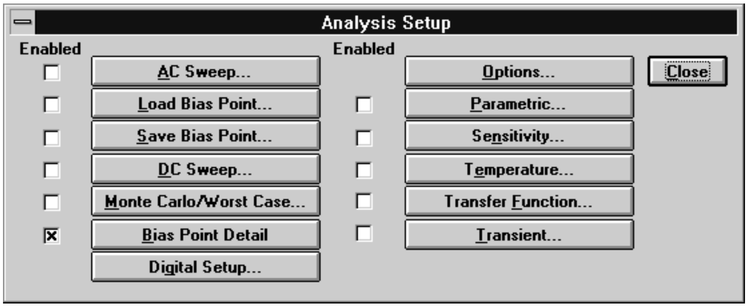
See Part Four, *Viewing Results*, for information about using Probe.

Setting Up Analyses

To set up one or more analyses

1 Select Analysis from the Setup menu in the schematic editor.

Shortcut: Click on ▤.



Specific information for setting up each type of analysis is discussed in the following chapters.

2 At the Analysis Setup dialog box, click on an analysis button.

3 If a setup options dialog box is displayed for the selected analysis type, complete the dialog box as needed.

See Output Variables on page 7-4 for a description of the output variables that can be entered in the setup options dialog box displayed for an analysis type.

4 If needed, click on the check box next to the button for the analysis type so that the analysis is enabled (check box is checked on).

5 Set up any other analyses you want to perform for the circuit.

Execution Order for Standard Analyses

During simulation, any analyses that are enabled are performed in the order shown in Table 7-2. Each type of analysis is done at most once per run.

Several of the analyses (small-signal transfer, DC sensitivity, and frequency response) depend upon the bias point calculation. Since so many analyses use the bias point, PSpice calculates it automatically.

Table 7-2 *Execution Order for Standard Analyses*

1. DC sweep	5. DC sensitivity
2. Bias point	6. Small-signal DC transfer
3. Frequency response	7. Transient response
4. Noise	8. Fourier components

Output Variables

Certain analyses (such as noise, Monte Carlo, sensitivity/worst-case, DC sensitivity, Fourier, and small-signal DC transfer function) require you 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, a dialog box is displayed when you click on the button for the analysis type in the Analysis Setup dialog box.

Voltage

Specify voltage in the following format:

v[*modifiers*](<*out id*>[,<*out id*>]) (1)

where <*out id* > is:

<*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)

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 reference designator, separated by period characters. For example, a resistor with reference designator R34 inside part Y1 placed on a top-level 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/or 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*>])

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 7-3** (which summarizes the accepted output variable formats) and Tables 7-4 through 7-7 (which list valid elements for two-terminal, three or four-terminal, transmission line devices, and AC specifications).

Table 7-3 *PSpice 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 7-4 *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)

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)

Table 7-5 *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)
		G (Gate terminal)
		S (Source terminal)
			V(B11:D)
			ID(B11)

Junction FET	J	D (Drain terminal)
		G (Gate terminal)
		S (Source terminal)
			VG(JFET)
			I(JFET:G)

MOSFET	M	B (Bulk, substrate terminal)
		D (Drain terminal)
		G (Gate terminal)
		S (Source terminal)
			VDG(M1)
			ID(M1)

bipolar transistor	Q	B (Base terminal)
		C (Collector terminal)
		E (Emitter terminal)
		S (Source terminal)
			V(Q1:B)
			I(Q1:C)

IGBT	Z	C (Collector terminal)
		E (Emitter terminal)
		G (Gate terminal)
			V(Z1:C)
			I(Z1:C)

Note *The IGBT device type is not supported by PSpice Basics.*

Table 7-6 *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)
		B (Port B)
			V(T32:A+)
			I(T32:B-)

Table 7-7 *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.

Starting Simulation

Once you have used MicroSim Schematics to enter your circuit design and set up the analyses to be performed, you can start simulation from Schematics by selecting Simulate from the Analysis menu. When you enter and set up your circuit this way, the files needed for simulation are automatically created by Schematics and the simulator is started from Schematics.

There may be situations, however, when you want to run PSpice outside of Schematics. You may want to simulate a circuit that wasn’t created in Schematics, for example, or you may want to run simulations of multiple circuits in batch mode.

This section includes the following:

Starting Simulation from Schematics, below

Starting Simulation Outside of Schematics on page 7-12

Setting Up Batch Simulations on page 7-12

The Simulation Status Window on page 7-14

Starting Simulation from Schematics

Once you have set up the analyses for the circuit, you can start simulation from Schematics any of the following ways:

- Select Simulate from the Analysis menu
- Press F11
- Click on the Simulate icon

Simulate icon:

Starting Simulation Outside of Schematics

To start PSpice outside of Schematics

1. Double-click on the PSpice icon in the MicroSim Program Group.
2. Select Open from the File menu.
3. Do one of the following:
    - Double-click on the circuit file name in the list box.
    - Enter the name of the circuit file to be simulated in the File Name text box.

Setting Up Batch Simulations

Multiple simulations can be run in batch mode when starting PSpice directly with circuit file input. You can use batch mode, for example, to run a number of simulations overnight without user intervention. There are two ways to do this as described below.

Multiple simulation setups within one circuit file

Multiple circuit/simulation descriptions can be concatenated into a single circuit file and simulated with one invocation of PSpice. Each circuit/simulation 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 then processes each one in sequence. The Probe 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 then concatenated all of the outputs.

Running simulations with multiple circuit files

You can direct PSpice to simulate multiple circuit files using one of the following methods.

Method 1

1. Click on the PSpice icon in the MicroSim program group.
2. Select Properties from the File menu.
3. Include the following switch in the command line:

   ```
   /wNO_NOTIFY
   ```

   This turns off the message that pops up each time a simulation is completed.
4. Select Open from the File menu from the PSpice status window.
5. Do one of the following:
   - Type each file name in the File Name text box separated by a space.
   - 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.

Method 2

1. Click on the PSpice icon in the MicroSim program group.
2. Select Properties from the File menu.
3. Include the following switch in the command line:

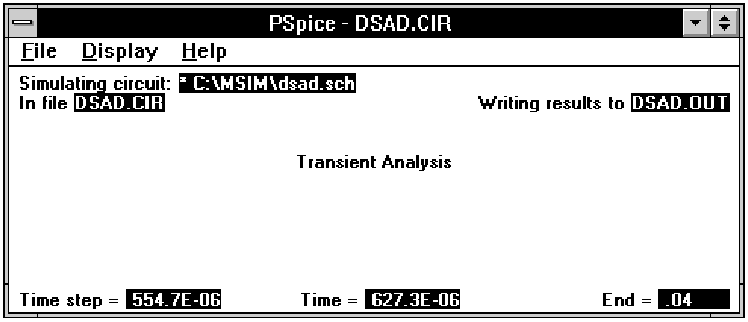
   ```
   /wNO_NOTIFY
   ```

   This turns off the message that pops up each time a simulation is completed.
4. Update the command line in one of the following ways:
   - Include a list of circuit file names separated by spaces.
   - Read a file of run time properties (using the @<*file name*> syntax) which contains a list of circuit file names.

Circuit file names may be fully qualified or contain the wild card characters * and ?.

The Simulation Status Window

As PSpice performs the circuit simulation, a status window is displayed so that you can monitor the progress of the simulation. Figure 7-1 shows an example of the PSpice status window.



Simulating circuit: * C:\MSIM\dsad.sch
In file DSAD.CIR
Writing results to DSAD.OUT

Transient Analysis

Time step = 554.7E-06
Time = 627.3E-06
End = .04

Figure 7-1 *PSpice Status Window*

The status window includes the following:

- **Title bar**
  This area at the top of the window identifies the name of the circuit file currently under simulation, and the name of the simulation output file where audit trail information will be written.
- **Menus**
  The menus accessed from the menu bar include items to control the simulator and customize the window display characteristics. These are especially useful when invoking PSpice directly.
- **Simulation progress display**
  The lower portion of the window displays the progress of each simulation as it proceeds.

DC Analyses

8

Chapter Overview

This chapter describes how to set up DC analyses and includes the following sections:

DC Sweep

Minimum Requirements to Run a DC Sweep Analysis

Minimum circuit design requirements

Table 8-1 *DC Sweep Circuit Design Requirements*

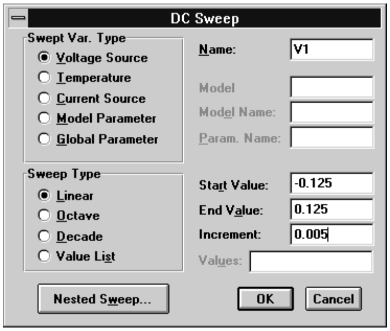
Swept Variable Type	Requirement
voltage source	voltage source with a DC specification (VDC, for example)
temperature	none
current source	current source with a DC specification (IDC, for example)
model parameter	PSpice model (.MODEL)
global parameter	global parameter defined with a parameter block (.PARAM)

Minimum software setup requirements

See Setting Up Analyses on page 7-3 for a description of the Analysis Setup dialog box.

- In the Analysis Setup dialog box, click on the DC Sweep button. Complete the DC Sweep dialog box as needed.
- If needed, click on the DC Sweep check box in the Analysis Setup dialog box so that it is checked on (enabled).
- Start the simulation as described in Starting Simulation on page 7-11.

Note *Do not specify a DC sweep and a parametric analysis for the same variable.*



DC Sweep

Swept Var. Type
● Voltage Source
○ Temperature
○ Current Source
○ Model Parameter
○ Global Parameter

Name: V1
Model:
Model Name:
Param. Name:

Sweep Type
● Linear
○ Octave
○ Decade
○ Value List

Start Value: -0.125
End Value: 0.125
Increment: 0.005
Values:

Figure 8-1 *DC Sweep Setup Example*

Overview of 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 Figure 8-1, 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 lines.

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. To set the DC attribute, select Attributes from the Edit menu.

Shortcut: Click on or double-click on the symbol.

The default DC value of V1 is overridden during the DC sweep analysis and is made to be the swept value. All of the other sources retain their values.

After running the analysis, the simulation output file (`example.out` for the `example.sch` circuit in Figure 8-2) contains a table of voltages relating V1, node OUT1, and node OUT2.

The example circuit `example.sch` is provided with the MicroSim software installation.

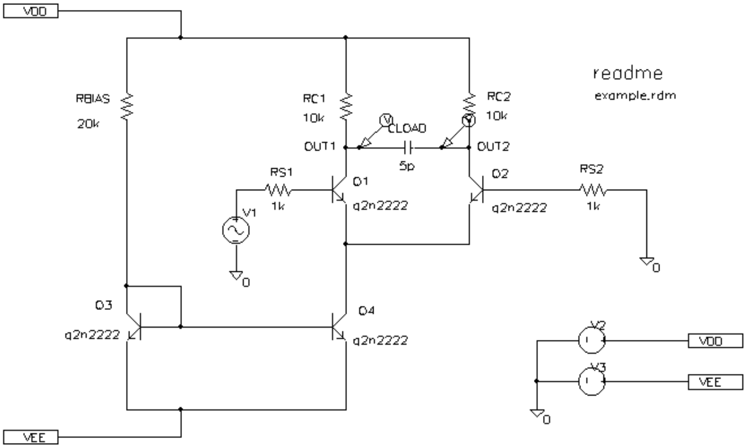
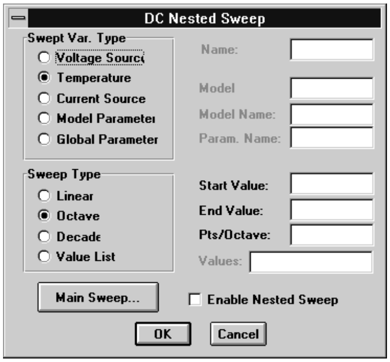


Figure 8-2 *Example Schematic example.sch*

To calculate the DC response of an analog circuit, PSpice 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.

In order to solve the circuit equations, PSpice uses an iterative algorithm using continuous equations.

Nested DC Sweeps



Swept Var. Type
○ Voltage Sourc
● Temperature
○ Current Source
○ Model Paramete
○ Global Parameter

Sweep Type
○ Linear
● Octave
○ Decade
○ Value List

☐ Enable Nested Sweep

A second sweep variable can be selected once a primary sweep value has been specified in the DC Sweep dialog box. When you specify a secondary sweep variable, it forms the outer loop for the 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

1. Click on the Nested Sweep button in the DC Sweep dialog box.
2. Specify a secondary DC sweep in the DC Nested Sweep dialog box.
3. Click on the Enable Nested Sweep check box.
4. Click on Main Sweep to return to the DC Sweep dialog box, or click on OK to return to the Analysis Setup dialog box.

Curve Families for DC Sweeps

Whenever a nested DC sweep is performed, the entire curve family is displayed. That is, the nested DC sweep is treated as a single Probe data section (or you can think of it as a single PSpice run).

For the circuit shown in Figure 8-3, you could set up a DC sweep analysis with an outer sweep of the voltage source VD and an inner sweep of the voltage source VG as listed in **Table 8-2**.

Table 8-2 *Curve Family Example Setup*

	Outer Sweep	Nested Sweep
Swept Var 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

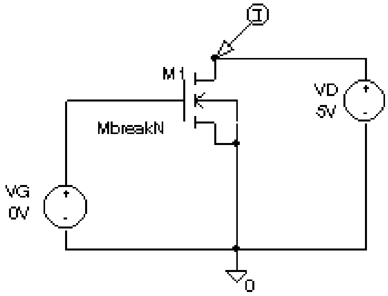


Figure 8-3 *Curve Family Example Schematic*

When the DC sweep analysis is run, add a current marker at the drain pin of M1 and display the simulation result in Probe. The result will look like Figure 8-4.

Use Mark Current Into Pin in MicroSim Schematics on the Markers menu to add a current marker.

We can add the load line for a resistor by adding a trace which 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
```

V_VD is the hierarchical name for VD created by netlisting the schematic. This is the name used by Probe.

This can be useful for determining the bias point for each member of a curve family as shown in Figure 8-5.

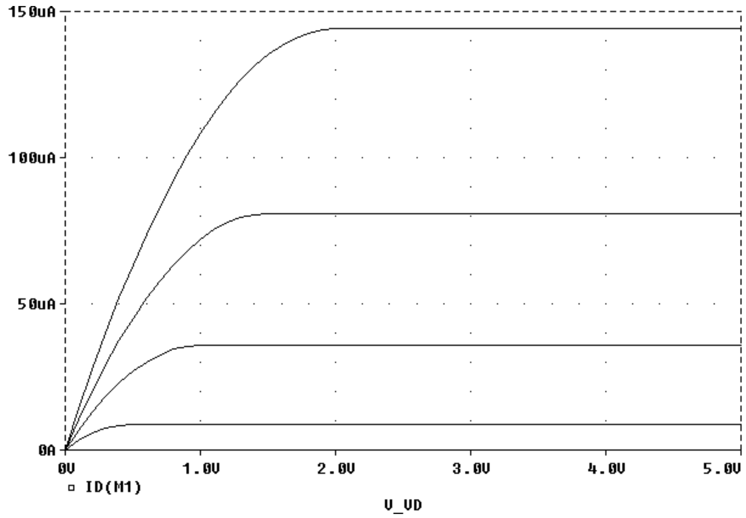


Figure 8-4 *Device Curve Family*

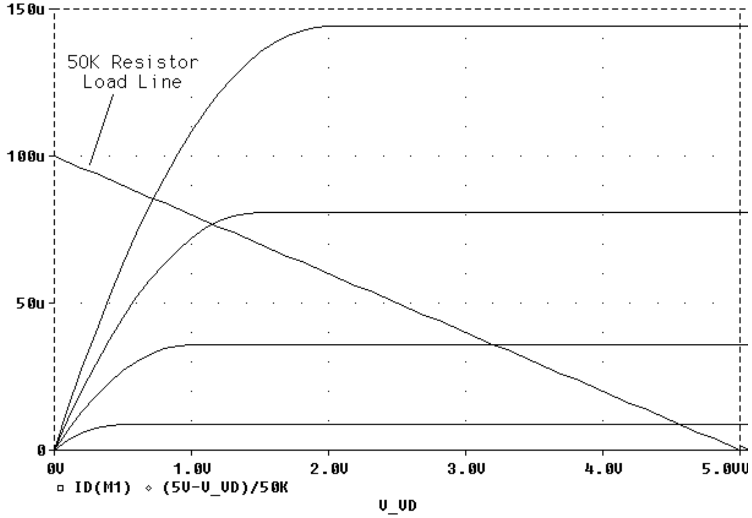


Figure 8-5 *Operating Point Determination for Each Member of the Curve Family*

Bias Point Detail

Minimum Requirements to Run a Bias Point Detail Analysis

Minimum circuit design requirements

None.

Minimum software setup requirements

- Click on the Bias Point Detail check box in the Analysis Setup dialog box so that it is checked on (enabled).
- Start the simulation as described in Starting Simulation on page 7-11.

See Setting Up Analyses on page 7-3 for a description of the Analysis Setup dialog box.

Overview of Bias Point Detail

The bias point is calculated for any analysis whether or not the Bias Point Detail analysis is enabled in the Analysis Setup dialog box. However, additional information is reported when the Bias Point Detail analysis is enabled.

Also see Save and Load Bias Point on page A-2.

When Bias Point Detail analysis is not enabled, only node voltages are reported to the output file.

When the Bias Point Detail analysis is enabled, the following information is reported to the output file:

- A list of all analog node voltages
- The currents of all voltage sources and their total power
- A list of the small-signal parameters for all devices

If Bias Point Detail is enabled, you can suppress the reporting of the bias point analog:

1 Select Setup from the Analysis menu.

2. In the Analysis Setup dialog box, click on Options.
3. In the Yes/No options for the Options dialog box, set NOBIAS to Y (for yes).

Small-Signal DC Transfer

Minimum Requirements to Run a Small-Signal DC Transfer Analysis

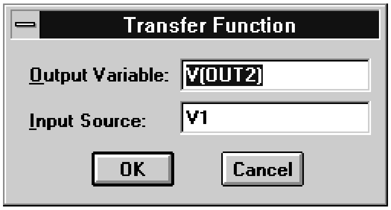
Minimum circuit design requirements

- The circuit should contain an input source, such as VSRC.

Minimum software setup requirements

See Setting Up Analyses on page 7-3 for a description of the Analysis Setup dialog box.

- In the Analysis Setup dialog box, click on the Transfer Function button. In the Transfer Function dialog box, specify the name of the input source desired. See Output Variables on page 7-4 for a description of output variable formats.
- If needed, click on the Transfer Function check box in the Analysis Setup dialog box so that it is checked on (enabled).
- Start the simulation as described in Starting Simulation on page 7-11.



Transfer Function

Output Variable: V(OUT2)

Input Source: V1

Overview of Small-Signal DC Transfer

The small-signal DC transfer analysis causes the small-signal transfer function to be calculated by linearizing the circuit around the bias point. The small-signal gain, input resistance, and output resistance are calculated and reported.

To calculate the small-signal gain, input resistance, and output resistance, you need to specify an output voltage or current through a voltage source in the Transfer Function dialog 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.

You also need to specify the input source name in the Transfer Function dialog box. The gain from the input source to the output variable is output 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 the gain from V1 to V(OUT2) is calculated. All calculations are reported to the simulation output file.

DC Sensitivity

Minimum Requirements to Run a DC Sensitivity Analysis

Minimum circuit design requirements

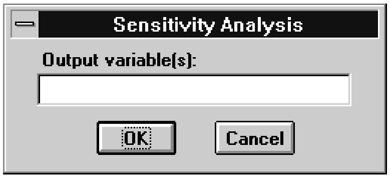
None.

See Setting Up Analyses on page 7-3 for a description of the Analysis Setup dialog box.

See Output Variables on page 7-4 for a description of output variables.

Minimum software setup requirements

- In the Analysis Setup dialog box, click on the Sensitivity button. In the Sensitivity Analysis dialog box, enter the output variable desired.
- If needed, click on the Sensitivity check box in the Analysis Setup dialog box so that it is checked on (enabled).
- Start the simulation as described in Starting Simulation on page 7-11.



Overview of 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.

AC Sweep Analysis

9

Chapter Overview

This chapter describes how to set up an AC sweep analysis and includes the following sections:

Overview of AC Sweep Analysis

Minimum Requirements to Run an AC Sweep Analysis

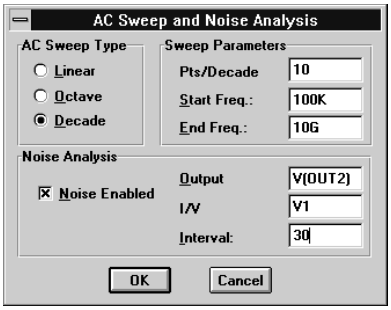
Minimum circuit design requirements

- An AC source with an AC specification, such as VAC or IAC.

See Setting Up Analyses on page 7-3 for a description of the Analysis Setup dialog box.

Minimum software setup requirements

- In the Analysis Setup dialog box, click on the AC Sweep button. Complete the AC Sweep and Noise Analysis dialog box as needed.
- If needed, click on the AC Sweep check box in the Analysis Setup dialog box so that it is checked on (enabled).
- Start the simulation as described in Starting Simulation on page 7-11.


AC Sweep and Noise Analysis

AC Sweep Type
○ Linear
○ Octave
● Decade

Sweep Parameters
Pts/Decade 10
Start Freq.: 100K
End Freq.: 10G

Noise Analysis
☒ Noise Enabled
Output V(OUT2)
I/V V1
Interval: 30

Figure 9-1 *AC Sweep Setup Example*

Overview of AC Sweep

The frequency response analysis calculates the small-signal response of the circuit, linearized around the bias point, to a combination of inputs.

Nonlinear devices, such as voltage-controlled or current-controlled switches, behave in a similar manner. That is, they are linearized about their bias point value and then a linear (small-signal) analysis is performed. In the case of the nonlinear controlled switches, the switch acts as a resistor based upon the controlling input value at the bias point.

For the AC sweep analysis specified in Figure 9-1, the frequency is swept from 100 kHz to 10 GHz by decades, with 10 points per decade. Unlike DC sweep, the AC sweep analysis does not specify an input source. Instead, each independent source contains its own AC specification for magnitude and phase.

In the example circuit in Figure 9-2, V1 has the AC attribute equal to 1 volt and a relative phase of 0 degrees by default. During the AC sweep analysis, the contributions from all sources are propagated throughout the circuit and summed at all the nodes. V1 is the only input to an amplifier, so it is the only source to have a nonzero AC value.

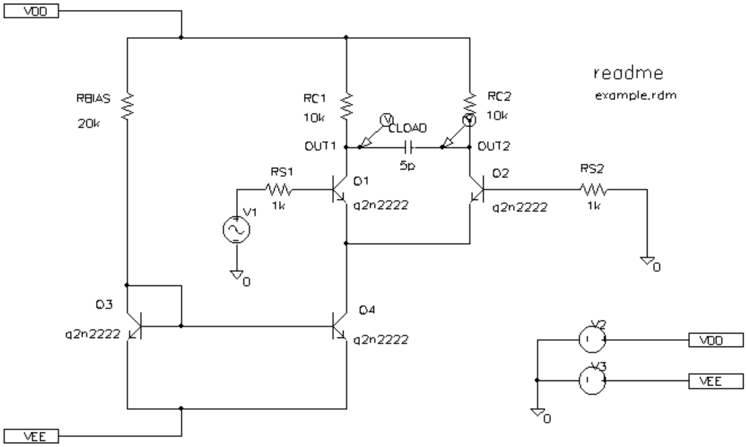


Figure 9-2 *Example Schematic example.sch*

The example circuit `example.sch` is provided with the MicroSim software installation.

Nonlinear Components in an AC Sweep Analysis

An AC Sweep analysis is a linear or small-signal analysis. In order to perform a traditional small-signal analysis of a transistor amplifier, for example, you would go through the following steps:

1. Compute the DC bias point for the circuit.
2. Compute the transconductances at this bias point.
3. Do linear circuit analysis.

You would generally use a number of simplifying approximations. For example, a bipolar transistor in common-emitter mode would be replaced by a constant transconductance (collector current proportional to base-emitter voltage), plus a number of constant impedances.

PSpice performs exactly the same sequence of operations. It determines partial derivatives for nonlinear devices at the bias point, and these are then used for small-signal analysis.

Other nonlinear devices are treated in exactly the same way. For example, suppose we have a behavioral modeling block that multiplies V(1) by V(2). Multiplication is a nonlinear operation, so AC sweep analysis requires that this block be replaced by its linear equivalent. Unless the bias point is known, however, the linear equivalent cannot be determined.

Suppose a 2 volt DC source is connected between node 1 and ground (with suitable dummy resistors). Then, at the bias point for the circuit, V(1) = 0 volts and V(2) = 2 volts. You can now replace the multiplier block with an equivalent block that has a fixed gain of 2 so that the output is 2*V(1) + 0*V(2). AC sweep analysis correctly shows the transfer function of the circuit as having a gain of 2.

Consider another example. Suppose the 2 volt DC source in the above example is replaced with an AC source with amplitude 1. When the bias point is computed, there are no DC sources in the circuit, so all nodes are at 0 volts at the bias point. Now 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, just a multiplier). Of course, a double-balanced mixer produces outputs at DC and at twice the input frequency, but these terms cannot be seen with a linear, small-signal analysis.

Noise Analysis

Noise analysis is an AC sweep option that is run when the Noise Analysis check box is checked on in the AC Sweep and Noise Analysis dialog box. Noise analysis calculates noise contributions from each resistor and semiconductor device and performs an RMS sum at a specified output node. This calculation is done for all the frequencies specified for the AC sweep analysis.

Output voltage requires that you enter an output variable. I/V Source specifies the name of an independent voltage or current source at which to calculate the equivalent input noise. For example, if you enter `V1` as the I/V Source, V1 is the place at which an equivalent input noise is calculated.

Output variables are described in Output Variables on page 7-4.

Interval specifies the print interval. For every *n*th frequency where *n* is the print interval, a detailed table is printed showing the individual contributions of all the circuit's noise generators to the total noise. If no value is specified, then no detailed table is printed.

For example, assume that you want to run a noise analysis on the circuit shown in Figure 9-2 on page 9-3. You would enable noise analysis by checking the Noise Analysis check box, then you might enter the following values in the AC Sweep and Noise Analysis dialog box:

Output Voltage	`V(OUT2)`
I/V Source	`V1`
Interval	`30`

This sets node OUT2 as the output node and V1 as the input (or source). This does *not* mean that V1 is the noise source. Rather, V1 is the place at which an equivalent input noise is calculated.

The noises are summed at node OUT2. This result is then divided by the gain from V1 to node OUT2 to get the amount of noise which, if injected at V1 into a noiseless circuit, would cause the previously calculated amount of noise at node OUT2. The interval value of 30 results in a detailed table of values for every 30th frequency.

There are two types of reports from the noise analysis: detailed tables and summary tables. The detailed tables are specified by the interval value in the Noise Analysis section of the dialog box. In the preceding example, the value of 30 specifies that for every 30th frequency of the AC analysis, a detailed table is to be reported. These tables are labeled NOISE ANALYSIS in the simulation output file.

The second type of output reported are summary outputs showing the RMS summed noise at node OUT2(ONOISE) and the equivalent input noise at V1(INOISE).

Transient Analysis

10

Chapter Overview

This chapter describes how to set up a transient analysis and includes the following sections:

Overview of Transient Analysis

Minimum Requirements to Run a Transient Analysis

Minimum circuit design requirements

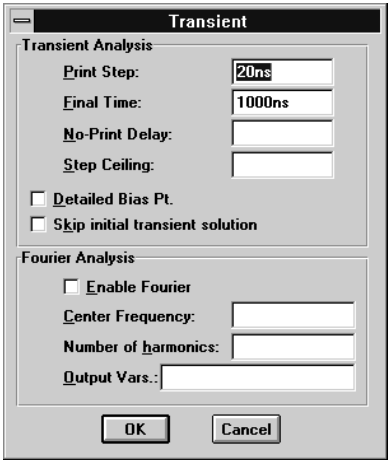
Circuit should contain one of the following:

- An independent source with a transient specification (see **Table 10-1**)
- An initial condition on a reactive element
- A controlled source that is a function of time

See Setting Up Analyses on page 7-3 for a description of the Analysis Setup dialog box.

Minimum software setup requirements

- In the Analysis Setup dialog box, click on the Transient button. Complete the Transient dialog box as needed.
- If needed, click on the Transient check box in the Analysis Setup dialog box so that it is checked on (enabled).
- Start the simulation as described in Starting Simulation on page 7-11.



Print Step: 20ns
Final Time: 1000ns
☐ Detailed Bias Pt.
☐ Skip initial transient solution
☐ Enable Fourier

Stimulus Generation

Overview of Stimulus Generation

Parts which generate stimuli for your circuit can be divided into two categories:

- stimulus parts whose transient behavior is characterized graphically using the Stimulus Editor
- stimulus parts whose transient (if applicable) behavior is characterized by manually defining their attributes within MicroSim Schematics

Their symbols are summarized in **Table 10-1**.

See The Stimulus Editor Utility on page 10-5 for a description of how to use the Stimulus Editor.

Note *PSpice Basics does not include the Stimulus Editor.*

Table 10-1 *Stimulus Parts Classified by Transient Specification*

Category	Symbol Name	Description
graphically characterized	VSTIM	voltage source
	ISTIM	current source
characterized by attribute	VSRC VEXP VPULSE VPWL VPWL_ENH VPWL_FILE VSFFM VSIN	voltage sources
	ISRC IEXP IPULSE IPWL IPWL_ENH IPWL_FILE ISFFM ISIN	current sources

All of these parts are used by placing instances in your schematic and defining their attributes using Attributes on the Edit menu. Each attribute-characterized stimulus has a distinct set of attributes depending upon the kind of transient behavior it represents. For VPWL_FILE, IPWL_FILE, and FSTIM, a separate file contains the stimulus specification.

As an alternative, the Stimulus Editor utility automates the process of defining the transient behavior of stimulus devices. The Stimulus Editor allows you to create analog stimuli which generate sine wave, repeating pulse, exponential pulse, single-frequency FM, and piecewise linear waveforms. This applies to both part instances placed in your schematic as well as new stimulus symbols that you might create.

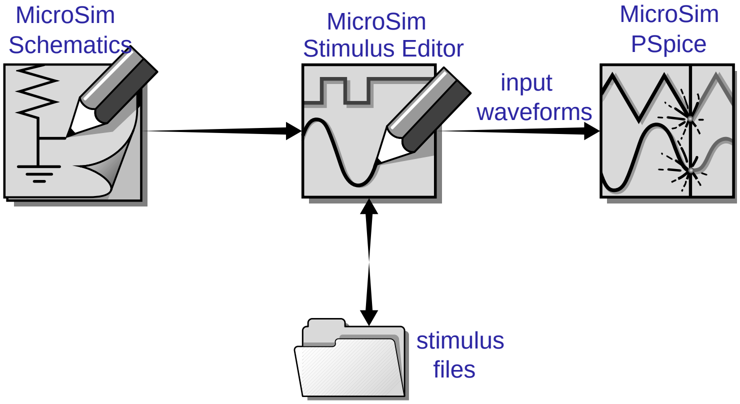


Figure 10-1 *Relationship of Stimulus Editor with Schematics and PSpice*

The stimulus specification created using the Stimulus Editor is saved to a file, automatically configured into the schematic, and associated with the corresponding VSTIM, ISTIM, or DIGSTIM part instance or symbol definition.

The Stimulus Editor Utility

not
included
in: **Basics+**

The Stimulus Editor is a utility which allows you to quickly set up and verify the input waveforms for a transient analysis. You can create and edit voltage sources and current sources for your circuit. Graphical feedback allows you to verify the waveform quickly.

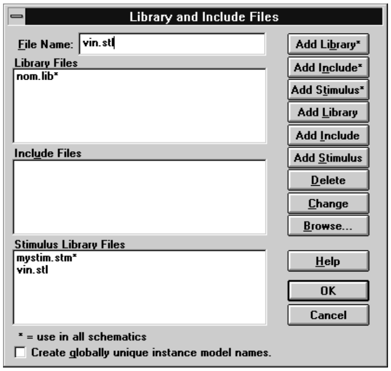
Stimulus Files

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) and I (current source) forms. Since the Stimulus Editor produces these statements automatically, you will never have to be concerned with their syntax. However, if you are interested in a detailed description of their syntax, see the descriptions of V and I devices in the *PSpice Reference Manual*.

MicroSim software versions without the Stimulus Editor must use the characterized-by-attribute sources listed in **Table 10-1 on page 10-3**.

Configuring Stimulus Files

In the schematic editor, Library and Include Files on the Analysis menu allows you to view the list of stimulus files pertaining to your current schematic, or to manually add, delete, or change the stimulus file configuration. The Stimulus Library Files 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 Schematics environment or local to the current schematic. Global files are marked with an asterisk (*) after the file name.



File Name: vin.stl

Library Files
nom.lib*

Include Files

Stimulus Library Files
mystim.stm*
vin.stl

* = use in all schematics

☐ Create globally unique instance model names.

When starting the Stimulus Editor from Schematics, stimulus files are automatically configured (added to the list) as local to the current schematic. Otherwise, new stimulus files can be added to the list by entering the file name in the File Name text box and then clicking on the Add Stimulus (local configuration) or Add Stimulus* (global configuration) button. All other commands work as described for model and include files in Global and local model files on page 5-36.

Starting the Stimulus Editor

The Stimulus Editor is fully integrated with Schematics and can be run from either the schematic editor or symbol editor.

You can start the Stimulus Editor by the following methods:

- Double-click on a stimulus instance.
- Select one or more stimulus instances in the schematic and select Stimulus from the Edit menu.
- Select Edit Stimulus from the Analysis menu.

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).

Defining Stimuli

1. Place stimulus part instances from the symbols VSTIM and ISTIM.
2. Start the Stimulus Editor as described in Starting the Stimulus Editor on page 10-6.
3. From within the Stimulus Editor:
    a. Edit the transient specification:

- As indicated by the dialog boxes and prompts, or by:
- Direct manipulation of the input waveform display for analog piecewise linear stimuli.

b Save the edits by selecting Save from the File menu.

Example: piecewise linear stimulus

1. Open an existing schematic or start a new one.
2. Select Get New Part from the Draw menu and either browse the `source.slb` Symbol Library file for VSTIM (and select it), or type `VSTIM` in the Part text box.
3. Place the symbol. It looks like a regular voltage source with a STIMULUS attribute displayed.
4. Double-click on the STIMULUS label and type `Vfirst`. This names the stimulus that we are going to create.
5. If you are working in a new schematic, use Save As from the File menu to save it. This is necessary since the schematic name is used to create the default stimulus file name.
6. Double-click on the VSTIM symbol. This starts the Stimulus Editor and displays the New Stimulus dialog box. You can see that the stimulus already has the name of Vfirst.
7. Select PWL in the dialog box and click on 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.
8. 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.
9. Click-right to stop adding points.
10. Select Save from the File menu.

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 on its handle and press Del.

To add additional points, either select Add Point from the Edit menu, press Alt + A , or click on the Add Point toolbar button.

At this point you can return to Schematics, edit the current stimulus, or go on to create another.

This example creates a 10 K sin wave with the amplitude parameterized so that it can be swept during a simulation.

Example: sine wave sweep

1. Open an existing schematic or start a new one.
2. Place a VSTIM symbol on your schematic.
3. To name the stimulus, double-click on the STIMULUS attribute and type `Vsin`.
4. Double-click on the VSTIM symbol to start the Stimulus Editor.
5. Define the stimulus parameter for amplitude:
   - **a** Select Cancel while in the New Stimulus dialog.
   - **b** Select Parameters from the Tools menu.
   - **c** Enter `AMP=1` in the Definition text box, and click OK.
   - **d** Select New from the Stimulus menu or click on 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.
6. Define the other stimulus properties:
   - **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 on OK.
   - **d** Select Save from the File menu.
7. Within Schematics, place and define the PARAM symbol:
   - **a** Select Get New Part from the Draw menu.

- **b** Either browse `special.slb` for the PARAM symbol or type in the name.
- **c** Place the symbol on your schematic and double-click on it to edit the attributes.
- **d** Set the value of the NAME1 attribute to AMP (no curly braces)
- **e** Set the value of the VALUE1 attribute to 1.

8 Set up the parametric sweep and other analyses:

- **a** Select Setup from the Analysis menu, and click on the Parametric button.
- **b** Select Global Parameter in the Swept Var. Type frame.
- **c** Select Linear in the Sweep type frame.
- **d** Enter `AMP` in the Name text box.
- **e** Specify values for the Start Value, End Value, and Increment text boxes.
- **f** You can now set up your usual Transient, AC, or DC analysis and run the simulation.

Creating New Stimulus Symbols

1 Use the symbol editor to edit or create a symbol with the following attributes:

STIMTYPE	Type of stimulus; valid values is ANALOG. If this attribute is nonexistent, the stimulus is assumed to be ANALOG.
STIMULUS	Name of the stimulus model.

2 Select Stimulus from the Edit menu. Schematics searches the configured list of global stimulus files. If you are creating a new stimulus symbol and the stimulus is not found, you are prompted for the name of the stimulus file in which the new definition should be saved.

3 From within the Stimulus Editor, edit the transient specification as indicated by the dialogs and prompts, or by direct manipulation of the input waveform display for analog piecewise linear stimuli.

4 Select Save from the File menu.

Editing a Stimulus

To edit an existing stimulus

1 Start the Stimulus Editor and select Get from the Stimulus menu.

2 Double-click on the trace name. 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.

PWL stimuli are a little different since they are a series of time/value pairs.

To edit a PWL stimulus

1 Double click on the trace name. This displays the handles for each defined data point.

2 Click on any handle to select it. To reshape the trace, drag it to a new location. To delete the data point, press Del.

3 To add additional data points, either select Add from the Edit menu or click on the Add Point icon.

4 Right-click to end adding new points.

This provides a fast way to scale a PWL stimulus.

To select a time and value scale factor for PWL stimuli

1 Select the PWL trace by clicking on its name.

2 Select Attributes from the Edit menu or click on the corresponding toolbar button.

Deleting and Removing Traces

To delete a trace from the displayed screen, select the trace name by clicking on its name, then press Del. 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. **Once a trace is removed, it is no longer retrievable. Delete traces with caution.**

Manual Stimulus Configuration

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. The procedure is described here.

To manually configure a stimulus

1. Start the Stimulus Editor by double-clicking on the Stimulus Editor icon in the MicroSim program group.
2. Open a stimulus file using Open on 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. Select Save from the File menu.

4. In the schematic editor, configure the Stimulus Editor's output file into your schematic:
   - a. Select Library and Include Files from the Analysis menu.
   - b. Enter the file name specified in step 2.
   - c. If the stimulus specifications are for local use in the current schematic, click on the Add Stimulus (or Add Include) button. For global use by a symbol in the Symbol Library or by any schematic, use Add Stimulus* (or Add Include*) instead.
   - d. Click on 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:
   - a. Place a stimulus part in your schematic from the symbol set: VSTIM, ISTIM, and DIGSTIM.
   - b. Click on the VSTIM, ISTIM, or DIGSTIM instance.
   - c. Select Attributes from the Edit menu.
   - d. Click on the STIMULUS= attribute, type in the name of the stimulus, and click on Save Attr.
   - e. Complete specification of any VSTIM or ISTIM instances by selecting Attributes from the Edit menu and editing their DC and AC attributes.

     Click on the DC= attribute, type its value in the Value text box, and click on Save Attr.

     Click on the AC= attribute, type its value in the Value text box, and click on Save Attr.

     Click on OK to return to the schematic.
7. To change stimulus references globally for a symbol:
   - a. Select Edit Library from the File menu to start the symbol editor.

b Create or change a symbol definition, making sure to define the following attributes:

PART	Symbol name. It is good practice to have the symbol name match the STIMULUS name.
STIMULUS	Stimulus name as defined in the Stimulus Editor.

See **Chapter 4,**Creating Symbols, for a description of how to create and edit symbols.

Using the VSTIM and ISTIM Sources

VSTIM and ISTIM source symbols are used to define voltage and current sources, respectively, where the transient specification is graphically defined using the Stimulus Editor.

To use the VSTIM or ISTIM source to define the transient stimulus behavior

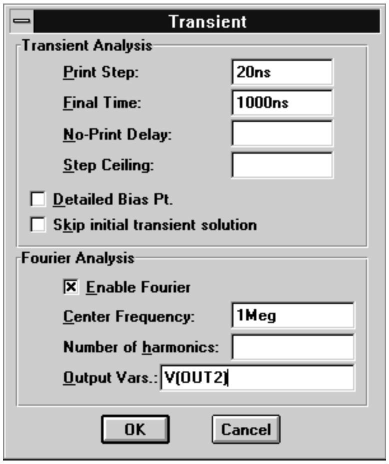
1. Place the VSTIM or ISTIM source symbol in your schematic.
2. Double-click on the source instance to start the Stimulus Editor. When you are asked whether you want to edit the named stimulus, click on OK.
3. Fill in the transient specification according to the dialogs and prompts. Piecewise linear stimuli can be specified by direct manipulation of the input waveform display.

See Stimulus Generation on page 10-3 for a description of the Stimulus Editor.

not included in: Basics+

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.sch` in Figure 10-2. (`Example.sch` is shown in Figure 10-3.) 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.



Transient

Transient Analysis

Print Step: 20ns

Final Time: 1000ns

No-Print Delay:

Step Ceiling:

☐ Detailed Bias Pt.

☐ Skip initial transient solution

Fourier Analysis

☒ Enable Fourier

Center Frequency: 1Meg

Number of harmonics:

Output Vars.: V(OUT2)

Figure 10-2 *Transient Analysis Setup for example.sch*

During a transient analysis, any or all of the independent sources may have time-varying values. In `example.sch`, the only source which has a time-varying value is V1 (VSIN part) with attributes:

```
VOFF = 0v
VAMPL = 0.1v
FREQ = 5Meg
```

V1's value varies as a 5 MHz sine wave with an offset voltage of 0 volts and a peak amplitude of 0.1 volts.

The example circuit `example.sch` is provided with the MicroSim software installation.

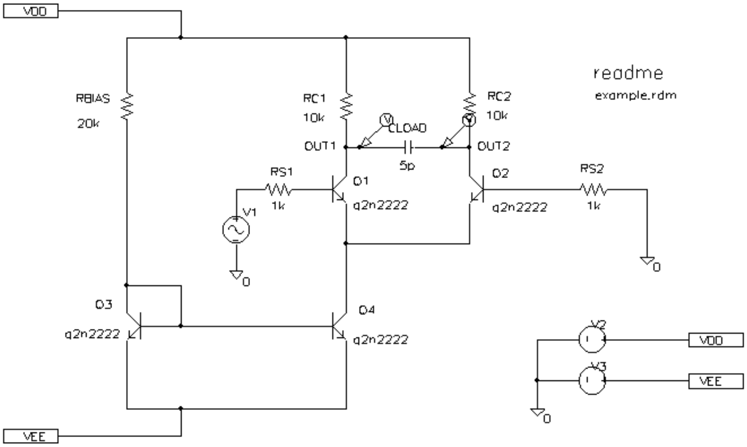


Figure 10-3 *Example Schematic example.sch*

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. If you want to report the small-signal parameters for the transient bias point, you should use the

Transient command and enable Detailed Bias Point. Otherwise, if all you want is the result of the transient run itself, you should only enable the Transient command.

In the simulation output file `example.out`, the bias-point report for the transient bias point is labeled INITIAL TRANSIENT SOLUTION.

Internal Time Steps in Transient Analyses

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 so in the Step Ceiling text box in the Transient dialog. 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 has been requested to be reported. The values at the print time steps are obtained by 2nd-order polynomial interpolation from values at the internal steps.

Switching Circuits in Transient Analyses

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.

This technique is described in:

V. Bello, "Computer Program Adds SPICE to Switching-Regulator Analysis," *Electronic Design*, March 5, 1981.

Plotting Hysteresis Curves

Transient analysis can be used to look at a circuit's hysteresis. Consider, for instance, the circuit shown in Figure 10-4 (netlist in Figure 10-5).

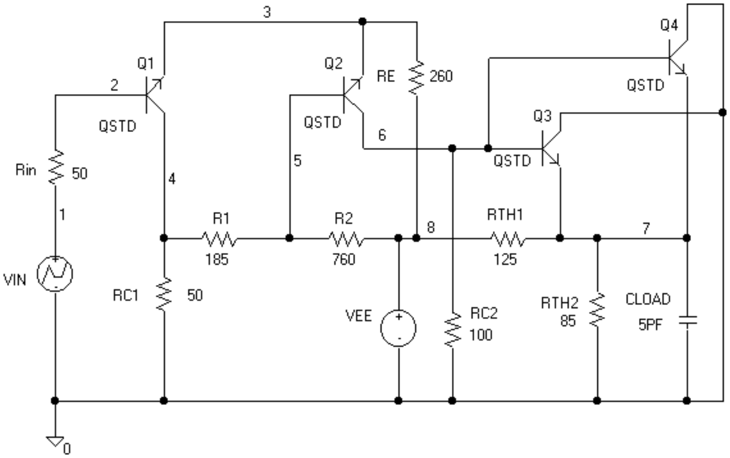


Figure 10-4 *ECL Compatible Schmitt Trigger*

```
* Schematics 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 10-5 *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, we 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 two advantages:

- it avoids convergence problems
- it covers both the upward and downward transitions in one analysis

After the simulation, when we are in Probe, the X axis variable is initially set to be Time. By selecting X Axis Settings from the Plot menu and clicking on the Axis Variable button, we can set the X axis variable to be V(1). Then we can 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 (X Axis Settings on the Plot menu). This plots the output of the Schmitt trigger against its input, which is what we want. The result looks similar to Figure 10-6.

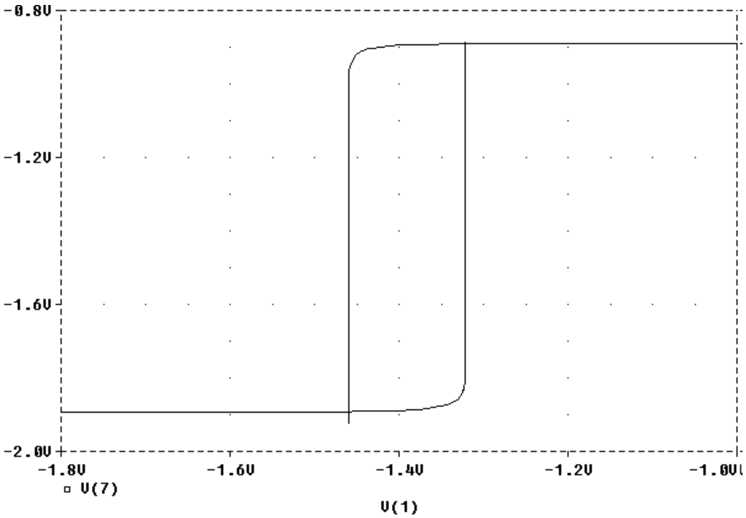


Figure 10-6 *Hysteresis Curve Example: Schmitt Trigger*

Fourier Components

The Fourier analysis is enabled through the transient analysis setup dialog box. Fourier analysis calculates the DC and Fourier components of the result of a transient analysis. By default, the 1st through 9th components are computed, however, more can be specified.

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 Probe, a Fast Fourier Transform (FFT) of the complete waveform can be calculated and its spectrum displayed.

In the example Fourier analysis specification shown in Figure 10-2 on page 10-14, the voltage waveform at node OUT2 from the transient analysis is to be used and the fundamental frequency is to be 1 megahertz for the harmonic decomposition.

The period of fundamental frequency is 1 microsecond (inverse of the fundamental frequency). Only the last 1 microsecond of the transient analysis is used, and that portion is assumed to repeat indefinitely. Since V1’s sine wave does indeed repeat every 1 microsecond, this is sufficient. In general, however, you must make sure that the fundamental Fourier period fits the waveform in the transient analysis.

Parametric and Temperature Analysis

11

Chapter Overview

This chapter describes how to set up parametric and temperature analyses. Parametric and temperature are both simple multi-run analysis types.

This chapter includes the following sections:

not included in: **Basics +**

Parametric Analysis

Minimum Requirements to Run a Parametric Analysis

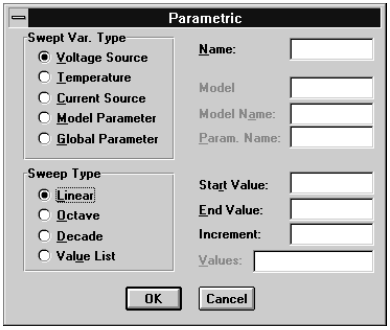
Minimum circuit design requirements

- Set up the circuit according to the swept variable type as listed in **Table 11-1**.
- Set up a DC sweep, AC sweep, or transient analysis.

Table 11-1 *Parametric Analysis Circuit Design Requirements*

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 model
global parameter	global parameter defined with a parameter block (PARAM)

See Setting Up Analyses on page 7-3 for a description of the Analysis Setup dialog box.



Minimum software setup requirements

- In the Analysis Setup dialog box, click on the Parametric button. Complete the Parametric dialog box as needed.
- If needed, click on the Parametric check box in the Analysis Setup dialog box so that it is checked on (enabled).
- Start the simulation as described in Starting Simulation on page 7-11.

Note *Do not specify a DC sweep and a parametric analysis for the same variable.*

Overview of 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.

See Parametric Analysis on page 2-22 for a description of how to set up a parametric analysis.

Example: RLC Filter

This example shows how to perform a parametric sweep and how to analyze the results with performance analysis.

With performance analysis, values can be derived from a series of simulator runs and plotted versus a parameter that varies between those runs. For this example, the derived values we wish to plot are the overshoot and the rise time versus the damping resistance of the filter. Deriving these values by hand is quite involved and letting the simulator do the work for us is an appealing alternative.

Entering the schematic

The schematic for the RLC filter (`rlcfilt.sch`) is shown in Figure 11-1.

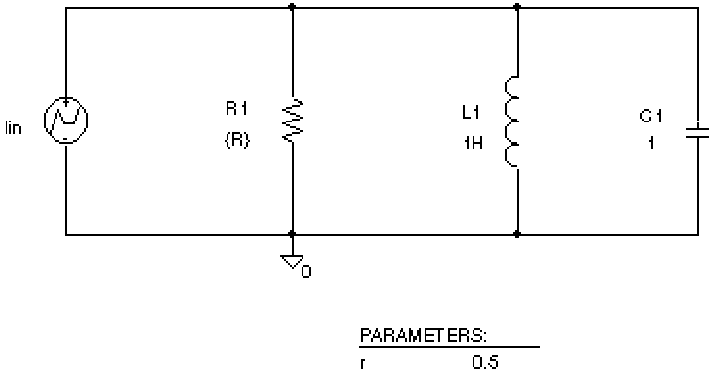


Figure 11-1 *Passive Filter Schematic*

This series of PSpice 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, we perform a transient analysis of approximately 20 seconds.

Create the circuit in MicroSim Schematics by placing a piecewise linear independent current source (IPWL from `source.slb`). Set the current source attributes as follows:

```
AC = 1a
T1 = 0s
I1 = 0a
T2 = 10ms
I2 = 0a
T3 = 10.1ms
I3 = 1a
```

Place an instance of a resistor and set its VALUE attribute to the expression, {R}. To define R as a global parameter, place a PARAM pseudocomponent defining its NAME1 attribute to R and VALUE1 attribute 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 (AGND from `port.slb`). Wire the schematic symbols together as shown in Figure 11-1.

Running the simulation

Run PSpice 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

Shortcut: Click on or press

After setting up the analyses, start the simulation by selecting Simulate from the Analysis menu.

Using performance analysis to plot overshoot and rise time

After performing the PSpice simulation that creates the data file called `rlcfilt.dat`, you can run Probe to compute the specified performance analysis goal functions.

When Probe is started, you are presented with a list of all the sections or runs in the Probe data file produced by PSpice. To use the data from every run, select All and click on OK in the Available Selections dialog box. In the case of Figure 11-2, the trace I(L1) from the ninth section was added by specifying the following in the Add Traces dialog box:

```
I(L1)@9
```

To access the Add Traces dialog box, select Traces from the Add menu in Probe.

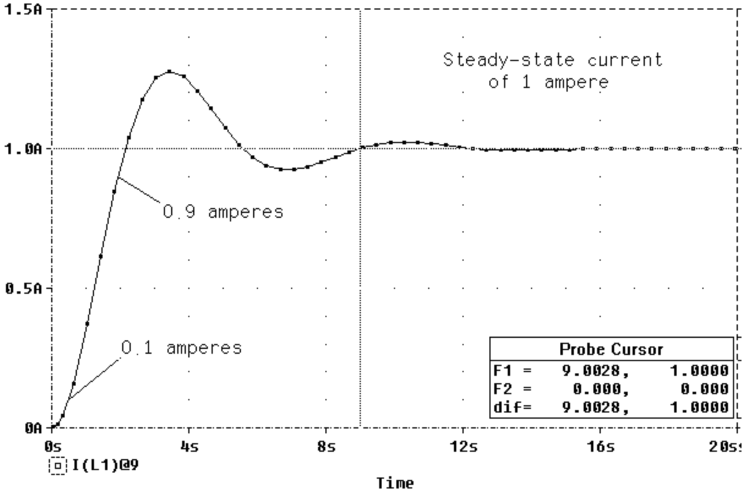


Figure 11-2 *Current of L1 when R1 is 1.5 Ohms*

To run performance analysis:

1. Select X Axis Setting from the Plot menu in Probe.
2. Click on the Performance Analysis check box so that it is checked on (enabled), then click on OK.

Probe resets the X axis variable for the graph to be the parameter that changed between PSpice runs. In the example, this is the R parameter.

To see the rise time for the current through the inductor L1, select the Add from the Trace menu and then enter:

```
genrise( I(L1) )
```

Troubleshooting tip

More than one PSpice run or data section is required for performance analysis because one data value is derived for each waveform in a related set of waveforms. A trace of the derived values cannot be displayed if there is only one run or data section because at least two data points are required to produce a trace.

However, you can use Eval Goal Functions on the Trace menu in Probe to evaluate a goal function on a single waveform, thus producing a single data point result.

The genrise and overshoot goal functions are contained in the file `msim.prb` in the MSIM directory.

In Figure 11-3, we can see 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. We now select Add from the Trace menu and enter:

```
overshoot( I(L1) )
```

Figure 11-3 shows how the overshoot increases with increasing resistance.

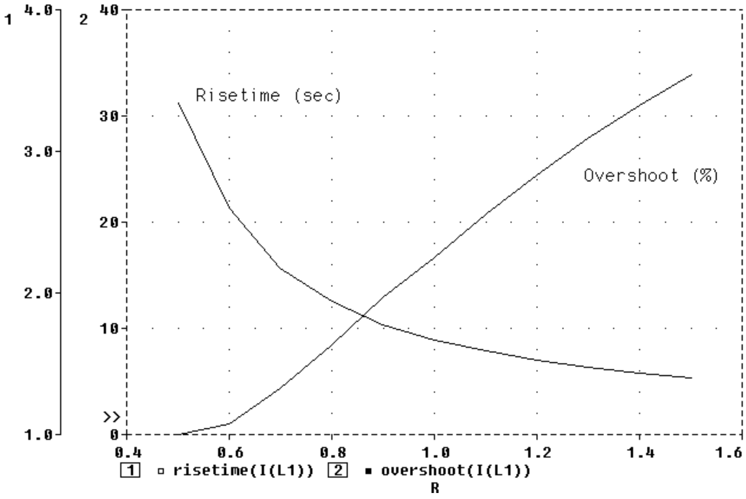


Figure 11-3 *Rise Time and Overshoot vs. Damping Resistance*

Now we can use the multiple X axes feature to view the original waveform family for inductor L1 current along with the derived rise time and overshoot data. We must first add a new plot by selecting Add Plot from the Plot menu. To set this plot’s X axis to a unique scale, select Unsync Plot from the Plot menu. You’ll notice that the new plot’s X axis is now labeled with range and variable information. However, it is still set for Performance Analysis (with resistance R as the X axis label). We can toggle off the Performance Analysis feature by selecting X Axis Settings from the Plot menu and disabling the Performance Analysis check box. This affects only the current plot. (The plot marked with SEL>> is the current plot.)

Now we can add the trace for I(L1) as we’ve done before (Add on the Trace menu), changing the Y axis range to 0A - 1.5A (Y

Axis Settings on the Plot menu), and the X axis range to 0s - 20s (X Axis Settings on the Plot menu). This produces the display shown in Figure 11-4.

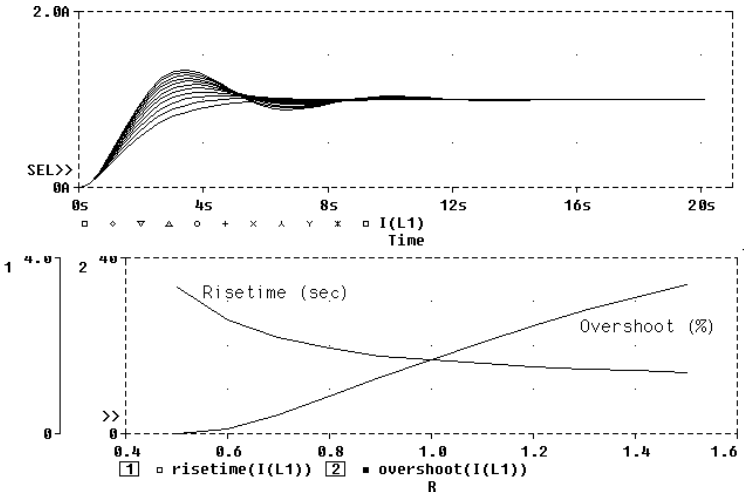


Figure 11-4 *Inductor Waveform Data Viewed with Derived Rise Time and Overshoot Data*

Example: Frequency Response vs. Arbitrary Parameter

Engineers often want to see 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 vs. its bias voltage. The capacitance is in parallel with a resistor, so a Probe expression is used to calculate the capacitance from the complex admittance of the R-C pair.

This technique for measuring branch capacitances works well in both simple and complex circuits.

Setting up the circuit

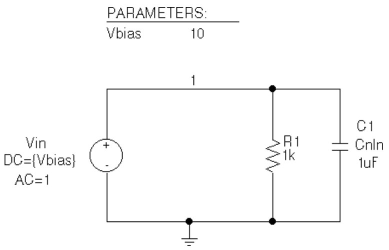


Figure 11-5 *RLC Filter Example Circuit*

Enter the circuit in Schematics as shown in Figure 11-5

To create the capacitor model in the schematic editor:

1. Place a Cbreak symbol.
2. Select it so that it is highlighted.
3. Select Model from the Edit menu.
4. Select Edit Instance Model Text. Enter the following:

```
.model Cnln CAP(C=1 VC1=-0.01 VC2=0.05)
```

Set up the circuit for a parametric AC analysis (sweep Vbias), and run PSpice. Include only the frequency of interest in the AC sweep.

Displaying results in Probe

Use Probe to display the capacitance calculated at the frequency of interest vs. the stepped parameter.

After analyzing the circuit with PSpice:

1. Run Probe.
2. Load all AC analysis sections.
3. Select Add from the Trace menu.

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 Macro on the Trace menu. The CvF macro should be defined as:

```
CvF(G)= IMG(G)/(2*3.1416*Frequency)
```

Note that -I(Vin)/V(1) is the complex admittance of the R-C branch; the minus sign is required for correct polarity.

To use performance analysis to plot capacitance vs. bias voltage

1. In Probe, select Performance Analysis from the Trace menu.
2. Click on Wizard.
3. Click on Next>.
4. Click on YatX in the Choose a Goal Function list, and then click on Next>.
5. In the Name of Trace text box, type the following:

   ```
   CvF(-I(Vin)/V(1))
   ```

6. In the X value text box, type `10K`.
7. Click on Next>. The wizard displays the gain trace for the first run to text the goal function (YatX).
8. Click on Finish. The resultant Probe plot is shown in Figure 11-6.

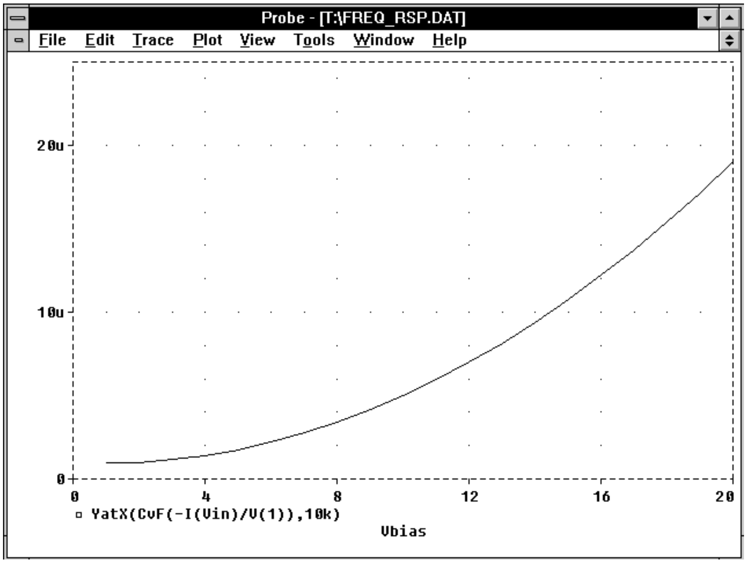


Figure 11-6 *Probe Plot of Capacitance vs. Bias Voltage*

Temperature Analysis

Minimum Requirements to Run a Temperature Analysis

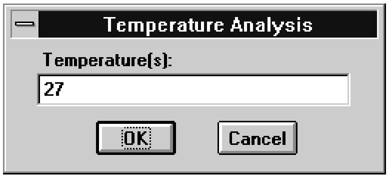
Minimum circuit design requirements

None.

Minimum software setup requirements

- In the Analysis Setup dialog box, click on the Temperature button. Specify the temperature or list of temperatures in the Temperature Analysis dialog box.
- If needed, click on the Temperature check box in the Analysis Setup dialog box so that it is checked on (enabled).
- Start the simulation as described in Starting Simulation on page 7-11.

See Setting Up Analyses on page 7-3 for a description of the Analysis Setup dialog box.



Temperature(s): 27

Overview of Temperature Analysis

When a temperature analysis is run, PSpice reruns standard analyses enabled in the Analysis Setup dialog box at different temperatures.

Temperature analysis allows zero or more temperatures to be specified. If no temperature is specified, the circuit is run at 27°C. If more than one temperature is listed, the effect is the same as running the simulation several times, once for each temperature in the list.

Running multiple analyses for different temperatures can also be achieved using parametric analysis (see Parametric Analysis on page 11-2). With parametric analysis, the temperatures can be specified either by list, or by range and increments within the range.

Setting the temperature to a value other than the default results in recalculating the values of temperature-dependent devices. In `example.sch` (see Figure 11-7), 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.

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.

The example circuit `example.sch` is provided with the MicroSim software installation.

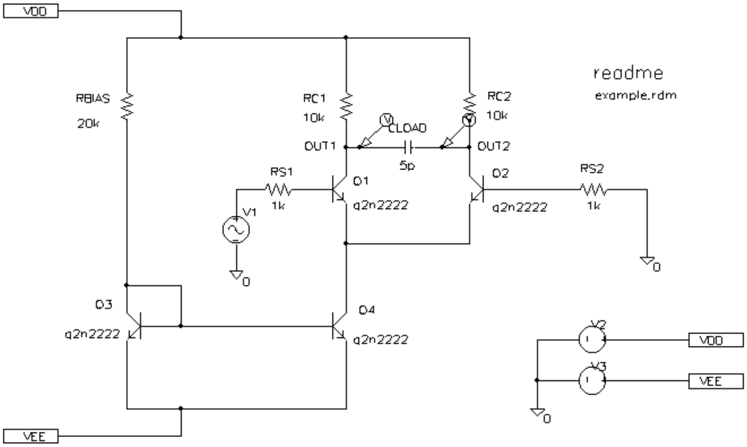


Figure 11-7 *Example Schematic example.sch*

Monte Carlo and Sensitivity/ Worst-Case Analyses

12

Chapter Overview

This entire chapter describes features which are not included in PSpice Basics.

This chapter describes how to set up Monte Carlo and sensitivity/worst-case analyses and includes the following sections:

Statistical Analyses on page 12-2

Monte Carlo Analysis on page 12-7

Worst-Case Analysis on page 12-21

not
included
in: Basics+

Statistical Analyses

Monte Carlo and sensitivity/worst-case are statistical analyses. This section describes information common to both types of analyses.

See Monte Carlo Analysis on page 12-7 for information specific to Monte Carlo analyses, and see Worst-Case Analysis on page 12-21 for information specific to sensitivity/worst-case analyses.

Overview of Statistical Analyses

The Monte Carlo and worst-case analyses 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 causes a Monte Carlo (statistical) analysis of the circuit to be performed.

A worst-case analysis causes a sensitivity and worst-case analysis of the circuit to be performed.

Sensitivity/worst-case analyses are different from Monte Carlo analysis in that they compute the parameters using the sensitivity data rather than 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 analysis selected is repeated in subsequent passes of the analysis.

Generating statistical results for Probe

As the number of Monte Carlo or worst-case runs increase, simulation takes longer and the Probe data file gets larger. Large Probe data files may be slow to open and slow to draw traces.

One way to avoid this problem is to set up an overnight batch job to run the simulation and execute Probe commands. You can even set up the batch job to produce a series of plots on paper which are ready for you in the morning.

Output Control for Statistical Analyses

Monte Carlo and sensitivity/worst-case analyses can 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.
- A summary of all the runs using a collating function.

Output is saved to the Probe data file for use by the Probe graphical waveform analyzer. For Monte Carlo analyses, Probe offers a special facility through the performance analysis feature to produce histograms of derived data.

For information about performance analysis, see Example: RLC Filter on page 11-3.

For information about histograms, see Creating histograms on page 12-17.

Model Parameter Values Reports

The List option in the MC Options section of the Monte Carlo or Worst Case dialog box produces a list of the model parameters actually used for each run.

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.

Note that for medium and large circuits, the List option can produce a large output file.

Waveform Reports

For Monte Carlo analyses, there are four variations of the output which can be specified in the Output section of the Monte Carlo or Worst Case dialog. These options are:

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*	Does specified analysis and generates outputs only for the listed runs. Up to 25 values can be specified in the list.

In excess of about 10 runs, the Probe display tends to become more of 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 becomes larger, the spread more closely approximates the actual worst-case limits for the circuit.

The * indicates that you can specify runs in the Value text box.

Values for the output variables specified in the selected analyses are saved to the simulation output file and Probe data file. Note that even a modest number of runs can produce large output files.

Collating Functions

You may want to compress the results of Monte Carlo and worst-case analyses further. Using the collating function specified in the Function section of the Monte Carlo or Worst Case dialog box, each run can be represented by a single number. A table of deviations per run is reported in the simulation output file.

Collating functions are listed in **Table 12-1**.

Table 12-1 *Collating Functions Used in Statistical Analyses*

Function	Description
YMAX	Find the greatest difference in each waveform from the nominal
MAX	Find the maximum value of each waveform

Table 12-1 *Collating Functions Used in Statistical Analyses*

Function	Description
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

Temperature Considerations in Statistical Analyses

Refer to *Temperature Effects on Monte Carlo Analysis* in the *Application Notes* manual for more information on this topic.

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, one quickly gets into detailed questions about how the two interact. Therefore, we recommend not enabling temperature analysis when using the Monte Carlo or worst-case analyses.

Also, it will not work to sweep the temperature in a DC sweep analysis while performing one of these statistical analyses, or to put tolerances on temperature coefficients. You will notice in `example.sch` how the temperature value is fixed at 35°C.

The example circuit `example.sch` is provided with the MicroSim software installation.

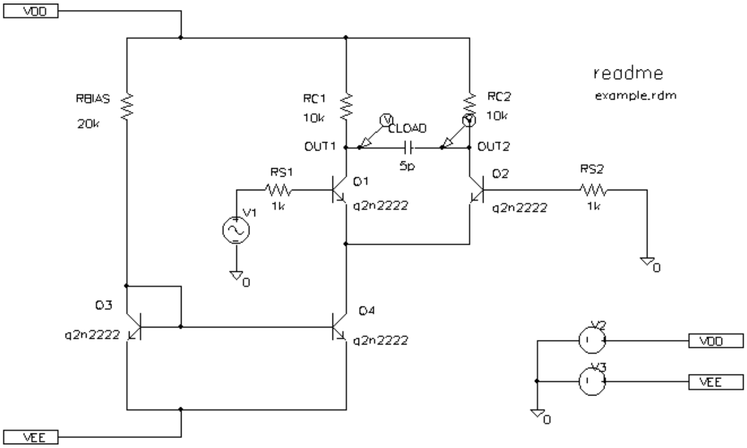


Figure 12-1 *Example Schematic example.sch*

Monte Carlo Analysis

not included in: **Basics+**

The Monte Carlo analysis computes the circuit response to changes in component values by randomly varying all of the device model parameters for which a tolerance is specified. This provides statistical data on the impact of a device parameter's variance.

With Monte Carlo analysis model parameters are given tolerances, and multiple analyses (DC, AC, or transient) are run using these tolerances. A typical application of Monte Carlo analysis is predicting yields on production runs of a circuit.

For `example.sch` in Figure 12-1 on page 12-6, effects due to variances in resistors RC1 and RC2 values can be analyzed by assigning a model description to these resistors that includes a 5% device tolerance on the multiplier parameter R. Then, you can run a Monte Carlo analysis that runs a DC analysis first with the nominal R multiplier value for RC1 and RC2, then the specified number of additional runs with the R multiplier varied independently for RC1 and RC2 within 5% tolerance.

To modify example.sch and set up simulation

1. Replace RC1 and RC2 with RBREAK symbols, setting attribute values to match the resistors that are being replaced (VALUE=10k) and reference designators to match previous names.

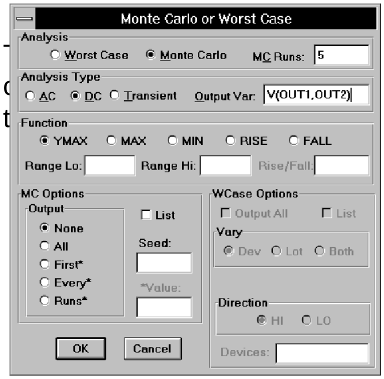
2. Select Model from the Edit menu, then select Edit Instance Model. Create the model CRES as follows:

   ```
   .MODEL CRES RES( R=1 DEV=5% TC1=0.02 TC2=0.0045 )
   ```

   By default, MicroSim Schematics saves the definition to the model file `example.lib` and automatically configures the file for local use with the current schematic.

3. Click on the resistor instance. Select Model from the Edit menu, then select Change Model Reference to change the model reference to CRES.

4. Set up a new Monte Carlo analysis as shown in Figure 12-2. The analysis specification instructs PSpice to do one nominal run and four Monte Carlo runs, saving the DC analysis output from those five runs.



Monte Carlo or Worst Case

Analysis: ○ Worst Case ● Monte Carlo MC Runs: 5

Analysis Type: ○ AC ● DC ○ Transient Output Var: V(OUT1,OUT2)

Function: ● YMAX ○ MAX ○ MIN ○ RISE ○ FALL

Range Lo: Range Hi: Rise/Fall:

MC Options — Output: ● None ○ All ○ First* ○ Every* ○ Runs* ☐ List Seed: *Value:

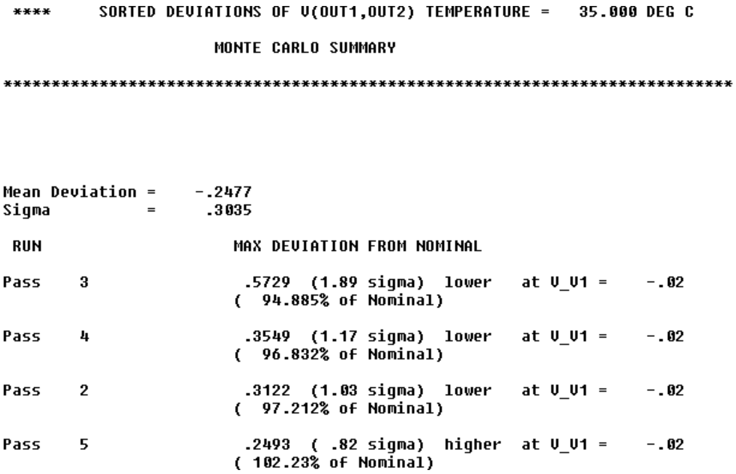
WCase Options: ☐ Output All ☐ List — Vary: ● Dev ○ Lot ○ Both — Direction: ● HI ○ LO — Devices:

Figure 12-2 *Monte Carlo Analysis Setup for example.sch*

PSpice starts as usual by running *all* of the analyses enabled in the Analysis Setup dialog with all parameters set to their nominal values. However, with Monte Carlo enabled, the DC sweep analysis results are saved for later reference and comparison.

After the nominal analyses are finished, more of the specified analysis runs are performed (DC sweep in this example). Subsequent runs use the same analysis specification as the nominal with one major exception. **Instead of using the nominal parameter values, the tolerances are applied to set new parameter values and thus, new component values.**

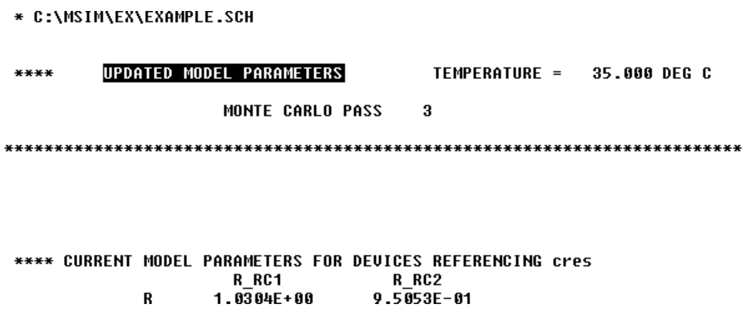
The summary report generated in this example specifies that the waveform generated from V(OUT1, OUT2) should be the subject of the collating function YMAX. In each of the last four runs, the new V(OUT1, OUT2) waveform is compared to the nominal V(OUT1, OUT2) waveform for the first run, calculating the maximum deviation in the Y direction (YMAX collating function). The deviations are printed in order of size along with their run number (see Figure 12-3).



```
 ****     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 12-3 *Summary of Monte Carlo Runs for example.sch*

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 12-4), run 3 exhibits the highest deviation.



```
 * C:\MSIM\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 12-4 *Parameter Values for Monte Carlo Pass 3*

There is a trade-off in choosing the number of Monte Carlo runs. More runs provide better statistics, but take proportionally more computer time. The amount of computer 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 status display includes a line showing the run number and the total number of runs to be done. This gives an idea of how far the program has progressed.

Probe offers a facility to generate histograms of data derived from Monte Carlo waveform families through the performance analysis feature.

For information about performance analysis, see Example: RLC Filter on page 11-3.

For information about histograms, see Creating histograms on page 12-17.

Tutorial: Monte Carlo Analysis of a Pressure Sensor

In this tutorial, the Monte Carlo analysis features provided in Schematics and PSpice are demonstrated for a pressure sensor circuit using a pressure-dependent resistor bridge. We will investigate how the performance of the circuit is affected by manufacturing tolerances.

Drawing the schematic

The bridge is constructed according to the diagram shown in Figure 12-5. Both V1 and Meter are generic voltage sources (Get New Part VSRC on the Draw menu).

The remaining parts are resistors (Get New Part R on the Draw menu). To rotate parts, use Ctrl + R while placing. To move values and/or reference designators, click on the value or reference designator to select it, then hold the mouse button down and drag it to the desired location; release the mouse button to place.

To specify values for the resistors, double-click on the current value (the default is 1K) and type the new value in the dialog provided. The value which is given for R3 (example: {1k*(1+P*Pcoeff/Pnom)}, where Pcoeff=-0.06, P=0 and is swept, and Pnom=1.0) is an expression that represents linear dependence of resistance on pressure.

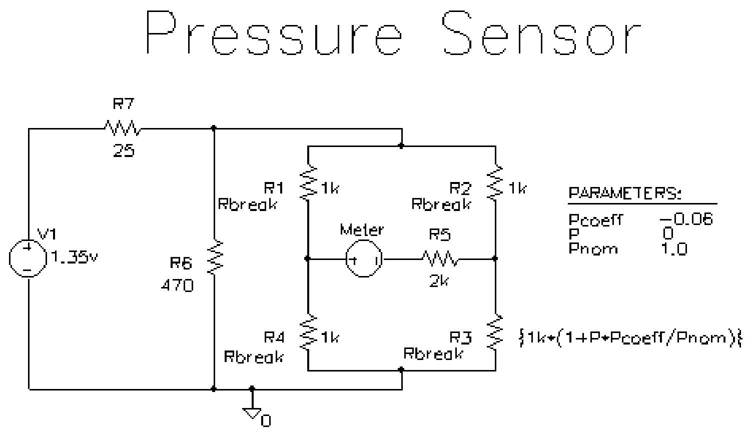


Figure 12-5 *Pressure Sensor Circuit*

To set the value of the DC attribute on the V1 source, double-click on the V1 source, then double-click on the DC= line in the Edit Attributes dialog, type `1.35v` in the Value text box, and click on the Save Attr button (or press Enter) to accept the changes. To make the value of the DC attribute (example: 1.35v) visible on the schematic, click on the Change Display button, and select the Value radio button in the Display box. Click on OK to accept the change. Click on OK to end the dialog and return to the schematic. The Meter source has no DC value since it is used only to measure current. The analog ground symbol is placed by using Get New Part on the Draw menu to place the AGND symbol.

Shortcut: Click on or press

Setting up the parameters

To set up the parameters Pcoeff, P, and Pnom, place a PARAM symbol on the schematic (using Get New Part on the Draw menu). Double-click on the PARAM symbol, then double-click on each of the NAMEn, VALUEn pairs (where n=1, 2, 3) to define the parameters and their values.

For example, double-click on NAME1 and type `Pcoeff` in the Value text box; press Enter or click on the Save Attr button to accept the changes. Then, double-click on VALUE1 and type `-0.06` in the Value text box; press Enter or click on the Save Attr button to accept the changes. When you are finished defining all three parameters, click on OK.

Varying the resistors

Once we've decided that we want to vary the resistors, and which resistors we want to vary, we will need to replace the desired R symbols on the page with Rbreak symbols (from `breakout.slb`). This is because the R symbol does not have an associated model, and Monte Carlo analysis operates on device (DEV) and lot (LOT) tolerances applied to model parameters.

The Rbreak symbol does have an associated model, specified using the MODEL attribute on the symbol. Replace each of R1 through R4 by selecting them (single-click), selecting Replace from the Edit menu, using Rbreak as the Replacement, and indicating Selected Parts Only. Click on OK to perform each replacement operation.

Note *You can also select more than one resistor at a time using a selection box and/or pressing* Shift*+click to add more selections to the group of selected items. Remember that you can move the attributes by clicking once to select it, then holding the mouse button down and dragging it to the desired location; release the mouse button to place.*

Saving the schematic

Before we can define models for the Rbreak resistors, we need to save the schematic by selecting Save As from the File menu. Type `psensor.sch` in the File Name text box, then click on OK.

Modifying the models for Monte Carlo

See Using the Model Editor (Text Editor) on page 5-21 for information about the model editor.

Now we need to define models for the resistors we are varying, and assign device (DEV) and lot (LOT) tolerances to the model parameters. The model editor allows us to edit model definitions of part instances. In this way, we can add whatever device and/or lot tolerances we may want to the model parameters of a given part.

Select R1, then select Model from the Edit menu. The displayed dialog box presents three main choices: Change Model Reference, Edit Instance Model (Text), and Edit Instance Model (Parts).

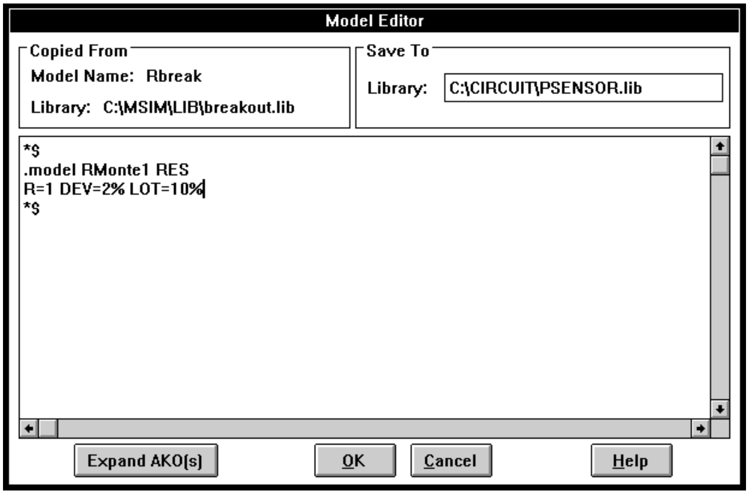
The first selection allows us to change the model referenced by that symbol. For example, if we already had a model called MyModel, we could change R1 to reference MyModel, instead of Rbreak, by clicking on the Change Model Reference button and typing MyModel in the text box provided.

The second selection, Edit Instance Model (Text), allows us to edit the model definition for that symbol instance using a text editor. That is, when we select Model from the Edit menu for R1, the model definition which appears in the model editor dialog box is a copy of the RBREAK model. We cannot edit the RBREAK model definition associated with every RBREAK part placed; that can only be done using the symbol editor (Symbol on the Edit menu).

The third selection, Edit Instance Model (Parts), allows us to edit the model definition for that symbol instance using the Parts utility.

Now click on the Edit Instance Model (Text) button. The message `Searching libraries, please wait...` appears in the status line at the bottom of the Schematics window. By default, the new name of the instance model is <*old model name*>-X (example: Rbreak-X); we can change this to whatever we like. Let's change it to RMonte1. Move the cursor to the beginning of Rbreak-X on the .model Rbreak-X RES line, press Shift+Ctrl+→ to select Rbreak-X, and type `RMonte1` in its place.

To add both a 2% device tolerance and a 10% lot tolerance to the resistance multiplier of this model, add R=1 DEV=2% LOT=10% on the line following the .MODEL statement. Your model editing box should look something like Figure 12-6.



Model Editor

Copied From

Model Name: Rbreak

Library: C:\MSIM\LIB\breakout.lib

Save To

Library: C:\CIRCUIT\PSENSOR.lib

```
*$
.model RMonte1 RES
R=1 DEV=2% LOT=10%
*$
```

Figure 12-6 *Model Definition for RMonte1*

Click on OK to accept the changes.

When we have finished editing this instance model (RMonte1), click on OK. The .MODEL definition of the RMonte1 is saved (by default) to the <*schematic name*>`.lib` model file—`psensor.lib` in this case. This file is automatically added to the set of model files associated with (local to) this schematic. If we specified a model name which was contained in a library/ include file not already configured (via Library and Include Files on the Analysis menu), we would have to explicitly configure it in order to be able to use the model during simulation (see Configuring the Library on page 5-36).

Now that RMonte1 is already defined, and saved to `psensor.lib`, we can reference this new model definition, by name, for R2 and R4. Select each of these symbols, one at a time, select Model from the Edit menu, and click on the Change Model Reference button. Type `RMonte1` in the text box provided.

For resistor R3, we would like to assign a 5% DEV tolerance and no LOT tolerance. For this, we will need a new instance model. Select Model from the Edit menu, and click on the Edit Instance Model (Text) button. Enter `RTherm` as the name of the instance model. Type `R=1` DEV=5% on the line following the .model RTherm RES line, and click on OK to accept the changes. Your schematic should look like Figure 12-7.

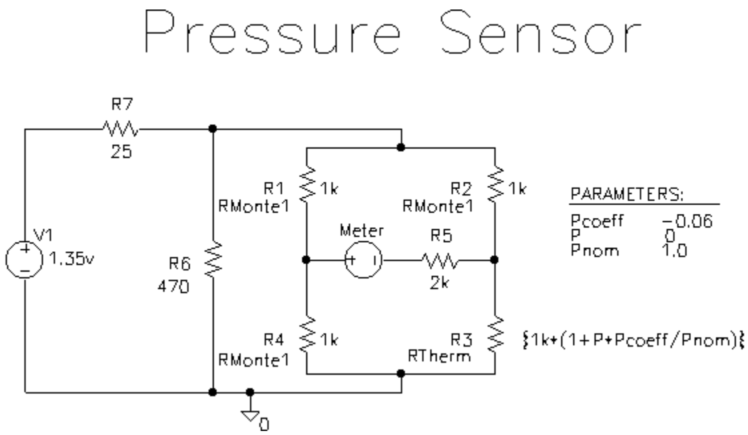


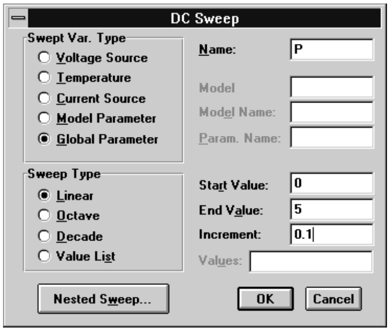
Figure 12-7 *Pressure Sensor Circuit with RTherm Model Definition Edited*

Setting up the analyses

First, let’s set up a DC analysis. We would like to sweep the value of pressure, which we have defined as P. Select Setup from the Analyses menu and then click on the DC Sweep button. Select Global Parameter in the Swept Var. Type box, type `P` in the Name text box, type `0` in the Start Value text box, `5.0` in the End Value text box, and `0.1` in the Increment text box as shown in Figure 12-8. Click on OK.

Now, let’s set up the Monte Carlo analysis. Select Setup from the Analysis menu and click on Monte Carlo/Worst Case. Type `10` in the MC Runs text box, select DC and type `I(Meter)` in the Output Var text box of the Analysis Type box, and select All in the Output function section of the MC Options box as shown in Figure 12-9. Click on OK.

To enable the analyses, click on the box to the left of the DC Sweep button and the box next to the Monte Carlo/Worst Case button. Click on OK.



DC Sweep

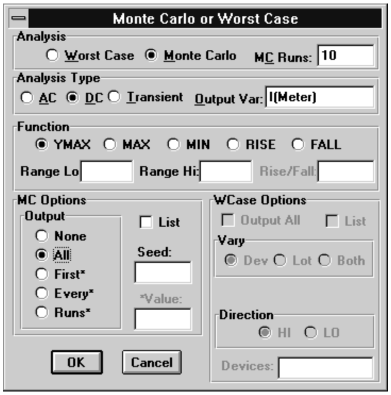
Swept Var. Type
○ Voltage Source
○ Temperature
○ Current Source
○ Model Parameter
● Global Parameter

Name: P
Model:
Model Name:
Param. Name:

Sweep Type
● Linear
○ Octave
○ Decade
○ Value List

Start Value: 0
End Value: 5
Increment: 0.1
Values:

Figure 12-8 *DC Analysis Setup Example*



Monte Carlo or Worst Case

Analysis
○ Worst Case ● Monte Carlo MC Runs: 10

Analysis Type
○ AC ● DC ○ Transient Output Var: I(Meter)

Function
● YMAX ○ MAX ○ MIN ○ RISE ○ FALL
Range Lo: Range Hi: Rise/Fall:

MC Options
Output
○ None
● All
○ First*
○ Every*
○ Runs*
☐ List
Seed:
*Value:

WCase Options
☐ Output All ☐ List
Vary
● Dev ○ Lot ○ Both
Direction
● HI ○ LO
Devices:

Figure 12-9 *Monte Carlo Analysis Setup Example*

Running the analysis and viewing the results

To start the simulation, select Simulate from the Analysis menu. If “Automatically Run Probe After Simulation” is enabled with Probe Setup on the Analysis menu, then a Probe window is displayed upon completion of the successful simulation.

Since it was a Monte Carlo analysis that we performed, we are asked to select the data sections (which runs) to display. Click on All and then on OK to view all sections.

To display the desired traces, bring the Schematics window back into the foreground (either by clicking on a visible portion of it, or by pressing Ctrl+Esc and selecting it from the task list), select Mark Current into Pin from the Markers menu, and place a current marker on the left-hand pin of the Meter source.

Return to the Probe window to see the family of curves for I(Meter) against P. To view the family of curves in Probe without making use of markers, select Add from the Trace menu in Probe and double-click on I(Meter) in the output variable list.

For more on analyzing Monte Carlo results in Probe, see the next section on Monte Carlo histograms.

Monte Carlo Histograms

A typical application of Monte Carlo analysis is predicting yields on production runs of a circuit. Probe can be used to display data derived from Monte Carlo waveform families as histograms, part of Probe's performance analysis feature.

For information about performance analysis, see Example: RLC Filter on page 11-3.

To illustrate this feature, we will 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 goal functions for bandwidth and center frequency, we will 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 schematic for the filter is shown in Figure 12-10. The stimulus specifications for V1, V2, and V3 are:

V1: DC=-15
V2: DC=+15
V3: AC=1

The components were rounded to the nearest available 1% resistor and 5% capacitor value. In our analysis, we are concerned with how the bandwidth and the center frequency vary when 1% resistors and 5% capacitors are used in the circuit.

Creating models for Monte Carlo analysis

Since we are interested in varying the resistors and capacitors in the filter circuit, we will need to create models for these components on which we can set some device tolerances for Monte Carlo analysis. The `breakout.slb` Symbol Library file contains generic devices for this purpose. The resistors and capacitors in this schematic are the Rbreak and Cbreak symbols from `breakout.slb`. Using the model editor, we must modify the models for these components as follows:

```
.model RMOD RES(R=1 DEV=1%)
.model CMOD CAP(C=1 DEV=5%)
```

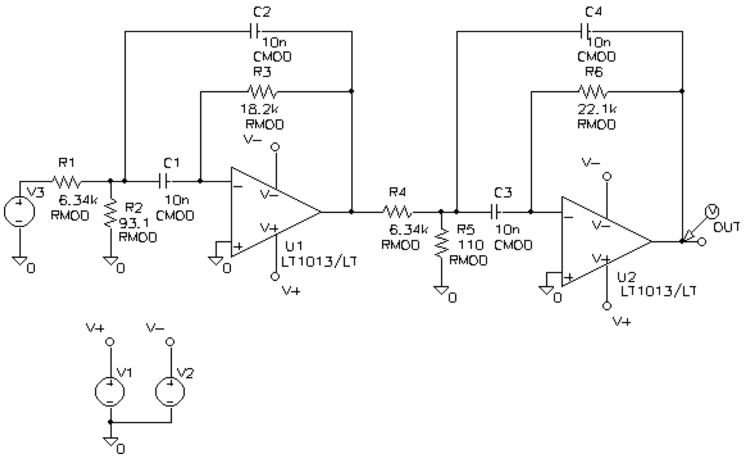
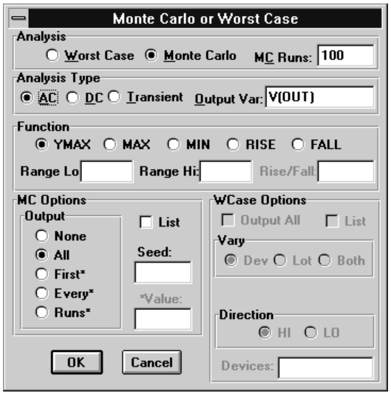


Figure 12-10 *Chebyshev Filter*

Setting up the analysis

To analyze our filter, we will 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 (see Figure 12-11). The analysis type is AC and the output variable that we are interested in is V(OUT). We will select All in the MC Options box.



Analysis

○ Worst Case ● Monte Carlo MC Runs: 100

Analysis Type

● AC ○ DC ○ Transient Output Var: V(OUT)

Function

● YMAX ○ MAX ○ MIN ○ RISE ○ FALL

Range Lo: Range Hi: Rise/Fall:

MC Options

Output

○ None
● All
○ First*
○ Every*
○ Runs*

☐ List

Seed:

*Value:

WCase Options

☐ Output All ☐ List

Vary

● Dev ○ Lot ○ Both

Direction

● HI ○ LO

Devices:

Figure 12-11 *Monte Carlo Analysis Setup Example*

Creating histograms

Because the data file can become quite large when running a Monte Carlo analysis and because we are only interested in the output of the filter, we will place a voltage marker at the output of the filter. The steps necessary to collect data for the marked node only are enumerated below.

1. Select Probe Setup from the Analysis menu.
2. Mark the Automatically Run Probe after Simulation check box.
3. Select At Markers Only in the Data Collection box.
4. Click on OK.

To run the simulation, select Simulate from the Analysis menu. After the simulation is complete, a Probe window is displayed.

Shortcut: Click on or press

Since we performed a Monte Carlo analysis, we are asked to select the runs for which we wish to display the data. Click on All and then on OK to view all sections.

The steps to display a histogram are enumerated below.

For information about performance analysis, see Example: RLC Filter on page 11-3.

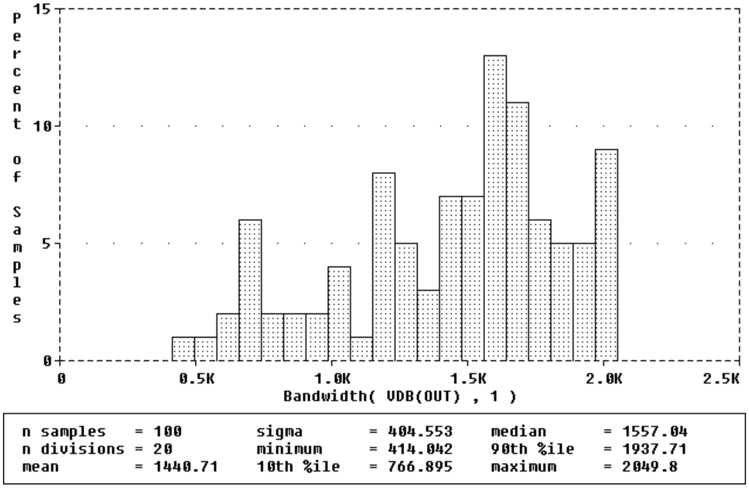
1. Select X Axis Settings from the Plot menu.
2. Check the Performance Analysis check box in the Processing Options box and click on OK.

The display changes to the histogram display where the Y axis is the percent of samples. To display a histogram of the distribution of the 1 dB bandwidth for our filter:

You can also display this histogram by using the performance analysis wizard to display Bandwidth (VDB(OUT) , 1).

1. Select Add from the Trace menu.
2. Click on the Bandwidth(1, db_level) goal function.
3. Click on V(OUT).
4. In the Trace Command text box, place the cursor after the V in Bandwidth(V(OUT) ,) and type `DB`. The text box should now read as `Bandwidth(VDB(OUT) , )`.
5. Place the cursor after the comma and type `1` for the 1 dB level. The text box should now read as `Bandwidth(VDB(OUT) , 1)`.
6. Click on OK to view the histogram.

To change the number of histogram divisions, select Options from the Tools menu and replace 10 with 20 in the Number of Histogram Divisions text box. Click on Save and then OK. The histogram of the 1 dB bandwidth is as shown in Figure 12-12.



```
n samples   = 100       sigma      = 404.553    median     = 1557.04
n divisions = 20        minimum    = 414.042    90th %ile  = 1937.71
mean        = 1440.71   10th %ile  = 766.895    maximum    = 2049.8
```

Figure 12-12 *1 dB Bandwidth Histogram*

The statistics for the histogram are displayed along the bottom of the display by default. They can be turned off by selecting Options from the Tools menu, clicking on the Display Statistics check box to remove the X, and clicking on Save and OK.

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 goa1 function values are less than or equal to the 10th percentile number, and 90% of the goal function values are greater than or equal to that number.

If there is more than one goal function value that satisfies this criteria, then the 10th percentile is the midpoint of the interval between the goal function values that satisfy the criteria. Similarly, the median and 90th percentile numbers represent goal function values such that 50% and 90% (respectively) of the goal function values are less than or equal to those numbers. Sigma is the standard deviation of the goal function values.

We can also show the distribution of the center frequency of our filter. The steps to display the center frequency are enumerated below.

1 Select Add from the Trace menu.

2 Select the CenterFreq(1, db_level) goal function by clicking on it.

3 Select V(OUT) by clicking on it.

4 In the Trace Command text box, place the cursor after the V in CenterFreq(V(OUT) ,) and type `DB`. The text box should now read as follows:

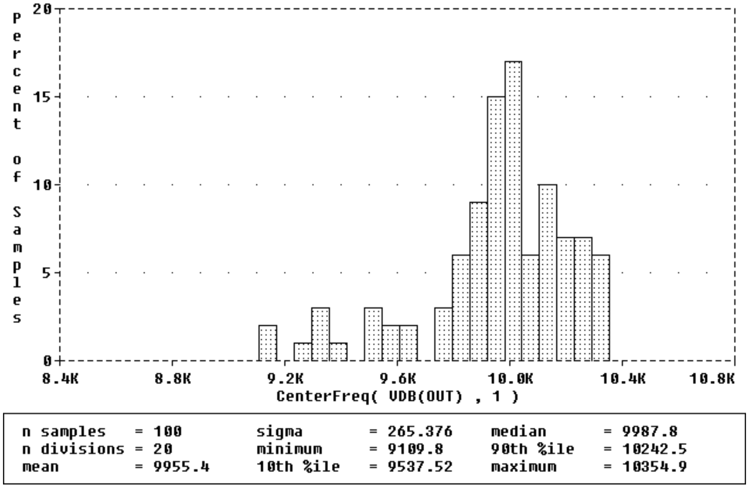
```
CenterFreq(VDB(OUT) , )
```

5 Place the cursor after the comma and type in `1` for the 1 dB level. The text box should now read as follows:

```
CenterFreq(VDB(OUT) , 1)
```

6 Click on OK to view the histogram.

The new histogram replaces the previous histogram. To display both histograms at once, use Add Plot on the Plot menu before selecting Add from the Trace menu. The histogram of the center frequency is as shown in Figure 12-13.



```
n samples   = 100         sigma        = 265.376     median      = 9987.8
n divisions = 20          minimum      = 9109.8      90th %ile   = 10242.5
mean        = 9955.4      10th %ile    = 9537.52     maximum     = 10354.9
```

Figure 12-13 *Center Frequency Histogram*

Worst-Case Analysis

not included in: Basics+

This section discusses the analog worst-case analysis feature of PSpice. It is hoped that the information provided here will help you to apply it properly and with realistic expectations.

Overview of Worst-Case Analysis

Worst-case 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

Besides the circuit description, two forms of information are required from the user:

- parameter tolerances
- a definition of what *worst* means

PSpice allows tolerances to be set on any number of the parameters that characterize a model. Models can be defined for nearly all primitive analog circuit components, such as resistors, capacitors, inductors, and semiconductor devices. PSpice reads the standard model parameter tolerance syntax specified in the .MODEL statement. For each model parameter, PSpice 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, the measurement must be reduced to a single value by one of these five collating functions:

Analog behavioral models can be used to measure waveform characteristics other than those detected by the available collating functions, such as rise time or slope. Analog behavioral models can also be used to incorporate several voltages and currents into one output variable to which a collating function may be applied. See **Chapter 6,**Analog Behavioral Modeling, for more information.

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

Worst is user-defined as the highest (HI) or lowest (LO) possible collating function relative to the nominal run.

Procedure

To establish the initial value of the collating function, worst-case analysis begins with a nominal run with 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).

This procedure saves time by performing the minimum number of simulations required to make an educated guess at the parameter values which produce the worst results. It also has some limitations, which will be discussed in the following sections.

Outputs

A summary of the sensitivity analysis is printed in the PSpice 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 with Probe.

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 which 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. Otherwise, there is no guarantee. Usually you cannot be certain if this condition is true, but insight into the operation of the circuit may alert you to possible anomalies.

Worst-Case Analysis Example

The schematic shown in Figure 12-14 is for an amplifier circuit which 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. Since 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*. Since the value of *Rb2* determines the bias on the BJT, it also affects the amplifier's gain.

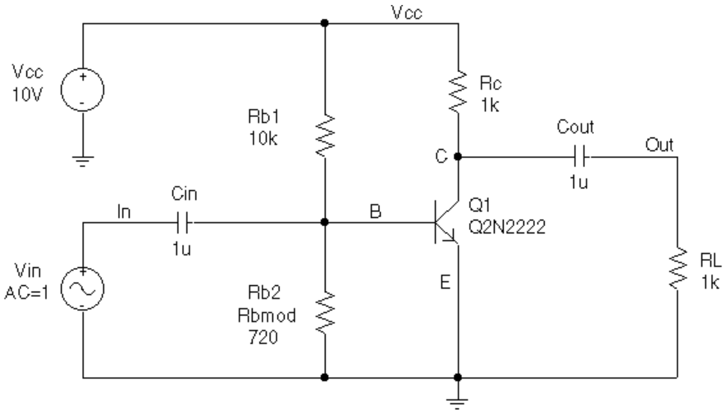


Figure 12-14 *Simple Biased BJT Amplifier*

Figure 12-16 is the circuit file used to run *either* 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 to determine the worst-case value for parameter R based upon a sensitivity analysis. PSpice allows only one of these analyses to be run in any given simulation. Note that 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 in Figure 12-16 is 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 (see Figure 12-17 and Figure 12-18).

For demonstration, the parametric analysis is run first, generating the curve shown in Figure 12-17 and Figure 12-18. This curve, derived using the YatX goal function shown in Figure 12-15, illustrates the non-monotonic dependence of gain on *Rb2*. To do this yourself, place the goal function definition in a `probe.gf` file in the circuit directory. Then run Probe, load all of the AC sweeps, set up the X axis for performance analysis, and add the following trace:

```
YatX(Vm([OUT]),100k)
```

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).

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]) (see Figure 12-17).

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 (see Figure 12-18). Note that even an optimizer, which checks the local gradients to determine how the parameters should be varied, is fooled by this circuit.

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

```
YatX(1, X_value)=y1
{
  1|sfxv(X_value)!1;
}
```

Figure 12-15 *YatX Goal Function*

The YatX goal function is used on the simulation results for the parametric sweep (.STEP) defined in Figure 12-16. The resulting curves are shown in Figure 12-17 and Figure 12-18.

```
* 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 *****
CoutC       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
```

Figure 12-16 *Amplifier Netlist and Circuit File*

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.

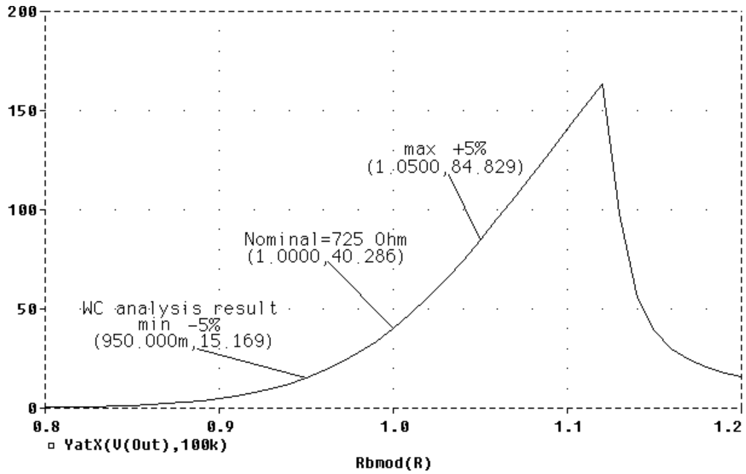


Figure 12-17 *Correct Worst-Case Results*

Output is monotonic within the tolerance range. Sensitivity analysis correctly points to the minimum value.

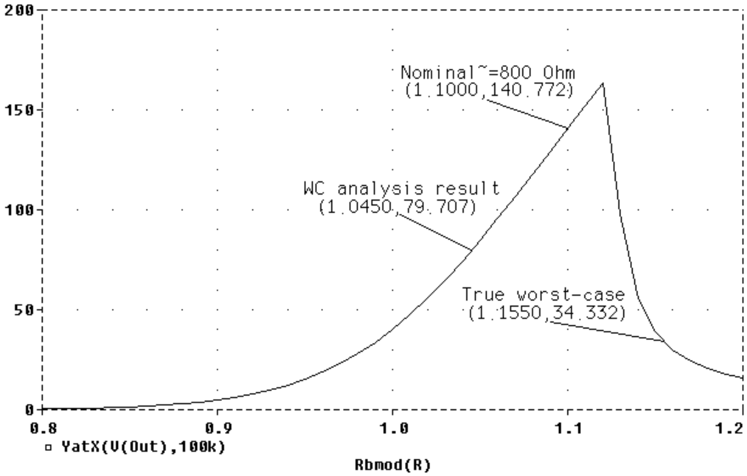


Figure 12-18 *Incorrect Worst-Case Results*

Output is non-monotonic within the tolerance range, thus producing incorrect worst-case results.

Hints and Other Useful Information

VARY BOTH, VARY DEV, and VARY LOT

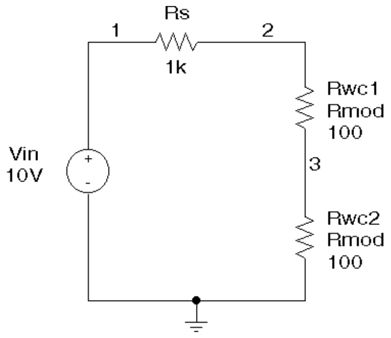


Figure 12-19 *Schematic Demonstrating Use of VARY*

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; 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, the parameter is changed for all devices by LOT + DEV in the determined direction. Consider the example schematic in Figure 12-19 and circuit file in Figure 12-20.

```
WCASE  VARY  BOTH  Test

Vin 1       0       10V
Rs  1       2       1K
Rwc12       3       Rmod    100
Rwc23       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 12-20 *Circuit File Demonstrating Use of VARY BOTH*

In this case, V(3) is maximized if:

- *Rwc1* and *Rwc2* are both increased by 10% per the LOT tolerance specification, and then
- *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%.

Here again, 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 3σ (three times sigma) for the worst-case analysis.

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. This may result in maximizing or minimizing the output variable value over the entire range of the sweep. This collating function is usually 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

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* order; for example, the collating function was its worst when the first parameter printed in the list was varied.

When the YMAX collating function is used, 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

Worst-case analysis can be used to perform *manual optimization* with PSpice. The monotonicity condition is usually met if the parameters have a very limited range. Performing worst-case analysis with tight tolerances on the parameters yields sensitivity and worst-case results (in the output file) which can be used 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. Parametric sweeps (.STEP), like the one performed in the circuit file shown in Figure 12-16, can be used to augment this procedure.

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, simply replace .WC in the circuit file with .MC <*#runs*>, where <*#runs*> is the number of simulations you are willing to perform. More runs provide higher confidence results. To save disk space, do not specify any OUTPUT options. The Monte Carlo summary in the output file lists the runs in decreasing order of collating function value.

Now add the following option to the .MC statement, and simulate again.

```
OUTPUT LIST RUNS <worst_run#>
```

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.

Part Four

Viewing Results

Part Four describes how to use Probe to view simulation results.

Chapter 13,Waveform Analysis, describes how to use Probe to perform waveform analysis of simulation results.

Chapter 14,Output Options, describes output options that can be used to control the simulation output.

Waveform Analysis

13

Chapter Overview

This chapter describes how to use Probe to perform waveform analysis of simulation results. This chapter includes the following:

Overview of Probe

MicroSim Probe is the waveform analyzer for PSpice simulations. Probe allows you to visually analyze and interactively manipulate the waveform data produced by circuit simulation.

Using high-resolution graphics, Probe allows you to view the results of a simulation both on the screen and on hard copy. In effect, Probe is a software oscilloscope. Running PSpice corresponds to building or changing a breadboard, and running Probe corresponds to looking at the breadboard with an oscilloscope.

Probe displays simple voltages and currents and also displays complex arithmetic expressions involving voltages and currents. Probe can also display the Fourier transform of these expressions.

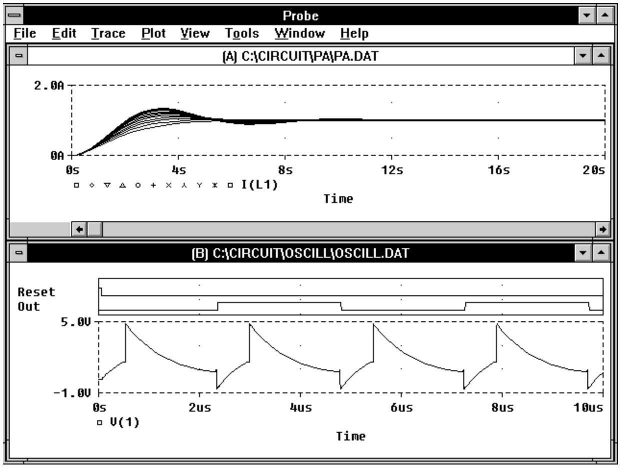
PSpice generates two forms of output: the simulation output file and the Probe data file. The calculations and results reported in the simulation output file act as an audit trail of the simulation. However, the graphical analysis of information stored in the Probe data file using the Probe program is the most informative and flexible method for evaluating simulation results.

Anatomy of a Plot Window

A plot window is a separately managed waveform display area. A plot window can include multiple plots. Figure 13-1 shows the Probe window with two plot windows displayed (toolbars disabled).

Since a plot window is a window object, it has standard window features which allow you to minimize/maximize the window or move and scale the window within the Probe window area. A toolbar can be displayed in the Probe window and applies to the currently active plot window.

Use Options on the Tools menu to display/hide the toolbar.



window A

window B (active)

Figure 13-1 *Probe Window with Two Plot Windows*

One or more Probe data files can be opened in a plot window. After the first file is loaded, other files can be loaded into the same plot window using Append on the File menu.

Managing Multiple Plot Windows

Any number of plot windows can be opened. Each plot window is an independent window.

The same Probe data file can be displayed in more than one plot window. Only one plot window is active at any given time, identified by a highlighted title bar. Menu, keyboard, and cursor operations affect only the active plot window. Another plot window can be made active by clicking anywhere within the window.

Printing multiple windows

You can print all or selected plot windows, with up to nine windows appearing on a single page. When you select Print from the File menu, a list of all current plot windows is displayed. Each window is identified by the unique identifier in parentheses in the title bar of the plot window.

The arrangement of plot windows on the page can be customized using the Page Setup dialog box. You can print in either portrait (vertical) or landscape (horizontal) orientation.

What You Need to Know to Run Probe

Starting Probe

If you are using MicroSim Schematics, you can automatically start MicroSim Probe after a simulation is run, or you start Probe separately from Windows.

When you start Probe, you can use the default `.prb` file or you can use a custom `.prb` file.

You do not need to exit Probe if you are finished examining the simulation results for one circuit and want to begin a new simulation. However, when you set up Probe to run automatically after simulation, Probe unloads the data file each time that you run a simulation. Once the simulation is complete, the new or updated Probe data file is loaded for viewing.

To automatically start Probe after simulation

1. In Schematics, select Probe Setup from the Analysis menu.
2. In the Auto-Run Option area, click on Automatically Run Probe after Simulation.
3. Click on any other options you want to use.
4. Click on OK.

To start Probe and monitor results during a simulation

1. Turn on waveform monitoring:
   - a. Select Probe Setup from the Analysis menu.
   - b. Select Monitor Waveforms (Auto-update). If this entry is grayed out, then disable the Text Data File Format (CSDF) check box.
   - c. Click on OK.
2. Select Simulate from the Analysis menu to start the simulation. Probe starts automatically and displays one window in monitor mode.
3. In Probe, select the waveforms to be monitored using Add on the Trace menu or by placing markers.

Once the simulation is complete (all data sections), the PSpice window reverts to manual mode.

If a new Probe window is opened (using New on the Window menu) while monitoring the data, the new window also starts in monitor mode since it is associated with the same Probe data file.

4. During a multiple run simulation (such as Monte Carlo, parametric or temperature), only the data for the first run is displayed. To view the curves for several runs:
    a. Select Close from the File menu to close the data file, then select Open from the File menu to reload it.
    b. Specify the data sections (runs) to load.
    c. Select the traces to monitor. Waveforms for all loaded sections are displayed.

To start Probe from Schematics

Select Run Probe from the Analysis menu, or press F12.

To start Probe in Windows

1. Display Program Manager.
2. Double-click the Probe icon in the MicroSim program group.

Setup Requirements

For information about customizing Probe colors in `msim.ini`, see *Appendix A* in the *Schematics User's Guide*.

The configuration file `msim.ini` contains settings which control the way Probe is run on your system. Foreground, background, and trace colors for display and hard copy can be configured for Probe by editing the file `msim.ini`.

`.prb` files, command files, and switches can be specified in the Run Probe Command text box of the Options/Editor Configuration/App Settings dialog box. The command line entered here is saved to `msim.ini`. Probe recognizes these options when you start it automatically after simulation or when you start it from Schematics by selecting Run Probe from the Analysis menu or by pressing F12.

Manual Startup

The command for running Probe at the Windows Properties command line is:

probe <*options*>* <*data file*>

where

options	are the command line options for running Probe with a command file (C switch), log file (L switch), etc., where switches are preceded with / or -.
data file	is the name of the Probe data file generated by the simulator.

Other Ways to Run Probe

Mid-analysis “snoop”

The mid-analysis “snoop” feature allows simulations to be paused and resumed so that you can use Probe to analyze waveforms before the simulation is finished. This feature is useful for verifying that a long transient analysis is proceeding correctly. You can start the analysis, check the waveforms for the first few transitions, and then either let the analysis continue or abort it.

To start mid-analysis “snoop”

1. In the Simulation Status window, select Pause Simulation from the File menu. A check mark is displayed.
2. Select Run Probe from the File menu.

To resume a simulation

1. Exit Probe.
2. In the Simulation Status window, select Pause Simulation from the File menu.

Starting Probe during a simulation

Once a simulation is in progress, it is possible to monitor the results for the data section currently being written by the simulator. This method applies when Monitor Waveforms (Auto-update) is not selected in the Probe Setup dialog box in Schematics.

To start Probe during a simulation

1. Start the simulation as described in Starting Simulation on page 7-11.
2. In Schematics, start Probe by selecting Run Probe from the Analysis menu. The current data file is automatically opened.
3. (In Probe) If multiple data sections are available, load the data section currently being generated by selecting the last data section listed in the Available Sections dialog box. This deselects all earlier data sections.
4. (In Probe) Select the waveforms to be monitored using Add on the Trace menu.

When started this way, the Probe window monitors the waveforms for as long as the data section is being written. Once complete, the window reverts to manual mode.

To monitor subsequently generated data sections

1. Select AC, DC, or Transient from the Plot menu.
2. In the Available Sections dialog box, select the latest data section that is being generated. This deselects all earlier data sections.

   If a new Probe window is opened (using New on the Window menu) while monitoring the data, the new window starts in manual mode.

Interacting with Probe while in monitor mode

All of the Probe functionality is available during monitor mode. However, functions that change the X axis domain (set a new X axis variable) pause monitoring and place the window in manual mode until the X axis is reverted back to its original domain. These functions include:

- fast Fourier transforms, activated by selecting X Axis Settings from the Plot menu and clicking on the Fourier check box
- Performance Analysis, activated by selecting X Axis Settings from the Plot menu and clicking on the Performance Analysis check box
- new X axis variable, activated by selecting X Axis Settings from the Plot menu, clicking on Axis Variable, specifying a new X axis variable
- goal function evaluation, activated by selecting Eval Goal Function from the Trace menu
- loading a previously completed data section

Configuring update intervals

The frequency at which Probe updates the waveform display can be customized by selecting Options from the Tools menu and specifying the auto-update interval. The default setting (Auto) leaves the interval determination to Probe. You can also define the interval at fixed time intervals (Every *n* sec) or according to the percentage of simulation completed (Every *n* %), where *n* is user-defined.

Schematic Markers

Markers can be placed on your schematic to identify the points where you want to see the waveform results displayed in Probe. Placement can occur before or after simulation takes place.

When placed before simulation, they can be used to limit results written to the Probe data file and/or automatically display those traces in Probe. Once Probe is running, placement of any additional markers on the schematic results in the automatic display of traces in the currently active plot window.

Table 13-1 summarizes the available marker types and how to select them for placement. The markers for dB, phase, group delay, real, and imaginary waveform characteristics can be used instead of the built-in Probe functions provided in output variable expressions (see Table 13-6 on page 13-25).

Table 13-1 *Schematic Markers*

Waveform	Markers Menu Selection	Symbol Selection
voltage	Mark Voltage/Level	(not required)
voltage differential	Mark Voltage Differential	(not required)
current	Mark Current Into Pin	(not required)
dB	Mark Advanced	VDB (voltage) IDB (current)
phase	Mark Advanced	VPHASE (voltage) IPHASE (current)
group delay	Mark Advanced	VGROUPDELAY (voltage) IGROUPDELAY (current)
real	Mark Advanced	VREAL (voltage) IREAL (current)
imaginary	Mark Advanced	VIMAGINARY (voltage) IIMAGINARY (voltage)

The Markers menu provides additional selections for controlling display of marked results in Probe, both after simulation and after initial marker placement:

Show All	causes all traces corresponding to markers placed on any page or level of the schematic to be displayed in Probe
Show Selected	selectively displays existing markers (highlight markers of interest and select Show Selected)
Clear All	removes all markers from the schematic and all corresponding traces in the Probe display

Probe Example

In this section, basic techniques for operating Probe are demonstrated using the circuit `example.sch`.

The example circuit `example.sch` is provided with the MicroSim software installation.

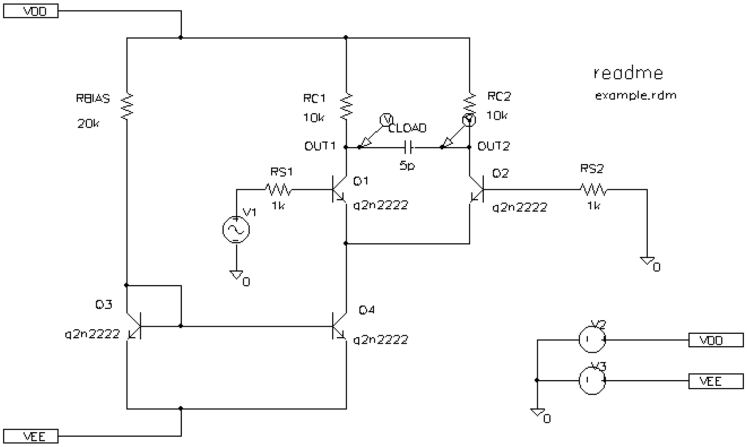


Figure 13-2 *Example Schematic example.sch*

When shipped, `example.sch` is set up with multiple analyses enabled. For purposes of this demonstration, we can *disable* the AC sweep, DC sweep, Monte Carlo/worst-case, and small-signal transfer function analyses by selecting Setup from the Analysis menu in Schematics and toggling the check box next to the appropriate analysis. The specification for each of these disabled analyses remains intact. We can run them in the future by simply toggling the check box back to the enabled state.

We will run with the Bias Point Detail, Temperature, and Transient analyses enabled. The temperature analysis is set to 35 degrees. The transient analysis is specified with:

Print Step	20ns
Final Time	1000ns
Enable Fourier	checked on
Center Frequency	5Meg
Output Vars	V(OUT2)

The simulation is started by selecting Simulate from the Analysis menu in Schematics. By default, Probe is set to automatically run. In this example, we will not be specifying goal functions or macros so that we can keep the default

Automatically Run Probe After Simulation option enabled in the Probe Setup dialog.

PSpice generates a binary Probe data file containing the results of the simulation. The Probe screen is displayed with the data file `example.dat` already loaded as shown in Figure 13-3.

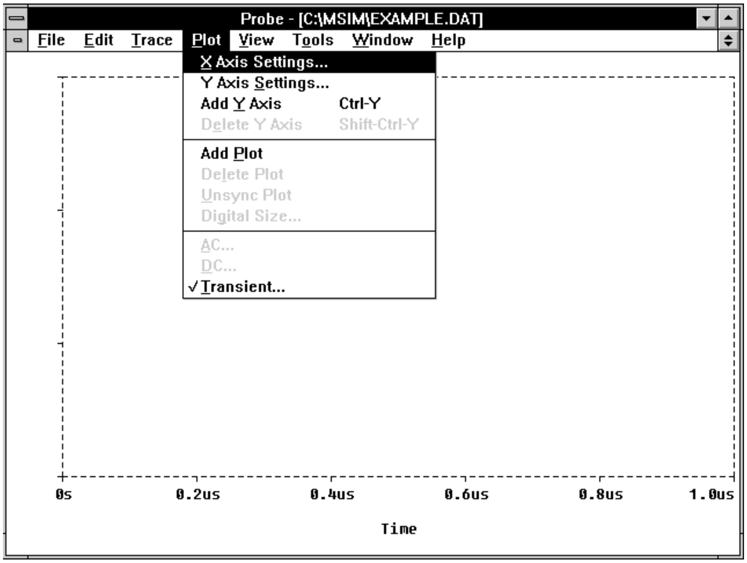


Figure 13-3 *Probe Main Window with Loaded example.dat and Open Plot Menu*

The name of the data file `example.dat` is indicated in the title bar. All Probe commands are activated through the menu items.

Notice that the Transient menu item on the Plot menu is checked, indicating that the data currently loaded are the transient analysis results. A menu does not always allow selection of all of its commands, as indicated by the grayed-out entries. The availability of commands depends on what activities are currently legal.

Having selected an analysis, voltages and currents across devices or at nets can be displayed in the Probe plot using either the schematic marker method or by explicitly specifying Probe output variables (as will be demonstrated in this example).

Probe output variables are specified in Probe by selecting Add from the Trace menu. A list of valid output variables is automatically displayed when Add is selected. A screen similar to Figure 13-4 is displayed.

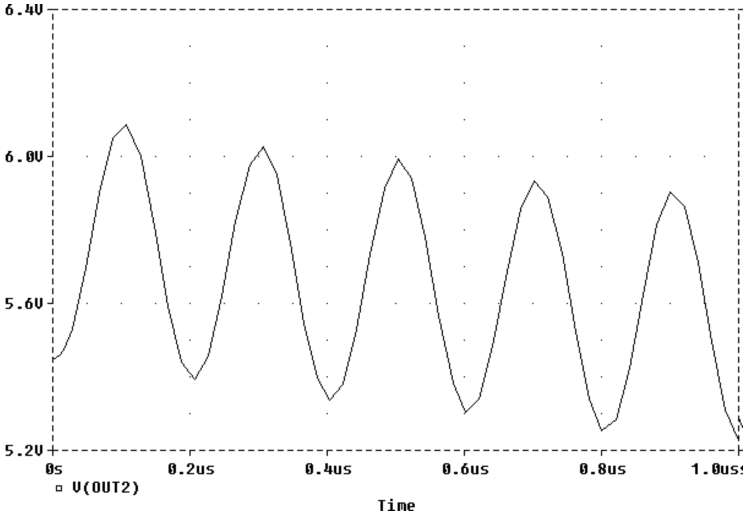


Figure 13-4 *Output from Transient Analysis: Voltage at*

User-Interface Features

Probe offers a number of direct manipulation techniques to facilitate analysis of the waveform data as well as shortcuts. These techniques are described in this section.

Zoom Regions

Probe provides a direct manipulation method for marking the zoom region of the plot. After placing the first bar, and with the mouse button still depressed, the second bar can be dragged left or right to demarcate the region of zoom. By subsequently selecting Area from the View menu, Probe changes the plot to display only the region of interest as marked by the zoom bars.

It is also possible to directly select a zoom region by dragging the mouse in the plot to produce a zoom box as shown in Figure 13-5. As with the case above, Area on the View menu redisplays the traces, showing only the region within the zoom box.

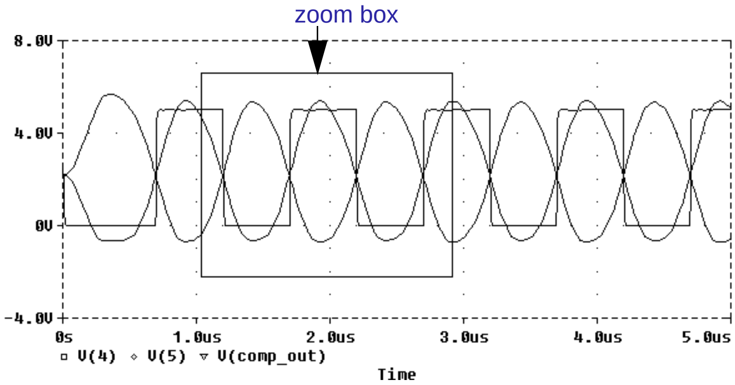


Figure 13-5 *Zoom Box*

Accelerator keys

Many of the menu functions in Probe have matching keystrokes which can be used instead. For instance, having placed a zoom box in the plot, we could have typed Ctrl + A instead of selecting Area from the View menu.

Scrolling Traces

By default, when a plot is zoomed, standard scroll bars will appear to the left and/or bottom of the plot area. These can be used to pan through the data. Scroll bar behavior can also be configured by selecting Options from the Tools menu and checking the appropriate Scrollbars box as follows:

Auto	Scroll bars appear when a plot is zoomed or additional traces are displayed in the plot but are not visible (default).
Never	Scroll bars are never displayed. This mode provides maximum plot size and is useful on VGA and lesser resolution displays.
Always	Scroll bars are displayed at all times, but are disabled if the applicable axis is full scale.

Modifying Traces and Labels

Traces, text labels, and ellipse labels that are currently displayed within the plot window may be modified, thus eliminating the need to delete and recreate any of these objects. The relevant object must be selected.

You can also double-click on a trace or text label to modify it.

For instance, to modify the expression for any given trace, the trace name must be selected by clicking on it (indicated by a color change) and then selecting Modify Object from the Edit menu. Text and ellipse labels may be similarly selected and modified. In the case of ellipse labels, the inclination angle may be modified.

Moving Traces and Labels

Traces and labels can also be selected and moved, either within the same plot window or to another plot window.

To copy or move traces

1. Select one (single click) or more (Shift+click) traces by trace name. Selected trace names are highlighted.
2. Select Copy or Cut from the Edit menu, depending upon whether you wish to leave the selected traces intact, or to remove them. In either case, the traces are saved to the clipboard.
3. From the plot window where traces are to be added, do one of the following:
   - To add traces to the end of the currently displayed set, select Paste from the Edit menu.
   - To add traces before a currently displayed trace, select the trace and then select Paste from the Edit menu.

Here are some considerations when moving traces between windows:

- If the new plot window is reading the same Probe data file, the copied or moved traces are identical to the original selection set.
- If the new plot window is reading a *different* Probe data file, the copied traces are displayed in the context of the waveform expression as applied to the new data.

  For example, suppose two data files, `mysim.dat` and `yoursim.dat` each contain a V(2) waveform. Suppose also that two plot 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 since it is derived from `yoursim.dat` instead of `mysim.dat`.

To copy or move labels

1. Select one (single click) or more (Shift+click) labels by label name, or select multiple labels by clicking and dragging to form a Selection Box around the label objects of interest. Selected label names are highlighted.
2. Select Copy or Cut from the Edit menu, depending upon whether you wish to leave the selected labels intact, or to remove them. In either case, the labels are saved to the clipboard.
3. From the plot window where labels are to be added, select Paste from the Edit menu. Labels can be moved by dragging them to a new location.

Tabulating Trace Data Values

It can be useful to generate a table of data points reflecting one or more traces in the plot window, and use this information in a document or spreadsheet. To view the table:

1. Select one (single click) or more (Shift+click) traces by trace name. Selected trace names are highlighted.
2. Save the trace data point values to the Clipboard by selecting Copy or Cut from the Edit menu, depending upon whether you wish to leave the selected traces intact.
3. From within Clipboard, select either Text or OEM Text from the Display menu.

To store the data points as an ASCII file, paste the data from the Clipboard into a text editor of your choice. This method guards against spurious clipboard file data appearing at the beginning of the file when saving the data directly from the Clipboard Viewer program.

Cursors

When one or more traces are displayed, Probe provides cursors which can be used 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.

Cursors are enabled by selecting Cursor from the Tools menu and then selecting Display. Both cursors are initially placed on the trace listed first in the trace legend. The corresponding trace symbol is outlined in a dashed line. On the Cursor submenu under the Tools menu, there are specific cursor placement commands: Peak, Trough, Slope, Min, Max, Point, and Search Commands.

The mouse and keyboard can also be used to move cursors to a chosen position on the trace. **Table 13-2** lists the cursor control commands for the mouse. Keyboard commands are listed in **Table 13-3**.

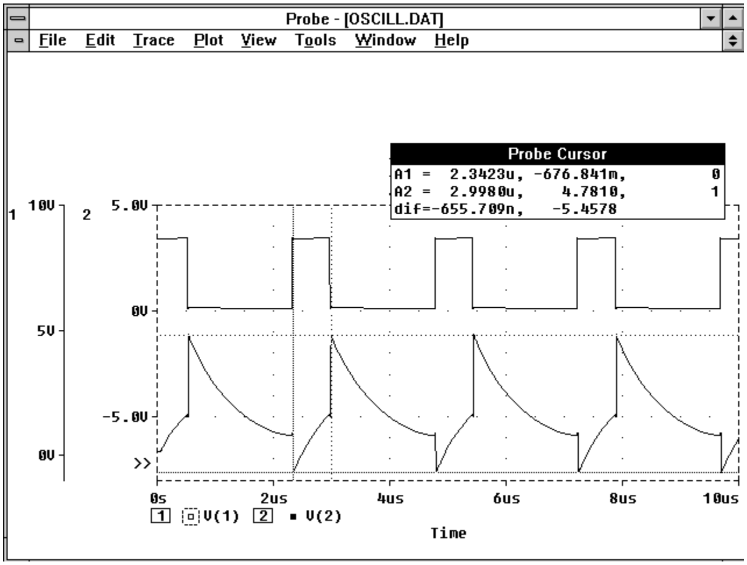
Table 13-2 *Mouse Commands for Cursor Control*

Category	Command	Function
cursor assignment	left-click on the trace symbol	associates the first cursor with the selected trace.
	right-click on the trace symbol	associates the second cursor with the selected trace.
cursor movement	left-click in the display area	moves the first cursor to the closest trace segment at the X position.
	right-click in the display area	moves the second cursor to the trace segment at the X position.

Table 13-3 *Keyboard Commands for Cursor Control*

Keyboard Sequence	**Function**
Ctrl+← and Ctrl+→	Changes the trace associated with the first cursor.
Shift+Ctrl+← and Shift+Ctrl+→	Changes the trace associated with the second cursor.
← and →	Moves the first cursor along the trace.
Shift+← and Shift+→	Moves the second cursor along the trace.
Home	Moves the first cursor to the beginning of the trace.
Shift+Home	Moves the second cursor to the beginning of the trace.
End	Moves the first cursor to the end of the trace.
Shift+End	Moves the second cursor to the end of the trace.

Figure 13-6 shows both cursors positioned on the V(1) waveform.



```
A1 =   2.3423u, -676.841m,          0
A2 =   2.9980u,    4.7810,          1
dif=-655.709n,   -5.4578
```

Figure 13-6 *Probe Screen with Cursors Positioned on the Trough and Peak of the V(1) Waveform*

Cursor 1 has been positioned on the minimum value of the V(1) waveform using the Trough option from the Cursor command in the Tools menu. Cursor 2 been positioned on the maximum value of the same waveform using Peak option. In the Probe Cursor box, cursor1 and cursor 2 coordinates are displayed (A1 and A2, respectively) with their difference shown at the bottom (dif).

The mouse buttons are also used to associate each cursor with a different trace by clicking appropriately (see **Table 13-2 on page 13-19**) on the trace symbol in the legend. These are outlined in the pattern corresponding to the associated cursor's crosshair pattern. Given the example in Figure 13-6, right-clicking on the V(2) symbol will associate cursor two with the V(2) waveform. The legend now appears as shown in Figure 13-7.

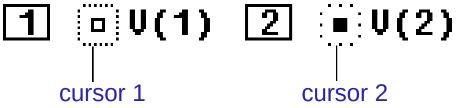


Figure 13-7 *Legend*

The Probe Cursor box also updates the A2 coordinates to reflect the X and Y values corresponding to the V(2) waveform.

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. Click on the desired curve.

Probe Trace Expressions

Traces are referred to by Probe output variable name. These can be selected from the list in the Add Traces dialog box, or typed at the Trace Command prompt. Probe output variables are similar to the PSpice output variables specified in the Schematics Analysis Setup dialog boxes for AC, DC, Monte Carlo, worst-case, transfer function, and Fourier analyses. However, there are additional alias forms that are valid within Probe. Both forms are discussed here.

Basic Output Variable Form

See **Chapter 14,**Output Options, for additional information about output options.

This form is representative of those available for specifying PSpice analyses.

<*output*>[*AC suffix*](<*name*>[,*name*])

where:

<*output*>	is the type of output quantity: V for voltage or I for current.
[*AC suffix*]*	specifies the quantity to be reported for an AC analysis, such as M (magnitude), P (phase), G (group delay).
<*name*> [,<*name*>]	specifies either the *out id* or (+ *out id*, - *out id*) pair for which the voltage is to be reported, or the *out device* for which a current is reported, where: • *out id* specifies either the *net id* or *pin id* (<*fully qualified device name*>:<*pin name*>) • *out device* specifies the fully qualified device name

Output Variable Form for Device Terminals

This form can only be specified in Probe. The primary difference between this and the basic form is that the terminal symbol appears before the *net id* or *device name* specification (whereas the basic form treats this as the *pin name* within the *pin id*).

<*output*>[*terminal*]*[*AC suffix*](<*name*>[,*name*])

where:

<*output*>	is the type of output quantity: V for voltage or I for current.
[*terminal*]*	specifies one or more terminals for devices with more than two terminals, such as D (drain), G (gate), S (source), B (base).
[*AC suffix*]*	specifies the quantity to be reported for an AC analysis, such as M (magnitude), P (phase), G (group delay).
<*name*> [,<*name*>])	specifies the *net id*, *net id* pair, or fully qualified *device name*.

Table 13-4 summarizes the allowed formats and **Table 13-5** provides examples of equivalent output variables. Note that some of the output variable formats are unique to Probe.

Table 13-4 *Probe Output Variable Formats*

Format	Meaning
V[*ac*](< + *out id* >, < - *out id* >)	Voltage between + and - *out ids*
V<*pin name*>[*ac*](< *2-terminal device* >)	Voltage at *pin name* of a *2-terminal 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 one end *z* of a *transmission line device*

Table 13-4 *Probe Output Variable Formats (continued)*

Format	**Meaning**
I<*pin name*>[*ac*](< *2-terminal device* *>)*	Current through *pin name* of a *2-terminal device*
I< *x* >[*ac*](< *3 or 4-terminal device* >)	Current through non-grounded terminal *x* of a *3 or 4-terminal device*
I< *z* >[*ac*](< *transmission line device* >)	Current through one end *z* of a *transmission line device*
< *DC sweep variable* >	Voltage or current source name
FREQUENCY	AC analysis sweep variable
TIME	Transient analysis sweep variable
V(ONOISE)	Total RMS summed noise at output net
V(INOISE)	Total RMS summed noise at input net

Table 13-5 *Examples of Probe Output Variable Formats*

A Basic Form	**An Alias Equivalent**	**Meaning**
V(NET3,NET2)	(same)	Voltage between 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
VIN	(same)	Voltage source named VIN
FREQUENCY	(same)	AC analysis sweep variable

Trace Expressions

Arithmetic expressions of output variables use the same operators as those used in PSpice (via Schematic symbol attribute definitions). Expressions may also include intrinsic functions. The intrinsic functions available in Probe similar to those available in PSpice, but with some differences, as shown in **Table 13-6**. A complete list of PSpice arithmetic functions can be found in **Table 3-6 on page 3-11**.

Table 13-6 *Probe's Arithmetic Functions*

Probe Function	Description	Available in PSpice?
ABS(x)	$\lvert x \rvert$	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 {in degrees}	YES
R(x)	real part of x	YES
IMG(x)	imaginary part of x	YES
G(x)	group delay of x {in seconds}	NO
PWR(x,y)	$\lvert x \rvert^{y}$	YES
SIN(x)	$\sin(x)$	YES
COS(x)	$\cos(x)$	YES
TAN(x)	$\tan(x)$	YES
ATAN(x)	$\tan^{-1}(x)$	YES
ARCTAN(x)		
d(x)	derivative of x with respect to the X axis variable	NO
s(x)	integral of x over the range of the X axis variable	NO
AVG(x)	running average of x over the range of the X axis variable	NO

Table 13-6 *Probe’s Arithmetic Functions (continued)*

Probe Function	Description	Available in PSpice?
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

Note *For AC analysis, Probe uses complex arithmetic to evaluate expressions. If the result of the expression is complex, then its magnitude is displayed.*

Explicit numeric values are input in the same form as the simulator (via Schematics symbol attributes), except that the suffixes M and MEG are replaced with m (milli, 1E-3) and M (mega, 1E+6), respectively. Also, MIL and mil are not supported. With the exception of the m and M scale suffixes, Probe is not case sensitive, therefore upper/lower case characters are equivalent (V(5) and v(5), for example).

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. The quantities 2e-3, 2mV, and 0.002V all have the same numerical value. For plotting purposes, Probe notes that the second and third forms are in volts, whereas the first is dimensionless.

The units which Probe recognizes are shown in Table 13-7.

Table 13-7 *Output Units Recognized by Probe*

Symbol	Unit
V	Volt
A	Amps
W	Watt

Table 13-7 *Output Units Recognized by Probe*

Symbol	Unit
d	degree (of phase)
s	second
H	Hertz

Probe also knows that W=V·A, V=W/A, and A=W/V. So, if you add a trace which is

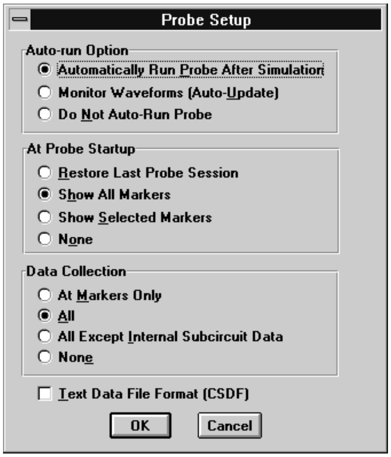
```
V(5)*ID(M13)
```

the axis values will be labeled with W”

For a demonstration of trace presentation, see Probe Example on page 13-12.

Saving Results in ASCII Format

The default Probe data file format is binary. However, the Probe data file may be saved in the Common Simulation Data Format (CSDF) instead, by selecting Probe Setup from the Analysis menu and clicking in the Text Data File Format check box. This causes simulation results to be written to the Probe data file in ASCII format following the CSDF convention.



Probe Setup

Auto-run Option
- ● Automatically Run Probe After Simulation
- ○ Monitor Waveforms (Auto-Update)
- ○ Do Not Auto-Run Probe

At Probe Startup
- ○ Restore Last Probe Session
- ● Show All Markers
- ○ Show Selected Markers
- ○ None

Data Collection
- ○ At Markers Only
- ● All
- ○ All Except Internal Subcircuit Data
- ○ None

☐ Text Data File Format (CSDF)

Limiting File Size Using Markers

By default, all simulation results are written to the Probe data file. However, if you have placed markers on your schematic prior to simulation, you can instruct the simulator to save only the results for the marked wires and pins. This is specified in the Data Collection section of the Analysis/Probe Setup dialog box by enabling At Markers Only. The default is All data. An All Except Internal Subcircuit Data selection is also available which excludes data for internal subcircuit nodes and devices.

Probe Data File Output Control

Probe data file format and content can be controlled directly from the schematic editor using options available in the Probe Setup dialog box in combination with marker placement.

Avoiding File Size Limits

When PSpice runs, it creates a Probe data file. The size of this file for transient analyses is roughly equal to (# transistors)·(# simulation steps)·24 bytes. The size for other analyses is about 2.5 times smaller. For long runs, especially transient runs, this can generate Probe data files which are over a megabyte. Even if this does not cause a problem with disk space, large Probe data files take longer to read in and longer to display traces on the screen.

One reason that Probe data files are large, is that *all net voltages and device currents are stored* for each time (or frequency, etc.) step. To avoid this, you can place markers on the schematic and specify in Schematics that data should be collected at markers only. This feature is found by selecting Probe Setup from the Analysis menu in Schematics.

The other reason for the large size of the Probe data file is that long runs, especially transient ones, involve storing many data points. For transient runs, you can suppress a part of the run by setting the No Print Delay value in the transient analysis specification. For example, setting No Print Delay to 0.9 ms tells the simulator to start output at 0.9 ms instead of time 0. Note that this does not affect the transient calculations themselves; these always start at time 0. It only suppresses the output.

Output Options

14

Chapter Overview

This chapter describes output options that can be used to control the simulation output. This chapter includes the following sections:

Viewpoints

Pseudocomponents that display simulation results on the schematic are called viewpoints. These apply to the analog portion of your circuit. Bias voltages and currents are supported.

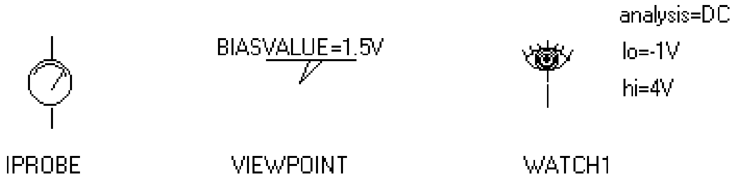


Figure 14-1 *Viewpoint Example*

IPROBE allows you to display the bias point current. This essentially places a zero-valued voltage source on the schematic. Place the IPROBE symbol on the net, and the bias point current is displayed.

VIEWPOINT allows you to display the bias point voltage of a net within the MicroSim Schematics window. Place the VIEWPOINT symbol on the net, and the bias point is displayed.

WATCH1 allows you to specify up to three output variables to be viewed in the PSpice window as DC, AC, and/or transient analyses proceed. All output variables are acceptable except group delay. Specify the type of analysis by editing the Value text box of the ANALYSIS attribute. Set the upper and lower limit values by editing the LO and HI attributes.

Printpoints

Pseudocomponents that write results to the simulation output file are called printpoints. Printpoints generate tabulated tables or line printer plots. Printpoint symbols and types are shown in Figure 14-2. Table 14-1 describes each of the printpoint symbols.

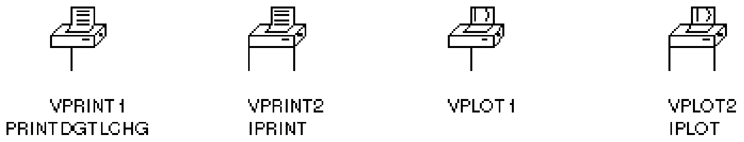


Figure 14-2 *Printpoint Symbols*

Table 14-1 *Printpoint Symbols*

Symbol	Description
IPLOT	plot showing current through a cut in the net; this symbol must be inserted into the circuit in series (like a current meter)
IPRINT	table showing current through a cut in the net; this symbol must be inserted into the circuit in series (like a current meter)
VPLOT1	plot showing voltages on the net to which the symbol terminal has been connected
VPLOT2	plot showing voltage differentials between the two nets to which the symbol terminal have been connected
VPRINT1	table showing voltages at the net to which the symbol terminal has been connected
VPRINT2	table showing voltage differentials between the two nets to which the symbol terminals have been connected

VPRINT1 supersedes the old PRINT1 pseudocomponent.

Except for the PRINTDGTLCHG symbol, you can specify one or more analysis types for which the information should be reported as follows:

1. Double-click on the symbol.

2. Double-click on the analysis type attribute of interest. Set its Value text box to any non-blank value, such as `Y`, `YES`, or `1`.

   Set any additional analysis types. If AC is set, then one or more output formats may also be set: MAG (magnitude), PHASE, REAL, IMAG (imaginary), or DB.

If no analysis types are selected, transient results is reported by default. If AC is selected but no output formats are selected, MAG (magnitude) is used.

Setting Initial State

A

Appendix Overview

This appendix includes the following sections:

not included in: Basics+

Save and Load Bias Point

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

See Setting Up Analyses on page 7-3 for a description of the Analysis Setup dialog box.

1. Click on Load Bias Point in the Analysis Setup dialog box.
2. Complete the Save Bias Point dialog box as described in the *Schematic Editor Reference* chapter of the *Schematics User's Guide*. Click on OK when finished.
3. The check box next to Save Bias Point should be checked on. If not, click on the check box to enable it.

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 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

See Setting Up Analyses on page 7-3 for a description of the Analysis Setup dialog box.

1. Run a simulation using Save Bias Point for the Analysis Setup dialog box.
2. Prior to running another simulation, click on Load Bias Point.
3. Specify a bias point file to load. Include the path if the file is not located in your working directory. Click on OK when finished.
4. The check box next to Load Bias Point should be checked on. If not, click on the check box to enable it.

Setpoints

Pseudocomponents that specify initial conditions are called setpoints. These apply to the analog portion of your circuit.

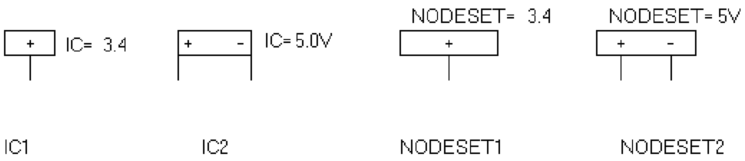


Figure A-1 *Setpoints*

The example is Figure A-1 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 the circuit 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 attribute to the desired initial condition. PSpice 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 pseudocomponents, 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 Value text box for the VALUE attribute. PSpice attaches a voltage source with a 0.0002 ohm series resistance to each net to which an IC symbol is connected.

These pseudocomponents are netlisted as PSpice .IC and .NODESET commands. Refer to these commands in the *PSpice Reference Manual* for more information. Setpoints can be created for inductor currents and capacitor voltages using the IC attribute described in Setting Initial Conditions on page A-6.

Setting Initial Conditions

The IC attribute allows initial conditions to be set on capacitors and inductors. These conditions are applied during all bias point calculations. However, if you enable 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 attribute 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.*

See Setpoints on page A-4 for more information about IC1 and IC2.

Applying an IC attribute for a capacitor has the same effect as applying one of the pseudocomponents IC1 or IC2 across its nodes. PSpice attaches a voltage source with a 0.002 ohm series resistance in parallel with the capacitor. The IC attribute allows the user to associate the initial condition with a device, while the IC1 and IC2 pseudocomponents allow the association to be with a node or node pair.

In the case of initial currents through inductors, the association is only with a device, and so there are no corresponding pseudocomponents. The internal implementation is analogous to the capacitor. PSpice attaches a current source with a 1 Gohm parallel resistance in series with the inductor.

If you want IC attributes to be ignored when Skip Initial Transient Solution is not enabled in the Transient Analysis Setup dialog box:

See Setting Up Analyses on page 7-3 for a description of the Analysis Setup dialog box.

1. Click on Options for the Setup Analysis dialog box.
2. Set NOICTRANSLATE to Y.

Index