MicroSim PSpice A/D

Circuit Analysis Software

Reference Manual

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

Chapter 2 Analog Devices

Chapter 3 Digital Devices

Chapter 4 Parts

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 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.

Overview

The MicroSim family of products is fully intergrated, giving you the flexibility to work through your circuit design in a consistent environment.

This manual contains the reference material needed when working special circuit analysis in PSpice A/D.

Included in this manual are detailed command descriptions, start-up option definitions, and a list of supported devices in the digital and analog device libraries.

How to Use this Manual

This manual has comprehensive reference material for all of the MicroSim Circuit Analysis programs, which include:

- PSpice A/D
- PSpice A/D Basics+
- PSpice
- PSpice Basics

This manual assumes that you are familiar with MicroSoft Windows (3.1, 3.11, 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*.

Note *For UNIX users: All screen captures in this manual are of Windows dialog boxes and windows. Most options in these dialog boxes and windows are available in your operating environment. When certain options are not available to you, or you must do something differently than what is primarily outlined, information specific to your platform is provided.*

Manual Conventions

Before using your circuit analysis software, it is important to understand the conventions used in this documentation.

Typographical Conventions

This manual 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
"Quoted Text"	"mydiodes.slb"	Library files and file names.
<KeyName>	Press <Enter> ...	A specific key or key stroke on the keyboard.
`monospace font`	Type `VAC`...	Output produced by a printer and commands/text entered from the keyboard.

Command Syntax Formats

The following table provides the command syntax formats.

Notation	Examples	Description
italic text	*model name* *filename.ext*	Specific items or values (file names and parameters) that the user must supply in the command line.
UPPERCASE	AC	Literal key strokes — user must enter keypad symbols, numerals, and alphabetic characters as shown; alphabetic characters *are not* case sensitive.
< >	<*model name*>	A required item in a command line (for example <*model name*> in a command line means that the model name parameter is required).
< >*	<*value*>*	The asterisk indicates that the item shown in italics must occur one *or more* times in the command line.
[ ]	[AC]	Optional item.
[ ]*	[*value*]*	The asterisk indicates that there is zero or more occurrences of the specified subject.
< | >	<YES | NO>	Specify one of the given choices.
[|]	[ON | OFF]	Specify zero or one of the given choices.

Numeric Value Conventions

The numeric value and expression conventions in the following table not only apply to the PSpice commands described in Chapter One, but also to the device declarations and interactive numeric entries described in subsequent chapters.

Literal numeric values are written in standard floating point notation. PSpice applies the default units for the numbers describing the component values and electrical quantities. However, these values can be scaled by following the number using the appropriate scale suffix as shown in the following table.

Scale	Symbol	Name
10^{-15}	F	femto-
10^{-12}	P	pico-
10^{-9}	N	nano-
10^{-6}	U	micro-
$25.4*10^{-6}$	MIL	--
10^{-3}	M	milli-
	C	Clock cycle*
10^{+3}	K	kilo-
10^{+6}	MEG	mega-
10^{+9}	G	giga-
10^{+12}	T	tera-

* Clock cycle varies and must be set where applicable.

Numeric Expression Conventions

Numeric values can also be indirectly represented by parameters (see the .PARAM command in Chapter One). Numeric values and parameters can be used together to form arithmetic expressions. PSpice expressions can incorporate the intrinsic functions shown in the following table.

The Function column lists expressions that PSpice and PSpice A/D recognize. The Meaning column lists the mathematical definition of the function. There are some differences between these functions available in PSpice and those available in Probe. Refer to Probe Help for more information.

Function*	Meaning	Comments
ABS(x)	|x|	
ACOS(x)	arccosine of x	-1.0 <= x <= +1.0
ARCTAN(x)	$\tan^{-1}(x)$	result in radians
ASIN(x)	arcsine of x	-1.0 <= x <= +1.0
ATAN(x)	$\tan^{-1}(x)$	result in radians
ATAN2(y,x)	arctan of (y/x)	result in radians
COS(x)	cos(x)	x in radians
COSH(x)	hyperbolic cosine of x	x in radians
DDT(x)	time derivative of x	applicable for transient analysis only
EXP(x)	e^x	
IF(t, x, y)	x if t=TRUE y if t=FALSE	t is a Boolean expression that evaluates to TRUE or FALSE and can include logical and relational operators (see *Parts Command Line Options* on page xli). X and Y are either numeric values or expressions. For example, {IF (v(1)<THL, v(1), v(1)*v(1)/THL)} Care should be taken in modeling the discontinuity between the IF and ELSE parts, or convergence problems can result.
IMG(x)	imaginary part of x	returns 0.0 for real numbers
LIMIT(x,min,max)		result is min if x < min, max if x > max, and x otherwise
LOG(x)	ln(x)	log base e

Function*	Meaning	Comments
LOG10(x)	log(x)	log base 10
M(x)	magnitude of x	this produces the same result as ABS(x)
MAX(x,y)	maximum of x and y	
MIN(x,y)	minimum of x and y	
P(x)	phase of x	returns 0.0 for real numbers
PWR(x,y)	$\|x\|^y$ or, {x**y}	the binary operator ** is interchangeable with PWR(x,y)
PWRS(x,y)	$+\|x\|^y$ (if x>0), $-\|x\|^y$ (if x<0)	
R(x)	real part of x	
SDT(x)	time integral of x	applicable for transient analysis only
SGN(x)	signum function	
SIN(x)	sin(x)	x in radians
SINH(x)	hyperbolic sine of x	x in radians
STP(x)	1 if x>=0.0 0 if x<=0.0	The unit step function can be used to suppress a value until a given amount of time has passed. For instance, {v(1)*STP(10ns-TIME)} gives a value of 0.0 until 10ns has elapsed and then gives v(1).
SQRT(x)	$x^{1/2}$	
TAN(x)	tan(x)	x in radians
TANH(x)	hyperbolic tangent of x	x in radians
TABLE ($x,x_1,y_1,x_2,y_{2,...}x_n,y_n$)		Result is the y value corresponding to x, when all of the x_n,y_n points are plotted and connected by straight lines. If x is greater than the max x_n, then the value is the y_n associated with the largest x_n. If x is less than the smallest x_n, then the value is the y_n associated with the smallest x_n.

* Most numeric specifications in PSpice allow for arithmetic expressions. Some exceptions do exist and are summarized in your *user's guide*. There are also some differences between the intrinsic functions available in PSpice and those available in Probe. Refer to your user's guide for more information on Probe.

Expressions can contain the standard operators as shown in the following table. These mathematical operators can be used in all MicroSim simulators.

Operators	Meaning
Arithmetic	
+	addition (or string concatenation)
-	subtraction
*	multiplication
/	division
**	exponentiation
Logical	
~	unary NOT
|	boolean OR
^	boolean XOR
&	boolean AND
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

Related Documentation

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

Table 0-1

Manual Name*	Description
MicroSim Schematics User's Guide	Provides information on 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. It provides examples for demonstrating the process of specifying simulation parameters, analyzing simulation data results, editing device stimuli, and creating models.
MicroSim PSpice & Basics User's Guide	Provides information on MicroSim PSpice & and PSpice Basics which are circuit analysis programs that allow you to create, simulate and test circuit designs containing analog components.
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 devices.
MicroSim Filter Designer User's Guide	Provides information about designing electronic frequency selective filters.
Library Reference Manual**	Provides a complete list of all 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*.

Command Line Options for MicroSim Application

Command Files

A command file is an ASCII text file which contains a list of commands to be executed. A command file can be specified in multiple ways:

- At the command line when starting Probe, StmEd, or Parts,
- By selecting Run Commands from the File menu and entering a command file name (for Windows Probe and StmEd only), or
- At the PROBECMD or STMEDCMD command line, found in the configuration file “msim.ini” (for Windows Probe and StmEd only).

The command file is read by the program and all of the commands contained within the file are executed. When the end of the command file is reached, commands are taken from the keyboard and the mouse. If no command file is specified, all of the commands are received from the keyboard and mouse.

The ability to “record” a set of commands can be useful when executing Probe, Parts, and StmEd. This is especially useful in Probe, when you are repeatedly doing the same simulation and looking at the same waveform with only slight changes to the circuit before each run. It can also be used to automatically create hard copy output at the end of very long (e.g., overnight) simulation runs.

Creating and Editing Command Files

You can create your own command file using a text editor, or in Probe and StmEd, you can use the File/Log Commands menu item (see “Log Files” on page xxxi for an example), to record a list of transactions in a “log” file, then use File/Run Commands to start running the logged file.

If you choose to create a command file using a text editor, note that the commands in the command file are the same as those available from the keyboard with these differences:

- The name of the command or its first capitalized letter can be used.

- Any line that begins with an "*" is a comment.
- Blank lines are ignored, therefore, they can be added to improve the readability of the command file.
- The commands "@CR", "@UP", "@DWN", "@LEFT", "@RIGHT", and "@ESC" are used to represent the <Enter>, <↑>, <↓>, <←>, <→>, and <Esc> keys, respectively.
- The command "Pause" causes Probe, Parts, or StmEd to wait until any key on the keyboard is pressed. In the case of Probe, this can be useful to examine a waveform before the command file draws the next one.

The commands are one to a line in the file, but comment and blank lines can be used to make the file easier to read.

Assuming that a Probe data file has been created by simulating the circuit "example.sch," you can manually create a command file (using a text editor) called "example.cmd" which contains the commands listed below. This set of commands draws a waveform, allows you to look at it, and then exits Probe.

```
* Display trace v(out2) and wait
Trace Add
v(out2)
Pause
* Exit Probe environment
File Exit
```

See "Simulation Command Line Specification Format" on page xxxiv and "Simulation Command Line Options" on page xxxiv for specifying command files on the simulation command line. See "Probe Command Line Specification Format" on page xxxviii and "Probe Command Line Options" on page xxxviii for details on specifying the /C or -c option for Probe.

Note *Once you activate cursors via the Tools/Cursor command, any mouse or keyboard movements that you make for moving the cursor will not be recorded in the command file. You can use the Tools/Cursor commands (i.e., Peak, Trough, etc.), and they will be recorded in your command file, but when replaying the command file, they may not work the same since cursor movement is not recorded. For example, the direction of the cursor movement is not recorded if you use the mouse or arrow keys.*

Note *The Search Commands feature is a Cursor option that allows you to position the cursor at a particular point. You can learn more about Search Commands by consulting Probe Help.*

Log Files

Instead of creating command files “by hand,” using a text editor, they can be automatically generated by creating a “log” file while running Probe, Parts, or StmEd. While executing the particular package, all of the commands given are saved in the log file. The format of the “log” file is correct for use as a command file.

You can automatically create a “.log” file in Windows Probe or StmEd by selecting File/Log Commands and entering a log file name. This will turn logging “on.” Any action taken after starting Log Commands is logged in the named file and can be run in another session by using the File/Run Commands option.

You can also create a log file for Probe, StmEd, or Parts by using the /l or -l option at the command line. For example:

```
PROBE /L EXAMPLE.LOG(PC)
probe -l example.log(Sun or HP)
```

Of course, you can use a name for the log file that is more recognizable, such as “acplots.cmd” (to Probe, Parts, and StmEd, the file name is any valid file name for your computer).

Note *Sun and HP users must use the dash (-) separators, and file names are case sensitive. PC users can use either separator (/) or (-), and file names can be in upper or lower case.*

Editing Log Files

After Probe, Parts, or StmEd is finished, the log file is available for editing to customize it for use as a command file. You can edit the following items:

- Add blank lines and comments to improve readability (perhaps a title and short discussion of what the file does).
- Add the “Pause” command for viewing waveforms before proceeding.

- Remove the “Exit” command from the end of the file, so that Probe, Parts, and StmEd do not automatically exit when the end of the command file is reached.

You can add or delete other commands from the file or even change the file name to be more recognizable. It is possible to “build” onto log files, either by using your text editor to combine files or by running Probe, Parts, and StmEd with both a command and log file:

```
PROBE /C IN.CMD /L OUT.LOG(PC)
probe -c in.cmd -l out.log(Sun & HP)
```

The file “in.cmd” gives the command to Probe, and Probe saves the (same) commands into the “out.log” file. When “in.cmd” runs out of commands, and Probe is taking commands from the keyboard, these commands also go into the “out.log” file.

Example: Log and Run Commands in Probe

Following is a simple example (using the data file “example.dat”), of how logging can be used in Probe to record and save user actions to a command file. Command files are useful if you need to remember the steps taken in order to display a set of waveforms for any given data file.

1. Start Probe.
2. Select File/Log Commands.
3. Enter “2traces” in the Log file name field, and click OK. When a check mark is placed in the box next to File/Log Command, this indicates that logging is turned on, and stays on until the box is no longer checked.
4. Select File/Open.
5. Select “example.dat” (from the examples directory) and click OK.
6. Select Trace/Add, click on the trace names V(OUT1) and V(OUT2), and click OK.
7. At this point, turn logging *off* by selecting File/Log Commands. This removes the check mark next to the command.

   Now that logging is turned off, any command issued is no longer logged in the command file. Probe accepts commands from the mouse and keyboard but does not record them.

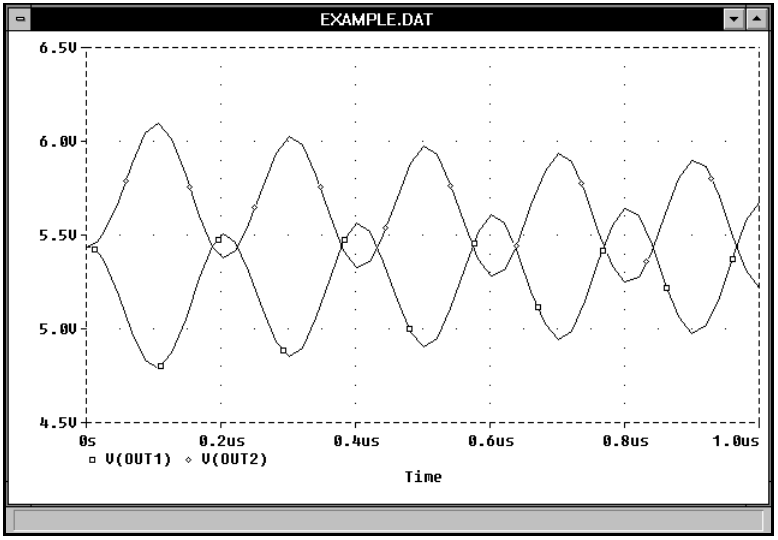
   You can view the command file from any text editor. Command files can be edited or appended to depending on the types of commands you want to store for future use. The file “2traces.cmd” should look as shown

below (with the exception of a different file path).

```
*Command file created by Probe - Wed Apr 17 10:33:55
File Open
/msim/probe/example.dat
OK
Trace Add
V(OUT1) V(OUT2)
OK
```

Now run the command file. The Probe window is still displayed and idle since we did not exit the program. To run the command file that was created:

1. Select File/Run Command.
2. Select “2traces.cmd” and click OK. The two traces should appear as before, and as shown in below.



Simulation Command Line Specification Format

The format for specifying command line options for PSpice and PSpice A/D are as follows.

PSpice A/D and PSpice

```
pspice.exe [option]*[input file][output file][data file]
```

input file — Specifies the name of a circuit file for PSpice or PSpice A/D to simulate after it starts. The *input file* name can include wildcard characters '*' or '?' in which case all file names matching the specification are simulated.

output file — Specifies the name of the output file. By default, the *output file* name has a ".out" extension. Also the name of the *output file* name defaults to the name of the *input file* with a ".out" extension.

data file — Specifies the name of the Probe data file. By default, the name of the Probe *data file* defaults to the name of the *input file* with a ".dat" extension.

option — One or more of the options listed in "Probe Command Line Options" on page xxxviii.

Simulation Command Line Options

The simulation command line can contain options and file specifications. These items can be specified in a few different ways.

From Schematics

When activated, Schematics uses initialization settings in the "msim.ini" initialization file. You can modify the way the simulator is activated from within Schematics, through the Options/Editor Configuration/App Settings dialog box. Command options entered through this dialog box take effect when the simulator is started through the Schematics Analysis/Simulate command. To specify a simulator option via Schematics, select Options/Editor Configuration/App Settings.

Type the name of the simulator executable and any / or - options in the Command field of the Simulate Command section of the App Settings dialog. See “Simulation Command Line Options” on page xxxiv for the available simulation options.

If you want to use a configuration file other than “msim.ini” for MicroSim programs started via Schematics, select Options/Editor Configuration/App Settings.

Click on the Other button in the Configuration File section of the App Settings dialog box, and specify the name and path of the initialization file you want Schematics to use. (Remember to use the -i option.)

From the “msim.ini” configuration file

You can customize your initialization file to include command line options by manually editing the PSPICECMD line in “msim.ini.” These options apply when the simulator is started from within Schematics using the default “msim.ini.”

From the Windows Program Manager

Select the PSpice or PSpice A/D icon, then select File/Properties from the Program Manager main menu. Enter the command line specification and options. These options apply when the simulation is started by double-clicking on the PSpice or PSpice A/D icon.

From the Unix Command Line (Sun and HP)

Type the name of the simulator executable and any - (dash) options on the command line.

Examples of using command line options are:

```
pspice.exe -wPause=5
pspice.exe -wOUT=new
```

Or, from the configuration file “msim.ini”

```
PSPICECMD=pspice -wtxt=doc
pspicecmd=pspice /bf=2
```

For more information about simulation command line options see the following table.

Option*	Description
-bf<*flush interval*>	Determines how often (in minutes) the simulator will "flush" the buffers of the Probe data file to disk. This is useful when a long simulation is left running and the machine crashes or is rebooted. In this case, the data file will be readable up to the last flush. The default is to flush every 10 minutes. The <*flush interval*> can be set between 0 and 1440 minutes. A value of zero means to never flush.
-bn<*number of buffers*>	Determines the number of buffers to potentially allocate for the Probe data file. Zero buffers means to do all writing directly to disk. Allocating a large number of buffers can speed up a large simulation, but will increase memory requirements. Exceeding physical memory will either slow down the simulation, or will make it fail. The default number of buffers is 4, or 1 buffer, if you are using the CSDF option.
-bs<*buffer size factor*>	Determines the size of the individual buffers for writing the Probe data file. Using a larger buffer size can reduce execution time, but at the expense of increasing the memory requirements. The values for the buffer files work as follows: option:-bs0 -bs1 -bs2 -bs3 -bs4 -bs5 -bs6 value: 256 512 1024 2048 4096 8192 16384 The default is 4K (8K if you are using CSDF).
@ <*command file*> (PC) -c <*command file*> (Unix)	Specifies the name of the command file to be run.
-i <*ini file name*>	Specifies the name of an alternate initialization file. If not specified, the simulator will use: `\windows\msim.ini`
-wOUT=<*suffix*>	Specifies the file suffix for the simulation output file. If <*suffix*> is not specified, the default ".out" will be used.
-wDAT=<*suffix*>	Specifies the file suffix for the Probe data file. If <*suffix*> is not specified, the default ".dat" will be used.
-wTEMP=<*path*>	Allows you to specify a directory to where PSpice can write temporary files.

Option*	Description
-wTXT=<*suffix*>	Specifies the file suffix for the Probe CSDF file. If <*suffix*> is not specified, the default ".txt" will be used.
-wNO_NOTIFY	Indicates that the simulator should not display the status message dialog after completion of each circuit file.
-wPAUSE=<*seconds*>	Specifies the maximum time that the status dialog box should be displayed. If <*seconds*> elapses before you click on one of the buttons, the dialog will close.
-wONLY	Causes the simulation to exit after all specified files have been simulated.
-d0 (Sun & HP only)	Suppresses simulation display. This option should be used on systems running the simulator as a batch job. If you wish to run the simulator from a remote node using "rlogin," or from an ASCII "dumb" terminal connected to one of the serial ports, you must specify -d0 or be running OpenWindows.
-iconic (Sun & HP only)	Starts iconic.

* On the PC, options can be entered using the dash (-) or slash (/) separator. Sun and HP users must use the dash (-) separator and file names are case sensitive.

Probe Command Line Specification Format

The format for specifying command line options for Probe is as follows.

```
probe[option]*[data file]
```

option — One or more of the options specified in *Probe Command Line Options* on page xl.

data file — Specifies the name of the Probe data file. By default the name of the Probe *data file* is the name of the *input file* with a ".dat" extension.

Probe Command Line Options

Probe options are specified as follows.

In Schematics

Run Probe Command field of the Options/Editor Configuration/App Settings dialog box. These options apply when Probe is started from Schematics.

In Windows

Probe options can be specified on the command line in the Program Manager's File/Properties dialog. These options apply when the simulation is started by double-clicking on the Probe icon. Options specified on the command line are not effective when Probe is started from Schematics.

On the Sun and HP

By typing the executable at the command line. See <Italicred>Probe Command Line Specification Format on page xli for details on specifying the executable on the command line. Note that for Unix platforms, file names are case sensitive.

On the PROBECMD line in the configuration file "msim.ini." These options apply when Probe is started from within Schematics using the default "msim.ini."

The command line options can be separated by spaces or in a continuous string, therefore:

```
PROBE -c makeplot.cmd -p newamp.prb
probe -cmakeplot.cmd-pnewamp.prb
```

are equivalent. The order of the options does not matter.

The command line options that use <*file name*> assume default extensions. These command line options can be used without specifying the extension to <*file name*>. For example:

```
probe -c makeplot -p  newamp
probe -c makeplot.cmd -p newamp.prb
```

achieve the same results. However, Probe searches first for the exact <*file name*> specified for these command line options and if that <*file name*> exists, Probe uses it. If the exact <*file name*> does not exist, Probe adds default extensions to <*file name*> and searches for those. The following default extensions are used:

```
<file name[.dat]>-Probe data file
-c<file name[.cmd]>-command file
-l<file name[.log]>-log file
-p<file name[.prb]>-macros, goal functions, and
displays file
```

Note *You can learn more about Probe macros by consulting Probe Help.*

The available Probe command line options are listed in the following table.

Table 0-2 *Probe Command Line Options*

Option*	Description
-bn <*number of buffers*>	Determines the number of buffers to potentially allocate for the Probe data file. Zero buffers means to do all reading directly from the disk. Allocating a large number of buffers can speed up a readin from a large simulation, but will increase memory requirements. Exceeding physical memory will either slow down the readin, or will make it fail. The default number of buffers is 4, or 1 buffer, if you are using CSDF.
-bs <*buffer size factor*>	Determines the size of the individual buffers which will be read from the Probe data file. Using a larger buffer size can reduce execution time, but at the expense of increasing the memory requirements. The values for the buffer files work as follows: option: -bs0 -bs1 -bs2 -bs3 -bs4 -bs5 -bs6 value: 256 512 1024 2048 4096 8192 16384 The default is 4K, or 8K, if you are using CSDF.
-c <*file name*>	Specifies the command file, which runs the session until the command file ends or Probe quits.
-i <*file name*>	Specifies the name of an alternate initialization file. If not specified, Probe will use \windows\msim.ini.
-l <*file name*>	Creates a “log” file, which saves the commands from this session. This log file can later be used as an input command file for Probe.
-p <*file name*>	Specifies a file which contains goal functions for Performance Analysis, macro definitions, and display configurations. This file is loaded after the global “.prb” file (specified in the “.ini” file by the line PRBFILE=msim.prb), and the local “.prb” file (<*file name*>.prb), have been loaded. Definitions in this file will replace definitions from the global or local “.prb” files that have already been loaded.

* On the PC, options can be entered using the dash (-) or slash (/) separator. Sun and HP users must use the dash (-) separator and file names are case sensitive.

Parts Command Line Options

The table below lists the available command line options for Parts.

Table 0-3 *Parts Command Line Options*

Option*	Description
-c <*file name*>	Specifies the “macro” command file, which runs the session until the command file ends or Parts quits.
-d <*file name*>	Specifies a particular device file, which defines the display and hard copy device types for Parts. If a device file is not specified, the default is “pspice.dev.”
-l <*file name*>	Creates a “log” file, which saves the commands from this session in a command file for later use.

* On the PC, options can be entered using the dash (-) or slash (/) separator. Sun and HP users must use the dash (-) separator and file names are case sensitive.

The command line options can be space separated or a continuous string, so that:

```
parts -d ega.dev -c makeplot.cmd
parts -cmakeplot.cmd-dega.dev
```

are equivalent. As shown in this example, the order of the options does not matter.

If you do not specify a device file, Parts looks for the device file “pspice.dev.” So:

```
parts
parts -d pspice.dev
```

are equivalent.

As you finish each device, its model or subcircuit is written to a file with that device’s name and with the extension “.mod.” For instance, if you extracted the parameters for the 1N914 diode, the 2N3306 transistor, and the OP27 opamp, at the end of the Parts session your working directory would contain the files D1N914.MOD, Q2N3306.MOD, and OP27.MOD with the corresponding models and subcircuits. Normally, you would copy the data from these result files into your library file(s), and then delete the individual model (“.mod”) files.

Stimulus Editor Command Line Options

StmEd command line options can be entered in the following format:

```
stmed [option]* [input file]
```

input file The name of a new or existing circuit file, and

option One or more of the options listed in the table that follows.

The command line options can be separated by spaces or listed as a continuous string. This means:

```
stmed -dega.dev -cmakestm.cmd newstm.cir
stmed -cmakestm.cmd-dega.dev newstm.cir
stmed -d ega.dev -c makestm.cmd newstm.cir
```

are all equivalent.

Option*	Description
-c <*command file*>	Specifies the name of the command file to be run. For the Sun and HP, -c specifies the “macro” command file which runs the session until the command file ends or StmEd quits.
-d <*file name*> (Sun and HP)	Specifies a particular device file, which defines the display and hard copy device types for StmEd. If a device file is not specified, the default is “pspice.dev.”
-i <*ini file name*>	Specifies the name of an alternate initialization file. If not specified, the simulator uses <*windows directory*>\msim.ini.
-l <*file name*>	Creates a “log” file, which saves the commands from this session. This log file can later be used as an input command file for StmEd.

*On the PC, options can be entered using the dash (-) or slash (/) separator. Sun and HP users must use the dash (-) separator and file names are case sensitive**.**

Commands

1

Overview

This chapter contains a table of naming and numerical conventions, having detailed descriptions of each PSpice “dot” command.

Command Reference

Table 1-1 lists all of the PSpice and PSpice A/D analysis "dot" commands. The "dot" command is contained in the circuit file. Schematics users can enter analysis specifications through the Analysis/ Setup dialog box.

Table 1-1 *Command Summary*

Type	Corresponding PSpice Command	Description	Page
Standard Analyses	.AC	Frequency response	1-4
	.DC	DC sweep	1-7
	.FOUR	Fourier components	1-14
	.NOISE	Noise	1-32
	.OP	Bias point	1-34
	.SENS	DC sensitivity	1-56
	.TF	Small-signal DC transfer function	1-69
	.TRAN	Transient	1-70
Simple Multi-Run Analyses	.STEP	Parametric	1-57
	.TEMP	Temperature	1-66
Statistical Analyses	.MC	Monte Carlo	1-21
	.WCASE	Sensitivity/Worst-Case	1-76
Initial Conditions	.IC	Clamp node voltage for bias point calculation	1-16
		Restored .NODESET bias point	
	.LOADBIAS	"Suggest" node voltage for bias calculation	1-20
	.NODESET	Store .NODESET bias point information	1-31
	.SAVEBIAS		1-52
Device Modeling	.END	End subcircuit definition	1-12
	.DISTRIBUTION	Model parameter tolerance distribution	1-10
	.MODEL	Modeled device definition	1-25
	.SUBCKT	Start subcircuit definition	1-63
Output Control	.PLOT	Analysis plot to output file (line printer format)	1-43
	.PRINT	Analysis table to output file	1-45
	.PROBE	Simulation results to Probe data file	1-46
	.VECTOR	Digital state output	1-72
	.WATCH	View simulation results in progress	1-74

Table 1-1 *Command Summary*

Type	Corresponding PSpice Command	Description	Page
Circuit File Processing	.END	End of circuit simulation description	1-12
	.FUNC	Expression function definition	1-15
	.INC	Include specified file	1-17
	.LIB	Reference specified library	1-18
	.PARAM	Parameter definition	1-41
Miscellaneous	.ALIASES	Starts alias definition	1-6
	.ENDALIASES	Ends alias definition	1-6
	.EXTERNAL	Identifies nets representing the outermost (or peripheral), connections to the circuit being simulated	1-13
	.OPTIONS	Sets miscellaneous simulation limits, analyses control parameters, and output characters	1-35
	.STMLIB	Specifies a file name containing .STIMULUS information	1-61
	.STIMULUS	Stimulus device definition	1-62
	.TEXT	Text expression, parameter, or file name used by digital devices	1-67

.AC (AC Analysis)

Purpose

The .AC command is used to calculate the frequency response of a circuit over a range of frequencies.

General Form

.AC <*sweep type*> <*points value*>
+ <*start frequency value*> <*end frequency value*>

Example

```
.AC LIN 101 100Hz  200Hz
.AC OCT  10  1kHz  16kHz
.AC DEC  20  1MEG 100MEG
```

<*sweep type*>

The sweep type must be either LIN, OCT, or DEC.

Parameter*	Description	Description
LIN	linear sweep	The frequency is swept linearly from the starting to the ending frequency. The <*points value*> is the total number of points in the sweep.
OCT	sweep by octaves	The frequency is swept logarithmically by octaves. The <*points value*> is the number of points per octave.
DEC	sweep by decades	The frequency is swept logarithmically by decades. The <*points value*> is the number of points per decade.

*One of the sweep types LIN, OCT, or DEC, must be specified.

<*points value*>

The points value (an integer), is the number of points in the sweep.

<*start frequency value*> <*end frequency value*>

The end frequency values *must not be less than* the start frequency value, and both must be greater than zero. The whole sweep must include at least one point.

The simulator calculates the frequency response by linearizing the circuit around the bias point. All independent voltage and current sources which have AC values are inputs to the circuit.

Note *A .PRINT, .PLOT, or .PROBE command must be used to get the results of the AC sweep analysis.*

If a group delay (“G” suffix) is specified as an output, the frequency

steps must be close enough together that the phase of that output changes smoothly from one frequency to the next. Group delay is calculated by subtracting the phases of successive outputs and dividing by the frequency increment.

During AC analysis, the only independent sources which have nonzero amplitudes, are those using AC specifications. The SIN specification does not count as it is used only during transient analysis.

AC analysis is a linear analysis. To analyze nonlinear functions, such as mixers, frequency doublers, and AGC, it is necessary to use transient analysis.

.ALIASES, .ENDALIASES (ALIASES and ENDALIASES)

Purpose

Aliases set up equivalences between node names and pin names, so that traces in the Probe display can be identified by naming a device and pin instead of a node. They are also used to equate a net name to a node name.

General Form

.ALIASES
<*device name*> <device *alias*> (<<*pin*>=<*node*>>)
_ _ (<<*net*>=<*node*>>)
.ENDALIASES

Example

```
.ALIASES
R_RBIAS   RBIAS (1=$N_0001   2=VDD )
Q_Q3       Q3 (c=$N_0001   b=$N_0001   e=VEE )
_       _ (OUT=$N_0007 )
.ENDALIASES
```

Comments

The first alias definition shown in the example allows the name “RBIAS” to be used as an alias for “R_RBIAS”, and it relates pin “1” of device “R_RBIAS” to node “$N_0001” and pin “2” to “VDD”. The last alias definition equates net name “OUT” to node name “$N_0007”.

.DC (DC Analysis)

Purpose

The .DC command performs a DC sweep analysis on the circuit.

General Form

.DC <*linear sweep type*> <*sweep variable name*>
+ <*start value*> <*end value*> <*increment value*>
+ *[nested sweep specification*]

.DC <*logarithmic sweep type*> <*sweep variable name*>
+ <*start value*> <*end value*> <*points value*>
+ [*nested sweep specification*]

.DC <*sweep variable name*> LIST <*value*>*
+[*nested sweep specification*]

Example

```
.DC VIN -.25  .25  .05
.DC LIN I2  5mA  -2mA  0.1mA
.DC VCE 0V 10V .5V  IB 0mA 1mA 50uA
.DC RES RMOD(R)  0.9  1.1  .001
.DC DEC NPN QFAST(IS) 1E-18 1E-14  5
.DC TEMP LIST 0 20 27 50 80 100
.DC PARAM Vsupply 7.5 15 .5
```

The DC sweep analysis calculates the circuit's bias point over a range of values for <*sweep variable name*>. The first form, and the first four examples, are for doing a linear sweep. The second form, and the fifth example, are for doing a logarithmic sweep. The third form, and the sixth example, are for using a list of values for the sweep variable.

Linear Sweeps

<*start value*> Can be greater or less than <*end value*>: that is, the sweep can go in either direction.

<*increment value*> The value must be greater than zero.

Logarithmic Sweeps (DEC or OCT)

<*start value*> The value must be positive and less than <end value>.

<*points value*> The value is the number of points in the sweep, and must be an integer.

Nested Sweep

For a nested sweep, a second sweep variable, sweep type, start, end, and increment values can be placed after the first sweep. In the nested sweep example, the first sweep is the “inner” loop: the entire first sweep is performed for each value of the second sweep.

The rules for the values in the second sweep are the same as for the first. The second sweep generates an entire .PRINT table or .PLOT plot for each value of the sweep. Probe allows nested sweeps to be displayed as a family of curves.

Sweep Type

The sweep can be linear, logarithmic, or a list of values. .

Parameter*	Description	Meaning
LIN	linear sweep	The sweep variable is swept linearly from the starting to the ending value. <*increment value*> is the step size.
OCT	sweep by octaves	Sweep by octaves. The sweep variable is swept logarithmically by octaves. The <*points value*> is the number of steps per octave.
DEC	sweep by decades	Sweep by decades. The sweep variable is swept logarithmically by decades. The <*points value*> is the number of steps per decade.
LIST	list of values	Use a list of values. In this case there are no start and end values. Instead, the numbers that follow the keyword LIST are the values that the sweep variable is set to.

*For [*linear sweep type*], the keyword LIN is optional, but either OCT or DEC must be specified for the <*logarithmic sweep type*>

<sweep variable name>

After the DC sweep is finished, the value associated with <*sweep variable name*> is set back to the value it had before the sweep started. The following items can be used as sweep variables in a DC sweep:

Parameter	Description	Meaning
Source	A name of an independent voltage or current source.	During the sweep, the source's voltage or current is set to the sweep value.
Model Parameter	A model type and model name followed by a model parameter name in parenthesis.	The parameter in the model is set to the sweep value. The following model parameters cannot be (usefully) swept: L and W for the MOSFET device (use LD and WD as a work around), and any temperature parameters, such as TC1 and TC2 for the resistor.
Temperature	Use the keyword TEMP for <*sweep variable name*>.	The temperature is set to the sweep value. For each value in the sweep, all the circuit components have their model parameters updated to that temperature.
Global Parameter	Use the keyword PARAM, followed by the parameter name, for <*sweep variable name*>.	During the sweep, the global parameter's value is set to the sweep value and all expressions are reevaluated.

.DISTRIBUTION (User-Defined Distribution)

Purpose

The .DISTRIBUTION command is used to define a user distribution for tolerances, and is only used with Monte Carlo and sensitivity/worst-case analyses. The curve described by a .DISTRIBUTION command controls the relative probability distribution of random numbers generated by PSpice to calculate model parameter deviations.

General Form

DISTRIBUTION <*name*> (<*deviation*> <*probability*>)*

Example

```
.DISTRIBUTION bi_modal (-1,1) (-.5,1) (-.5,0) (.5,0)
+ (.5,1) (1,1)
.DISTRIBUTION triangular (-1,0) (0,1) (1,0)
```

The distribution curve is defined by (<*deviation*> <*probability*>) pairs, or corner points, in a piecewise linear fashion. Up to 100 value pairs are allowed.

<*deviation*> The deviation must be in the range (-1,+1), which matches the range of the random number generator. No <*deviation*> can be less than the previous <*deviation*> in the list, although it can repeat the previous value.

<*probability*> This represents a relative probability, and must be positive or zero.

The updated value of a parameter is derived from a combination of a random number, the distribution, and the tolerance specified.

The steps taken are:

1. A <*temporary random number*> in the range (0, 1) is generated.
2. The area under the specified distribution is normalized.
3. The <*final random number*> is set to the point where the area under the normalized distribution equals the <*temporary random number*>.
4. This <*final random number*> is multiplied by the specified tolerance.

This method permits distributions which have different excursions in the positive and negative directions. It also allows the use of one distribution even if the tolerances of the components are different so long as the general shape of the distributions are the same.

To illustrate, assume there is a one µfd capacitor which has a variation of -50% to +25%, and another which has tolerances of -10% to +5%. Note that both capacitors' tolerances are in the same general shape, that is, both have negative excursions twice as large as the positive excursions.

```
.distribution cdistrib (-1,1) (.5, 1) (.5, 0) (1, 0)
c1 1 0 cmod 11u
c2 1 0 cmod2 1u
.model cmod1 cap (c=1 dev/cdistrib 50%)
.model cmod2 cap (c=1 dev/cdistrib 10%)
```

The steps taken are:

1. Assume that a <*temporary random value*> of .3 is generated.
2. The area under the cdistrib distribution (1.5) is normalized to 1.0.
3. The <*final random number*> is therefore -0.55 (the point where the normalized area equals .3).
4. For c1, this -0.55 is then scaled by 50% resulting in -.275, for c2, it is scaled by 10% resulting in -0.055. (Separate random numbers are generated for each parameter that has a tolerance unless a tracking number is specified.)

Comments

When using Schematics, several distributions can be defined by configuring an "include" file containing the .DISTRIBUTION command. For details on how to do this, refer to your PSpice user's guide.

If you are not using Schematics, a user-defined distribution can be specified as the default by setting the DISTRIBUTION parameter in the .OPTIONS command.

.END (End of Circuit)

Purpose

The .END command marks the end of the circuit. All the data and every other command must come before it. When the .END command is reached, PSpice does all the specified analyses on the circuit.

General Form

.END

There can be more than one circuit in an input file. Each circuit and each command are marked by a .END command. PSpice processes all the analyses for each circuit before going on to the next one.

Everything is reset at the beginning of each circuit. Having several circuits in one file gives the same results as having them in separate files and running each one separately. However, all the simulation results go into one ".out" file and one ".dat" file. This is a convenient way to arrange a set of runs for overnight operation.

Note *The last statement in an input file must be a .END command.*

Example

```
* 1st circuit in file
... circuit definition
.END
* 2nd circuit in file
... circuit definition
.END
```

.EXTERNAL (External Port)

Purpose

External ports are provided as a means of identifying and distinguishing those nets representing the outermost (or peripheral), connections to the circuit being simulated.

General Form

.EXTERNAL <*attribute*> <*node-name*>*

Example

```
.EXTERNAL  INPUT  Data1, Data2, Data3
.EXTERNAL  OUTPUT P1
.EXTERNAL  BIDIRECTIONAL BPort1 BPort2 BPort3
```

<*attribute*>

Is one of the keywords, “INPUT,” “OUTPUT,” or “BIDIRECTIONAL,” describing the usage of the port.

<*node-name*>

Is one or more valid PSpice A/D node name(s).

The external port statement, .EXTERNAL, applies only to nodes that have digital devices attached to them. When a node is included in a .EXTERNAL statement it is identified as a primary observation point. For example, if a PCB-level description is being modeled and simulated, .EXTERNALs (or their Schematics symbol counterparts), would be placed on the “edge pin” nets, thereby describing them as the “external interface points” of the network.

PSpice recognizes the nets marked as .EXTERNAL when reporting any sort of timing violation. When a timing violation occurs, PSpice analyzes the conditions that would permit the effects of such a condition to propagate through the circuit. If, during this analysis, a net marked as “external” is encountered, PSpice reports the condition as a “Persistent Hazard,” signifying that it has a potential effect on the externally visible behavior of the circuit.

Comments

For more information on Persistent Hazards, refer to your PSpice user’s guide.

Note

Port specifications are inserted into the netlist by Schematics whenever an external port symbol, EXTERNAL_IN, EXTERNAL_OUT, or EXTERNAL_BI is used. Refer to your PSpice user’s guide.

.FOUR (Fourier Analysis)

Purpose

Fourier analysis decomposes the results of a transient analysis into Fourier components.

General Form

.FOUR <*frequency value*> [*no. harmonics value*] <*output variable*>

Example

```
.FOUR 10kHz V(5) V(6,7) I(VSENS3)
.FOUR 60Hz 20 V(17)
.FOUR 10kHz V([OUT1],[OUT2])
```

The analysis results are obtained by performing a Fourier integral on the results from a transient analysis. The analysis must be supplied with specified output variables using evenly spaced time points. The time interval used is <*print step value*> in the .TRAN command, or 1% of the <*final time value*> (TSTOP) if smaller, and a 2nd-order polynomial interpolation is used to calculate the output value used in the integration. The DC component, the fundamental, and the 2nd through 9th harmonics of the selected voltages and currents, are calculated by default, although more harmonics can be specified. A .FOUR command requires a .TRAN command. Fourier analysis does not require a .PRINT, .PLOT, or .PROBE command. The tabulated results are written to the output file (".out") as the transient analysis is completed.

<*output variable*> Is an output variable of the same form as in a .PRINT command or .PLOT command for a transient analysis.

<*frequency value*> Is the fundamental frequency. Not all of the transient results are used, only the interval from the end, back to 1/<*frequency value*> before the end is used. This means that the transient analysis must be at least 1/<*frequency value*> seconds long.

Note *The results of the .FOUR command are only available in the output file. They cannot be viewed in Probe.*

.FUNC (Function)

Purpose

The .FUNC command is used to define "functions" that are used in expressions. Besides their obvious flexibility, they are useful for where there are several similar subexpressions in a circuit file.

General Form

.FUNC <*name*> ([*arg*]*) {<*body*>}

Example

```
.FUNC E(x) {exp(x)}
.FUNC DECAY(CNST) {E(-CNST*TIME)}
.FUNC TRIWAV(x) {ACOS(COS(x))/3.14159}
.FUNC MIN3(A,B,C) {MIN(A,MIN(B,C))}
```

.FUNC

This command does not have to precede the first use of the "function" name. Functions cannot be redefined and the function name must not be the same as any of the predefined functions (e.g., SIN and SQRT). See Before you Begin for a list of valid expressions.

Note *.FUNC arguments cannot be node names.*

<*body*>

Can refer to other (previously defined) functions: the second example, DECAY, uses the first example, E.

[*arg*]

Up to 10 arguments can be used in a definition. The number of arguments in the use of a function must agree with the number in the definition. Functions can be defined as having no arguments, but the parentheses are still required. Parameters, TIME, other functions, and the Laplace variable "s" are allowed in the body of function definitions.

The <*body*> of a defined function is handled in the same way as any math expression, it is enclosed in curly braces ({}), as shown in the examples. Previous versions of PSpice did not require this, so, for compatibility, the <*body*> can be read without braces but a warning is generated.

Note *Creating a file of popular .FUNC definitions and accessing them using a .INC command near the beginning of the circuit file can be helpful. .FUNCs can also be defined in subcircuits. In those cases they only have local scope.*

.IC (Initial Bias Point Condition)

Purpose

The .IC command is used to set initial conditions for both small-signal and transient bias points. Initial conditions can be given for some or all of the circuit's nodes.

General Form

.IC < V(<*node*> [,<*node*>])=<*value*> >*
.IC <I(<*inductor*>)=<*value*>>*

Example

```
.IC V(2)=3.4 V(102)=0 V(3)=-1V I(L1)=2uAmp
.IC V(InPlus,InMinus)=1e-3 V(100,133)=5.0V
```

The voltage between two nodes and the current through an inductor can be specified. During bias calculations, PSpice clamps the voltages to specified values by attaching a voltage source with a 0.0002 ohm series resistor between the specified nodes. After the bias point has been calculated and the transient analysis started, the node is "released."

<*value*> Is a voltage which is assigned to <*node*>, or a current which is assigned to an inductor, for the duration of the bias point calculation.

Note *The .IC sets the initial conditions for the bias point only. It does not affect a DC sweep.*

If the circuit contains both the .IC command and .NODESET command for the same node or inductor, the .NODESET command is ignored (.IC overrides .NODESET).

Refer to your PSpice user's guide for more information on setting initial conditions.

Note *A .IC command which impose nonzero voltages on inductors cannot work properly, since inductors are assumed to be short circuits for bias point calculations. However, inductor currents can be initialized.*

.INC (Include File)

Purpose

The .INC command is used to insert the contents of another file.

General Form

.INC <"*file name*">

Example

```
.INC "SETUP.CIR"
.INC "C:\LIB\VCO.CIR"
```

<"*file name*">

Can be any character string which is a legal file name for the computer system.

Note

For Unix based systems, file names are case sensitive. The file extension is not defaulted to ".inc". If a file name is specified, it must include its extension.

Including a file is the same as bringing the file's text into the circuit file. Everything in the included file is actually read in. The comments of the included file are then treated just as if they were found in the parent file.

Included files can contain any legal PSpice statements, but the following notes must apply:

- The included files should not contain title lines unless they are commented
- .END command (if present), should mark only the end of the included file,
- Included files can be nested. Up to 4 levels of "including" are allowed.

Comments

Every model and subcircuit definition, even if not needed, takes up space in the memory.

.LIB (Library File)

Purpose

The .LIB command is used to reference a model or subcircuit library in another file.

General Form

.LIB ["*file name*"]

Example

```
.LIB
.LIB linear.lib
.LIB "C:\lib\bipolar.lib"
```

file name

Can be any character string which is a legal file name for the computer system.

Note *For Unix based systems, file names are case sensitive. The file extension is not defaulted to ".lib". If a file name is specified, it must include its extension.*

If *file name* is left off, all references are defaulted to the master library file, "nom.lib." When a library file is referenced in Schematics, PSpice first searches for the file in the current working directory, and then in the directory specified by the LIBPATH variable (set in "msim.ini").

When any library is modified, PSpice creates an index file only on the first time that the library is used. The index file is organized in a way which allows PSpice to find a particular .MODEL or .SUBCKT quickly, in spite of how large the library file is.

Comments

The index files have to be regenerated each time the library is changed. Because of this, it is advantageous to configure separately any frequently changed libraries.

The NOM.LIB normally contains references to all parts in MicroSim's Standard Model Library. NOM.LIB can be edited to include your custom model references.

Library files can contain:

- comments,
- .MODEL commands,
- subcircuit definitions (including the .ENDS command),
- .PARAM commands,
- .FUNC commands, and
- .LIB commands.

No other statements are allowed. For further discussion of library files, refer to your PSpice user's guide.

.LOADBIAS (Load Bias Point File)

Purpose

The .LOADBIAS command is used to load the contents of a bias point file.

General Form

.LOADBIAS <"*file name*">

Example

```
.LOADBIAS "SAVETRAN.NOD"
.LOADBIAS "C:\PROJECT\INIT.FIL"
```

Normally, the bias point file has been produced by a previous circuit simulation using the .SAVEBIAS command, described on **1-52**.

<“*file name*”>

Can be any character string which is a legal computer system file name, but it must be enclosed in quotation marks.

The bias point file is a text file which contains one or more comment lines, and a .NODESET command having the bias point voltage or inductor current values. If a fixed value for a transient analysis bias point needs to be set, this can be edited to replace the .NODESET command with a .IC command.

Note

Any nodes mentioned in the loaded file that are not present in the circuit are ignored, and a warning message will be generated.

Comments

To echo the .LOADBIAS file contents to the output file, use the EXPAND option on the .OPTIONS command.

.MC (Monte Carlo Analysis)

Purpose

The .MC command causes a Monte Carlo (statistical) analysis of the circuit and multiple runs of the selected analysis (DC, AC, or transient) are performed.

General Form

.MC <*#runs value*> <*analysis*> <*output variable*>
+ <*function*> [*option*]* [SEED=*value*]

Example

```
.MC 10 TRAN V(5) YMAX
.MC 50 DC IC(Q7) YMAX LIST
.MC 20 AC VP(13,5) YMAX LIST OUTPUT ALL
.MC 10 TRAN V([OUT1],[OUT2]) YMAX SEED=9321
```

The first run uses nominal values of all components. Subsequent runs use variations on model parameters as specified by the DEV and LOT tolerances on each .MODEL parameter (see .MODEL (Model) command section for details on the DEV and LOT tolerances).

<*#runs value*> The total number of runs to be performed (for printed results the upper limit is 2,000, and if the output is to be viewed using Probe, the limit is 400).

The other specifications on the .MC command control the output generated by the Monte Carlo analysis.

<*analysis*> At least one DC, AC, or TRAN must be specified for <*analysis*>. This analysis is repeated in subsequent passes. All analyses that the circuit contains are performed during the nominal pass. Only the selected analysis is performed during subsequent passes.

<*output variable*> Identical in format to that of a .PRINT output variable; see **1-45** for .PRINT examples.

<*function*>

Specifies the operation to be performed on the values of <*output variable*> to reduce these to a single value. This value is the basis for the comparisons between the nominal and subsequent runs.

The <*function*> must be one of the following.

Function	Definition
YMAX	Find the absolute value of the *greatest difference* in each waveform from the nominal run.
MAX	Find the *maximum value* of each waveform.
MIN	Find the *minimum value* of each waveform.
RISE_EDGE(<*value*>)	Find the *first occurrence* of the waveform crossing *above* the threshold <*value*>. The waveform must have one or more points at or below <*value*> followed by one above; the output value listed is the first point that the waveform increases above <*value*>.
FALL_EDGE(<*value*>)	Find the *first occurrence* of the waveform crossing *below* the threshold <*value*>. The waveform must have one or more points at or above <*value*> followed by one below; the output value listed is where the waveform decreases below <*value*>.

Note *<function> and all [option]s (except for <output type>) have no effect on the Probe data that is saved from the simulation. They are only applicable to the output file.*

[*option*]* Can include zero or more of the following.

Option	Definition	Type Example
LIST	Lists, at the beginning of each run, the model parameter values actually used for each component during that run.	
OUTPUT <*output type*>	Asks for an output from subsequent runs, after the nominal (first) run. The output from any run is governed by a .PRINT, .PLOT, and .PROBE command in the file. If OUTPUT is omitted, then only the nominal run produces output. The <*output type*> is one of the ones shown in the Type Example column	ALL forces all output to be generated (including the nominal run). FIRST <*N*> generates output only during the first *n* runs. EVERY <*N*> generates output every n^{th} run. RUNS <*N*>* does analysis and generates output only for the listed runs. Up to 25 values can be specified in the list.
RANGE* (<*low value*>, <*high value*>)	Restricts the range over which <*function*> is evaluated. An "*" can be used in place of a <*value*> to show "for all values". See the range examples in the Type Example column.	YMAX RANGE(*,.5) YMAX is evaluated for values of the sweep variable (e.g., time and frequency) of .5 or less. MAX RANGE(-1,*) The maximum of the output variable is found for values of the sweep variable of -1 or more.

* If RANGE is omitted, then <function> is evaluated over the whole sweep range. This is equivalent to RANGE(*,*).

[SEED=*value*] Defines the seed for the random number generator within the Monte Carlo analysis (*The Art of Computer Programming*, Donald Knuth, vol. 2, pg. 171, “subtractive method”).

<*value*> Must be an odd integer ranging from 1 to 32,767. If the seed value is not set, it defaults to 17,533.

For almost all analyses, the default seed value is adequate to achieve a constant set of results. The seed value can be modified within the integer value as required.

Comments For more information on Monte Carlo analysis, refer to your PSpice user’s guide.

.MODEL (Model)

Purpose

The .MODEL command defines a set of device parameters which can be referenced by devices in the circuit.

General Form

.MODEL <*model name*> [AKO: <*reference model name*>]
+ <*model type*>
+ ([<*parameter name*> = <*value*> [*tolerance specification*]]*
+ [T_MEASURED=<*value*>] [[T_ABS=<*value*>] or
+ [T_REL_GLOBAL=<*value*>] or [T_REL_LOCAL=<*value*>]])

Example

```
.MODEL RMAX RES (R=1.5 TC1=.02 TC2=.005)
.MODEL DNOM D (IS=1E-9)
.MODEL QDRIV NPN (IS=1E-7 BF=30)
.MODEL MLOAD NMOS(LEVEL=1 VTO=.7 CJ=.02pF)
.MODEL CMOD CAP (C=1 DEV 5%)
.MODEL DLOAD D (IS=1E-9 DEV .5% LOT 10%)
.MODEL RTRACK RES (R=1 DEV/GAUSS 1% LOT/UNIFORM 5%)
.MODEL QDR2 AKO:QDRIV NPN (BF=50 IKF=50m)
```

The examples are of the .MODEL parameter. The last example uses the AKO syntax to reference the parameters of the model QDRIV in the third example.

<*model name*> The model name which is used to reference a particular model.

<*reference model name*>

The model types of the current model and the AKO (A Kind Of) reference model must be the same. The value of each parameter of the referenced model is used unless overridden by the current model, e.g., for QDR2 in the last example, the value of IS derives from QDRIV, but the values of BF and IKF come from the current definition. Parameter values or formulas are transferred, but not the tolerance specification. The referenced model can be in the main circuit file, accessed through a .INC command, or it can be in a library file (see the .LIB (Library File) command, 1-18).

<*model type*> The <*model type*> is the device type and must be one of the types outlined in the following table.

Devices can only reference models of a corresponding type; e.g.,

- A JFET can reference a model of types NJF or PJF, but not of type NPN.
- There can be more than one model of the same type in a circuit, although they must have different names.

Following the <*model type*> is a list of parameter values enclosed by parentheses. None, any, or all of the parameters can be assigned values. Default values are used for all unassigned parameters. The lists of parameter names, meanings, and default values are found in the individual device descriptions.

Model Type	Instance Name	Type of Device
CAP	Cxxx	capacitor
CORE	Kxxx	nonlinear, magnetic core (transformer)
D	Desexed	diode
DINPUT	Nxxx	digital input device (receive from digital)
DOUTPUT	Oxxx	digital output device (transmit to digital)
GASFET	Bxxx	N-channel GaAs MESFET
IND	Lxxx	inductor
ISWITCH	Wxxx	current-controlled switch
LPNP	Qxxx	lateral PNP bipolar transistor
NIGBT	Zxxx	N-channel insulated gate bipolar transistor (IGBT)
NJF	Jxxx	N-channel junction FET
NMOS	Mxxx	N-channel MOSFET
NPN	Qxxx	NPN bipolar transistor
PJF	Jxxx	P-channel junction FET
PMOS	Mxxx	P-channel MOSFET
PNP	Qxxx	PNP bipolar transistor
RES	Rxxx	resistor
TRN	Txxx	lossy transmission line
UADC	Uxxx	multi-bit analog-to-digital converter
UDAC	Uxxx	multi-bit digital-to-analog converter

Model Type	Instance Name	Type of Device
UDLY	Uxxx	digital delay line
UEFF	Uxxx	edge-triggered flip-flop
UGATE	Uxxx	standard gate
UGFF	Uxxx	gated flip-flop
UIO	Uxxx	digital I/O model
UTGATE	Uxxx	tristate gate
VSWITCH	Sxxx	voltage-controlled switch

[*tolerance specification*]

Can be appended for each parameter, using the format:

[DEV [*track & dist*] <*value*>[%]] [LOT [*track & dist*] <*value*>[%]]

to specify an individual device (DEV) and the device lot (LOT) parameter value deviations. The tolerance specification is used by the .MC analysis only.

The LOT tolerance requires that all devices that refer to the same model use the same adjustments to the model parameter. DEV tolerances are independent, that is each device varies independently. The "%" shows a relative (percentage) tolerance. If it is omitted, <*value*> is in the same units as the parameter itself.

[*track & dist*] Specifies the tracking and non-default distribution, using the format:

[/<*lot #*>][/<*distribution name*>].

These specifications must immediately follow the keywords DEV and LOT (without spaces) and are separated by "/".

<*lot #*> Specifies which of ten random number generators, numbered 0 through 9, are used to calculate parameter value deviations. This allows deviations to be correlated between parameters in the same model, as well as between models. The generators for DEV and LOT tolerances are distinct: there are ten generators for DEV tracking and ten generators for LOT tracking. Tolerances without <*lot #*> get individually generated random numbers.

<distribution name>

The distribution name is one of the following. The default distribution can be set by using the .OPTIONS command DISTRIBUTION parameter.

Distribution Name	Function
UNIFORM	Generates uniformly distributed deviations over the range ±<*value*>.
GAUSS	Generates deviations using a Gaussian distribution over the range ±3σ and <*value*> specifies the ±1σ deviation (i.e., this generates deviations greater than ±<*value*>).
<*user name*>	Generates deviations using a user-defined distribution and <*value*> specifies the ±1 deviation in the user-definition; see the .DISTRIBUTION (User-Defined Distribution) command (**1-10**).

Comments

For more information refer to your PSpice user's guide.

Temperature Setting Parameters

Some Passive and semiconductor devices (C, L, R, B, D, J, M, and Q) have two levels of temperature attributes which can be customized on a model by model basis. First, the temperature at which the model parameters were measured can be defined using one of the following model parameter formats in the .MODEL command line:

T_MEASURED = <*literal value*>
T_MEASURED = { <*parameter*> }

This overrides the nominal TNOM value which is set in the .OPTIONS command line (default = 27°C). All other parameters listed in the .MODEL command are assumed to have been measured at T_MEASURED.

In addition to the measured model parameter temperature, current device temperatures can be customized to override the circuit's "global temperature" specification defined by the .TEMP command line (or equivalent .STEP TEMP or .DC TEMP). There are three forms as shown in Table 1-5

Table 1-2 *Model Parameters for Device Temperature*

Description	.MODEL Format	Parameter Format	Referencing Device Temperature
absolute temperature	standard	T_ABS=<*value*>	T_ABS
relative to current temperature	standard	T_REL_GLOBAL= <*value*>	*global temperature* + T_REL_GLOBAL
relative to AKO model temperature	AKO	T_REL_LOCAL= <*value*>	T_ABS(*AKO Model*) + T_REL_LOCAL

For all formats, <*value*> can be a literal value or a parameter of the form {<*parameter name*>}. *A maximum of one device temperature customization from* **Table 1-2** *can coexist using the* T_MEASURED *customization.* For instance,

```
.MODEL PNP_NEW PNP( T_ABS=35 T_MEASURED=0 BF=90 )
```

defines a new model PNP_NEW where BF was measured at 0°C. Any bipolar transistor referencing this model has an absolute device temperature of 35°C.

The following example demonstrates device temperatures set relative to the global temperature of the circuit:

```
.TEMP 10 30 40
.MODEL PNP_NEW PNP( T_REL_GLOBAL=-5 BF=90 )
```

This produces three PSpice runs where global temperature changes from 10° to 30° to 40°C, respectively, and any bipolar transistor that references the PNP_NEW model has a device temperature of 5°, 25°, or 35°C, respectively.

The following example sets the device temperature relative to a referenced AKO model:

```
.MODEL PNP_NEW PNP( AKO:PNP_OLD T_REL_LOCAL=10)
.MODEL PNP_OLD PNP( T_ABS=20)
```

Any bipolar transistor referencing the PNP_NEW model has a device temperature of 30°C.

There are a few special considerations when using these temperature parameters:

1. If the technique for current device temperature is using the value relative to an AKO model's absolute temperature (T_ABS), and the AKO referenced model does not specify T_ABS, then the T_REL_LOCAL specification is ignored and the standard global temperature specification is used.

2. These temperature parameters cannot be used with the DEV and LOT model parameter tolerance feature.

3. A DC sweep analysis can be performed on these parameters so long as the temperature parameter assignment is to a variable parameter. For example:

```
.PARAM PTEMP 27
.MODEL PNP_NEW PNP ( T_ABS={PTEMP} )
.DC PARAM PTEMP 27 35 1
```

   This method produces a single DC sweep in PSpice where any bipolar transistor referencing the PNP_NEW model has a device temperature which is swept from 27°C to 35°C in 1°C increments.

   A similar effect can be obtained by performing a parametric analysis. For instance:

```
.PARAM PTEMP 27
.MODEL PNP_NEW PNP( T_ABS={PTEMP} )
.STEP PARAM PTEMP 27 35 1
```

   This method produces 9 PSpice runs where the PNP_NEW model temperature steps from 27°C to 35°C in increments of 1°C, one step per run.

4. The effect of a temperature parameter is evaluated once prior to the bias point calculation, unless parameters are swept by means of a .DC PARAM or .STEP PARAM analysis described above. In these cases, the temperature parameter's effect is reevaluated once for each value of the swept variable.

.NODESET (Endostea)

Purpose

The .NODESET command helps calculate the bias point by providing an initial best guess for some node voltages and/or inductor currents. Some or all of the circuit's node voltages and inductor currents can be given the initial guess, and in addition, the voltage between two nodes can be specified.

General Form

.NODESET < V(<*node*> [,<*node*>])=<*value*> >*
.NODESET <I(<*inductor*>)=<*value*>>

Example

```
.NODESET V(2)=3.4 V(102)=0 V(3)=-1V I(L1)=2uAmp
.NODESET V(InPlus,InMinus)=1e-3 V(100,133)=5.0V
```

This command is 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, nor during a transient analysis.

Unlike the .IC command, .NODESET provides only an initial guess for some initial values. It does not clamp those nodes to the specified voltages. However, by providing an initial guess, .NODESET can be used to "break the tie" in, for instance, a flip-flop, and make it come up in a required state.

If both the .IC command and .NODESET command are present, the .NODESET command is ignored for the bias point calculations (.IC overrides .NODESET).

Comments

For Schematics-based designs, refer to your PSpice user's guide for more information on setting initial conditions.

.NOISE (Noise Analysis)

Purpose

The .NOISE command causes a noise analysis of the circuit to be performed.

General Form

.NOISE V(<*node*> [,<*node*>]) <*name*> [*interval value*]

Example

```
.NOISE V(5)   VIN
.NOISE V(101) VSRC 20
.NOISE V(4,5) ISRC
.NOISE V([OUT1],[OUT2]) V1
```

Note *A noise analysis is performed in conjunction with AC analysis and requires a .AC command.*

V(<*node*> [,<*node*>])

An output voltage. It has a form such as V(5), which is the voltage at the output node five, or a form such as V(4,5), which is the output voltage between two nodes four and five.

<*name*>

The name of an independent voltage or current source where the equivalent input noise is calculated. The <*name*> is not itself a noise generator, but only a place where the equivalent input noise is calculated.

The noise-generating devices in a circuit are the resistors and the semiconductor devices. For each frequency of the AC analysis, each noise generator's contribution is calculated and propagated to the output nodes. At that point, all the propagated noise values are RMS-summed to calculate the total output noise. The gain from the input source to the output voltage, the total output noise, and the equivalent input noise are all calculated. If

<*name*> is a voltage source then the input noise units are

volt/hertz$^{1/2}$

<*name*> is a current source then the input noise units are

amp/hertz$^{1/2}$

The output noise units are always volt/hertz$^{1/2}$.

[*interval value*] The interval value is an integer which specifies low after the detailed noise analysis data is written to the output file.

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. These values are the noise amounts propagated to the output nodes, not the noise amounts at each generator. If [*interval value*] is not present, then no detailed table is printed.

The detailed table is printed while the analysis is being performed, and does not need a .PRINT command or a .PLOT command. The output noise and equivalent input noise can be printed in the output by using a .PRINT command or a .PLOT command.

.OP (Bias Point)

Purpose

The .OP command causes detailed information about the bias point to be printed.

General Form

.OP

Example

```
.OP
```

This command does not write output to the Probe data file. The bias point is calculated whether or not there is a .OP command. Without the .OP command, the only information about the bias point in the output is a list of the node voltages, voltage source currents, and total power dissipation.

Using a .OP command can cause the small-signal (linearized) parameters of all the nonlinear controlled sources and all the semiconductor devices to be printed in the output file.

The .OP command controls the output for the regular bias point only. The .TRAN command controls the output for the transient analysis bias point.

Note *If no other analysis is performed, no Probe data file can be created.*

.OPTIONS (Analysis Options)

Purpose

The .OPTIONS command is used to set all the options, limits, and control parameters for the simulator.

General Form

.OPTIONS [*option name*]* [<*option name*>=<*value*>]*

Example

```
.OPTIONS NOECHO NOMOD DEFL=12u DEFW=8u DEFAD=150p
+ DEFAS=150p
.OPTIONS ACCT RELTOL=.01
.OPTIONS DISTRIBUTION=GAUSS
.OPTIONS DISTRIBUTION=USERDEF1
```

The options can be listed in any order. There are two kinds of options: those with values, and those without values. Options without values are flags which are activated by simply listing the option name.

The .OPTIONS command is cumulative. That is, if there are two (or more) of the .OPTIONS command, the effect is the same as if all the options were listed together in one .OPTIONS command. If the same option is listed more than once, only its last value is used.

Table 1-3 lists the flag options. The default for any flag option is “off” or “no” (i.e., the opposite of specifying the option). Flag options affect the output file unless otherwise specified.

Table 1-3 *Flag Options*

Flag Option	Meaning
ACCT	Summary and accounting information is printed at the end of all the analyses (refer to your PSpice user's guide for further information on ACCT).
EXPAND	Lists devices created by subcircuit expansion and lists contents of the bias point file (see .SAVEBIAS (Save Bias Point to File) and .LOADBIAS (Load Bias Point File)).
LIBRARY	Lists lines used from library files.
LIST	Lists a summary of the circuit elements (devices).
NOBIAS	Suppresses the printing of the bias point node voltages.
NODE	Lists a summary of the connections (node table).
NOECHO	Suppresses a listing of the input file(s).
NOICTRANSLATE	Suppresses the translation of initial conditions (IC attributes) specified on capacitors and inductors into .IC statements (IC pseudocomponents). This means that IC attributes are ignored if the keyword Skip Bias Point (SKIPBP) is not put at the end of the .TRAN statement. [See .TRAN (Transient Analysis)].
NOMOD	Suppresses listing of model parameters and temperature updated values.
NOOUTMSG	Suppresses simulation error messages in output file.
NOPAGE	Suppresses paging and the banner for each major section of output.
NOPRBMSG	Suppresses simulation error messages in Probe data file.
NOREUSE	Suppresses the automatic saving and restoring of bias point information between different temperatures, Monte Carlo runs, worst-case runs, and parametric analyses (.STEP). [See also .SAVEBIAS (Save Bias Point to File) and .LOADBIAS (Load Bias Point File)].
OPTS	Lists values for all options.
STEPGMIN	Enable GMIN stepping. This causes a GMIN stepping algorithm to be applied to circuits that fail to converge. GMIN stepping is applied first, and if that fails, the simulator falls back to supply stepping.

The following option has a name as its value.

Table 1-4 *Option With a Name as its Value*

Option	Meaning	Default
DISTRIBUTION	default distribution for Monte Carlo deviations	UNIFORM

The default distribution is used for all of the deviations throughout the Monte Carlo analyses, unless specifically overridden for a particular tolerance. The default value for the default distribution is UNIFORM, but can also be set to GAUSS or a user-defined (<*user name*>) distribution. If a user-defined distribution is selected (as illustrated in the last example at the end of this section), a .DISTRIBUTION command must be included in the circuit file to define the user distribution for the tolerances. An example would be:

Example

```
.DISTRIBUTION USERDEF1 (-1,1) (.5,1) (.5,0) (1,0)
.OPTIONS DISTRIBUTION=USERDEF1
```

Table 1-5 lists the options containing values, with their default values.

Table 1-5 *Options With Their Default Values*

Options With Values	Meaning	Units	Default
ABSTOL	Best accuracy of currents.	amp	1pA
CHGTOL	Best accuracy of charges.	coulomb	.01pC
CPTIME*	CPU time allowed for this run.	sec	0**
DEFAD	MOSFET default drain area (AD).	meter^2	0
DEFAS	MOSFET default source area (AS).	meter^2	0
DEFL	MOSFET default length (L).	meter	100u
DEFW	MOSFET default width (W).	meter	100u
DIGFREQ	Minimum digital time step is 1/DIGFREQ.	hertz	10GHz
DIGDRVF	Minimum drive resistance (Input/Output UIO type model, DRVH (high) and DRVL (low) values).	ohm	2
DIGDRVZ	Maximum drive resistance (UIO type model, DRVH and DRVL values).	ohm	20K
DIGERRDEFAULT	Default error limit per digital constraint device.		20
DIGERRLIMIT	Maximum digital error message limit.		0**
DIGINITSTATE	Sets initial state of all flip-flops and latches in circuit: 0=clear, 1=set, 2=X.		2
DIGIOLVL	Default digital I/O level: 1-4; see UIO in .MODEL (Model).		1
DIGMNTYMX***	Default delay selector: 1=min, 2-typical, 3=max, 4=min/max.		2
DIGMNTYSCALE	Scale factor used to derive minimum delays from typical delays.		0.4

Table 1-5 *Options With Their Default Values*

Options With Values	Meaning	Units	Default
DIGOVRDRV	Ratio of drive resistances required to allow one output to override another driving the same node.		3
DIGTYMXSCALE	Scale factor used to derive maximum delays from typical delays.		1.6
GMIN	Minimum conductance used for any branch.	ohm^{-1}	1E-12
ITL1	DC and bias point “blind” repeating limit.		150
ITL2	DC and bias point “educated guess” repeating limit.		20
ITL4	The limit at any repeating point in transient analysis.		10
ITL5*	Total repeating limit for all points for transient analysis (ITL5=0 means ITL5=infinity).		0**
LIMPTS*	Maximum points allowed for any print table or plot (LIMPTS=0 means LIMPTS=infinity).		0**
NUMDG	Number of digits output in print tables (maximum 8 useful digits).		4
PIVREL*	Relative magnitude required for pivot in matrix solution.		1E-3
PIVTOL*	Absolute magnitude required for pivot in matrix solution.		1E-13
RELTOL	Relative accuracy of V’s and I’s.		.001
TNOM	Default nominal temperature (also the temperature at which model parameters are assumed to have been measured).	°C	27
VNTOL	Best accuracy of voltages.	volt	1uV
WIDTH	Same as “.WIDTH OUT=” statement (can be set to either 80 or 132).		80

*These options are available for modification in PSpice, but it is recommended that the program defaults be used.

**For these options zero means infinity.

***Setting the DIGMNTYMX=4 (min/max) directs PSpice to perform digital worst-case timing simulation. Refer to your PSpice user’s guide for a complete description.

Other PSpice features (such as those that originate for the digital CONSTRAINT devices monitoring timing relationships of digital nodes) produce warning messages in simulations. These messages are directed to the PSpice output file (and in Windows, to the Probe data file).

Options are available for controlling where and how many of these messages are generated. Table 1-9 summarizes the PSpice message origins and a brief description of their meaning. Currently, the condition messages supported are specific to digital device timing violations and digital worst-case timing hazards. Refer to your PSpice user's guide for information on digital worst-case timing.

Table 1-6 *PSpice Simulation Condition Messages*

Message Type	Meaning
Timing Violations	
FREQUENCY	The minimum or maximum frequency specification for a signal has **not** been satisfied. Minimum frequency violations show that the period of the measured signal is too long, while maximum frequency violations describe signals changing too rapidly.
GENERAL	A boolean expression described within the GENERAL constraint checker was evaluated and produced a "true" result.
HOLD	The minimum time required for a data signal to be stable *after* the assertion of a clock, has not been met.
SETUP	The minimum time required for a data signal to be stable *prior* to the assertion of a clock, has not been met.
RELEASE	The minimum time for a signal that has gone inactive (usually a control such as CLEAR) to remain inactive before the asserting clock edge, has not been met.
WIDTH	The minimum pulse width specification for a signal has not been satisfied. That is, a pulse that is too narrow was observed on the node.
Hazards	
AMBIGUITY CONVERGENCE	The convergence of conflicting rising and falling states (timing ambiguities) arriving at the inputs of a primitive, have produced a pulse (glitch) on the output.
CUMULATIVE AMBIGUITY	Signal ambiguities are additive, increased by propagation through each level of logic in the circuit. When the ambiguities associated with both edges of a pulse increase to the point where they would overlap, this is flagged as a cumulative ambiguity hazard.
DIGITAL INPUT VOLTAGE	When a voltage is out of range on a digital pin, PSpice uses the state whose voltage range is closest to the input voltage and continues using the simulation. A warning message is reported.

Table 1-6 *PSpice Simulation Condition Messages*

Message Type	Meaning
NET-STATE CONFLICT	When two or more outputs attempt to drive a net to different states, PSpice represents the conflict as an X (unknown) state. This usually results from improper selection of a bus driver's enable inputs.
SUPPRESSED GLITCH	A pulse applied to the input of a primitive that is shorter than the active propagation delay is ignored by PSpice. This can or can**not** be significant, depending upon the nature of the circuit. The reporting of the suppressed glitch hazard shows that there might be a problem with either the stimulus, or the path delay configuration of the circuit.
PERSISTENT HAZARD	If the effects of any of the other logic hazard messages mentioned in the output file are able to propagate to either an EXTERNAL port, or to any storage device in the circuit, they are flagged as PERSISTENT HAZARDs. (Refer to your PSpice user's guide for more details on PERSISTENT HAZARDs.)
ZERO-DELAY-OSCILLATION	If the output of a primitive changes more than 50 times within a single digital time step, the node is considered to be oscillating. PSpice reports this and aborts the run.

.PARAM (Parameter)

Purpose

The .PARAM statement defines the value of a parameter. A parameter name can be used in place of most numeric values in the circuit description. Parameters can be constants, or expressions involving constants, or a combination of these, and they can include other parameters.

General Form

.PARAM < <*name*> = <*value*> >*
.PARAM < <*name*> = { <*expression*> } >*

Example

```
.PARAM VSUPPLY = 5V
.PARAM VCC = 12V, VEE = -12V
.PARAM BANDWIDTH = {100kHz/3}
.PARAM PI = 3.14159, TWO_PI = {2*3.14159}
.PARAM VNUM = {2*TWO_PI}
```

<*name*>

Cannot begin with a number, and it cannot be one of the following predefined parameters, TIME, or .TEXT names.

There are several predefined parameters:

Predefined Parameter	Meaning
TEMP	temperature (*works using digital models only*)
VT	thermal voltage (*reserved*)
GMIN	shunt conductance for semiconductor *p-n* junctions

The parameter values must be either constants or expressions.

<*value*>

Constants (<*value*>) do not need “{” and “}”.

<*expression*>

Can contain constants or parameters.

The .PARAM statements are order independent. They can be used inside a subcircuit definition to create local subcircuit parameters.

Once defined, a parameter can be used in place of almost all numeric values in the circuit description with the following exceptions:

- **Not** in the transmission line parameters NL and F.
- **Not** in the *in-line* temperature coefficients for resistors (parameters can be used for the TC1 and TC2 resistor model parameters).
- **Not** in the PWL values for independent voltage and current source (V and I device) parameters.
- **Not** the E, F, G, and H device SPICE2G6 syntax for polynomial coefficient values and gain.

Parameters **cannot** be used in place of node numbers, nor can the values on an analysis command (e.g., TRAN and AC) be parameterized.

A .PARAM command can be in a library. The simulator can search libraries for parameters not defined in the circuit file, in the same way it searches for undefined models and subcircuits.

.PLOT (Plot)

Purpose

The .PLOT command allows results from DC, AC, noise, and transient analyses to be an output in the form of “line printer” plots in the “out” file.

General Form

.PLOT <*analysis type*> [*output variable*]*
+ ([<*lower limit value*> , <*upper limit value*>])*

Example

```
.PLOT DC  V(3)  V(2,3) V(R1)  I(VIN) I(R2) IB(Q13) VBE(Q13)
.PLOT AC  VM(2) VP(2)  VM(3,4) VG(5) VDB(5) IR(D4)
.PLOT NOISE INOISE ONOISE DB(INOISE) DB(ONOISE)
.PLOT TRAN V(3) V(2,3) (0,5V)   ID(M2) I(VCC) (-50mA,50mA)
.PLOT TRAN D(QA) D(QB) V(3) V(2,3)
.PLOT TRAN V(3) V(R1) V([RESET])
```

The last example illustrates how to plot the voltage at a node which has a name rather than a number. The first item to plot is a node voltage, the second item is the voltage across a resistor, and the third item to plot is another node voltage, even though the second and third items both begin with the letter “R”. The square brackets force the interpretation of names to mean node names.

Note *Lower and upper limit values do not apply to AC Analysis.*

Plots are made by using text characters to draw the plot, therefore, they work using any kind of printer.

<*analysis type*> Can be one of DC, AC, NOISE, or TRAN. Only one analysis type can be specified.

<*output variable*> Following the analysis type is a list of the output variables and (possibly) Y axis scales. A maximum of 8 output variables are allowed on one .PLOT command. However, an analysis can have any number of a .PLOT command. See the .PROBE (Probe) section on **1-46** for the syntax of the output variables.

The range and increment of the X axis is fixed by the analysis being plotted. The Y axis defaults to a “nice” range determined by the ranges of the output variables.

Note *The Y axis of frequency response plots (AC) is always logarithmic.*

If the different output variables differ considerably in their output ranges, then the plot is given more than one Y axis using ranges corresponding to the different output variables.

(<*lower limit value*>, <*upper limit value*>)

The range of the Y axis can be set by including the lower and upper limit values at the end of the .PLOT command.

This forces all output variables on the same Y axis to use the specified range. The same form, (<*lower limit value*>, <*upper limit value*>), can also be inserted one or more times in the middle of a set of output variables. Each occurrence defines one Y axis that has the specified range. All the output variables which come between it and the next range to the left in the .PLOT command are put on its corresponding Y axis. In the fourth example, the two voltage outputs go on the Y axis using the range (0,5V) and the two current outputs go on the Y axis using the range (-5mMA, 50mA).

.PRINT (Print)

Purpose

The .PRINT command allows results from DC, AC, noise, and transient analyses to be an output in the form of tables, referred to as print tables in the output file.

General Form

.PRINT[/DGTLCHG] <*analysis type*> [*output variable*]*

Example

```
.PRINT DC  V(3)  V(2,3) V(R1) I(VIN) I(R2) IB(Q13) VBE(Q13)
.PRINT AC  VM(2) VP(2) VM(3,4) VG(5) VDB(5) IR(6) II(7)
.PRINT NOISE INOISE ONOISE DB(INOISE) DB(ONOISE)
.PRINT TRAN V(3) V(2,3) ID(M2) I(VCC)
.PRINT TRAN D(QA) D(QB) V(3) V(2,3)
.PRINT/DGTLCHG TRAN QA QB RESET
.PRINT TRAN V(3) V(R1) V([RESET])
```

[/DGTLCHG] This is for digital output variables only. Values are printed for each output variable whenever one of the variables changes.

<*analysis type*> Can be one of DC, AC, NOISE, or TRAN. Only one analysis type can be specified for each .PRINT command.

<*output variable*> Following the analysis type is a list of the output variables. There is no limit to the number of output variables: the printout is split up depending on the width of the data columns (set using NUMDGT option) and the output width (set using WIDTH option). See the .PROBE (Probe) command for the syntax of output variables.

The values of the output variables are printed as a table having each column correspond to one output variable. The number of digits which are printed for analog values can be changed by NUMDGT on the .OPTIONS command.

The last example illustrates how to print a node which has a name rather than a number. The first item to print is a node voltage, the second item is the voltage across a resistor, and the third item to print is another node voltage, even though the second and third items both begin with the letter “R”. The square brackets force the interpretation of names to mean node names.

An analysis can have any number of a .PRINT command.

.PROBE (Probe)

Purpose

The .PROBE command writes the results from DC, AC, and transient analyses to a data file that is used by the Probe waveform analyzer.

General Form

.PROBE[/CSDF][*output variable*]*

Example

```
.PROBE
.PROBE V(3) V(2,3) V(R1) I(VIN) I(R2) IB(Q13) VBE(Q13)
.PROBE/CSDF
.PROBE V(3) V(R1) V([RESET])
.PROBE D(QBAR)
```

The first example (with no output variables) writes all the node voltages and all the device currents to the data file. The list of device currents written is the same as the device currents allowed as output variables.

The second example writes only those output variables specified to the data file.

The third example creates a data file in a text format using the Common Simulation Data File (CSDF) format, not a binary format. This format is primarily used for transfers between different computer families. This example illustrates how to specify a node which has a name rather than a number. The first item to output is a node voltage, the second item is the voltage across a resistor, and the third item to output is another node voltage, even though the second and third items both begin with the letter “R”. The square brackets force the interpretation of names to mean node names.

The last example only writes the output at digital node QBAR to the data file.

Comments

Refer to your PSpice user’s guide for a description of Probe and for information about using the Probe data file.

Note *Unlike the .PRINT command and .PLOT command, there are no analysis names before the output variables. Also, the number of output variables is unlimited.*

<output variable> This section describes the types of output variables allowed in a .PRINT, .PLOT, and .PROBE command. Each .PRINT or .PLOT can have up to 8 output variables. This format is similar to that used when calling up waveforms while running Probe.

See the tables below for a description of the possible output variables. If .PROBE is used without specifying a list of output variables, all of the circuit voltages and currents are stored for post-processing. When an output variable list is included, the data stored is limited to the listed items. This form is intended for users who want to limit the size of the Probe data file.

DC Sweep and Transient Analysis

For DC sweep and transient analysis, these are the available output variables:

General Form	Meaning of Output Variable
D(<*name*>)	digital value of <*name*> (a digital node)*
I(<*name*>)	current through a two terminal device
I*x*(<*name*>)	current into a terminal of a three or four terminal device (x is one of “B”, “D”, “G”, or “S”)
Iz(<*name*>)	current into one end of a transmission line (z is either “A” or “B”)
V(<*node*>)	voltage at a node
V(<+ *node*>, <- *node*>)	voltage between two nodes
V(<*name*>)	voltage across a two-terminal device
V*x*(<*name*>)	voltage at a non-grounded terminal of a device (see Ix)
V*z*(<*name*>)	voltage at one end of a transmission line (z is either “A” or “B”)
Vxy(<*name*>)	voltage across two terminals of a three or four terminal device type

*These values are available for transient and DC analysis only. For the .PRINT/DGTLCHG statement the “D( )” is optional.

Examples	Meaning of Output Variable
D(QA)	the value of digital node QA
I(D5)	current through diode D5
IG(J10)	current into gate of J10
V(3)	voltage between node three and ground
V(3,2)	voltage between nodes three and two
V(R1)	voltage across resistor R1
VA(T2)	voltage at port A of T2
VB(Q3)	voltage between base of transistor Q3 and ground
VGS(M13)	gate-source voltage of M13

For the V(<*name*>) and I(<*name*>) forms, where <*name*> must be the name of a two-terminal device, the devices are:

Character ID	Two-Terminal Device
C	capacitor
D	diode
E	voltage-controlled voltage source
F	current-controlled current source
G	voltage-controlled current source
H	current-controlled voltage source)
I	independent current source
L	inductor
R	resistor
S	voltage-controlled switch
V	independent voltage source
W	current-controlled switch

For the V*x*(<*name*>), V*xy*(<*name*>), and I*x*(<*name*>) forms, where <*name*> must be the name of a three or four-terminal device and *x* and *y* must each be a terminal abbreviation, the devices and the terminals are:

Three & Four-Terminal Device Type	Terminal Abbreviation
B (GaAs MESFET)	D (drain)
	G (gate)
	S (source)
J (Junction FET)	D (drain)
	G (gate)
	S (source)
M (MOSFET)	D (drain)
	G (gate)
	S (source)
	B (bulk, substrate)
Q (Bipolar transistor)	C (collector)
	B (base)
	E (emitter)
	S (substrate)
T (transmission line)	Va (near side voltage)
	Ia (near side current)
	Vb (far side voltage)
	Ib (far side current)
Z (IGBT)	C (collector)
	G (gate)
	E (emitter)

For the V*z*(<*name*>) and I*z*(<*name*>) forms, <*name*> must be the name of a transmission line (T device) and *z* must be “A” or “B”.

AC Analysis

For AC analysis, the output variables listed in the preceding section are augmented by adding a suffix. These are the available suffixes:

Suffix	Meaning of Output Variables for AC Analysis
none	magnitude
DB	magnitude in decibels
G	group delay (-*d*PHASE/*d*FREQUENCY)
I	imaginary part
M	magnitude
P	phase in degrees
R	real part

Examples	Meaning of Output Variables for AC Analysis
II(R13)	imaginary part of current through R13
IGG(M3)	group delay of M3's gate current
IR(VIN)	real part of I through VIN
IAG(T2)	group delay of current at port A of T2
V(2,3)	magnitude of complex voltage across nodes 2 & 3
VDB(R1)	db magnitude of V across R1
VBEP(Q3)	phase of base-emitter V at Q3
VM(2)	magnitude of V at node 2

Note *Current outputs for the F and G devices are not available for DC and transient analyses.*

For these devices, a zero-valued voltage source must be put in series with the device (or terminal) of interest. Then, the current through this voltage source can be printed or plotted .

Note *For AC analysis, the suffixes are ignored for a .PROBE command, but can be used in a .PRINT command and a .PLOT command, and when adding a trace in Probe. For example, in a .PROBE command, VDB(R1) is translated to V(R1) which is the raw data.*

Noise Analysis

For noise analysis, the output variables are predefined as follows.

Output Variable	Meaning of Output Variables for Noise Analysis
INOISE	Total RMS summed noise at input node
ONOISE	INOISE equivalent at output node
DB(INOISE)	INOISE in decibels
DB(ONOISE)	ONOISE in decibels

Note *The noise from any one device cannot be .PRINTed or .PLOTed. However, the print interval on the .NOISE command can be used to output this information.*

Comments Refer to your PSpice user's guide for more information on the use of text files in Probe. You can also consult Probe Help.

.SAVEBIAS (Save Bias Point to File)

Purpose

The .SAVEBIAS command is used to save the bias point node voltages and inductor currents, to a file. It is used subsequently with .LOADBIAS.

General Form

.SAVEBIAS <“*file name*”> <[OP] [TRAN] [DC]> [NOSUBCKT]
+[TIME=<*value*> [REPEAT]] [TEMP=<*value*>]
+ [STEP=<*value*>] [MCRUN=<*value*>] [DC=<*value*>]
+ [DC1=<*value*>] [DC2=<*value*>]

Example

```
.SAVEBIAS "OPPOINT" OP
.SAVEBIAS "TRANDATA.BSP" TRAN NOSUBCKT TIME=10u
.SAVEBIAS "SAVETRAN.BSP" TRAN TIME=5n REPEAT TEMP=50.0
.SAVEBIAS "DCBIAS.SAV" DC
.SAVEBIAS "SAVEDC.BSP" DC MCRUN=3 DC1=3.5 DC2=100
```

For the first example, the small-signal operating point (.AC or .OP) bias point is saved. In the second example, the transient bias point is written out at the time closest to, but not less than 10 usec. No bias point information for subcircuits is saved.

Use of the [REPEAT] keyword in the third example causes the bias point to be written out every 5ns when the temperature of the run is 50 degrees.

In the fourth example, because there are no parameters supplied, only the very first DC bias point is written to the file.

The fifth example saves the DC bias point when the following three conditions are all met: the first DC sweep value is 3.5, the second DC sweep value is 100, and the simulation is on the third Monte Carlo run. If only one DC sweep is being performed, then the keyword DC can be substituted for DC1.

<“*file name*”>

Any character string that is a legal file name for the computer system, and must be enclosed in quotation marks.

Only one analysis is specified in a .SAVEBIAS command (OP, TRAN, or DC). However, a circuit file can contain a .SAVEBIAS command for each of the three analysis types. If the simulation parameters do not match the keywords and values in the .SAVEBIAS command, then no file is produced.

[NOSUBCKT] When used, the node voltages and inductor currents for subcircuits are not saved.

[TIME=<*value*> [REPEAT]]
Used to define the transient analysis time at which the bias point is to be saved.

If REPEAT is not used, then the bias at the next time point greater than or equal to TIME=<*value*> is saved. If REPEAT is used, then TIME=<*value*> is the interval at which the bias is saved. However, only the latest bias is saved; any previous times are overwritten. The [TIME=<*value*> [REPEAT]] can only be used with a transient analysis.

[TEMP=<*value*>] Defines the temperature at which the bias point is to be saved.

[STEP=<*value*>] The step value at which the bias point is to be saved.

[MCRUN=<*value*>]
The number of the Monte Carlo or worst-case analysis run for which the bias point is to be saved.

[DC=<*value*>], [DC1=<*value*>], and [DC2=<*value*>]
Used to specify the DC sweep value at which the bias point is to be saved.

The [DC=<*value*>] should be used if there is only one sweep variable. If there are two sweep variables, then [DC1=<*value*>] should be used to specify the first sweep value and [DC2=<*value*>] should be used to specify the second sweep value.

The saved bias point information is in the following format: one or more comment lines that list items such as:

- circuit name, title, date and time of run, analysis, and temperature, or
- a single .NODESET command containing the bias point voltage values and inductor currents.

Only one bias point is saved to the file during any particular analysis. At the specified time, the bias point information and the operating point data for the active devices and controlled sources are written to the output file. When the supplied specifications on the .SAVEBIAS command line match the “state” of the simulator during execution, the bias point is written out.

Example of Usage

A .SAVEBIAS command and a .LOADBIAS command can be used to shorten the simulation time of large circuits, and also to aid in convergence.

A typical application for a .SAVEBIAS and a .LOADBIAS command is for a simulation which takes a considerable amount of time to converge to a bias point. The bias point can be saved using a .SAVEBIAS command and when the simulation is run again, the previous bias point calculated is used as a starting point for the bias solution to save processing time.

The following example illustrates this procedure for a transient simulation.

```
.SAVEBIAS "SAVEFILE.TRN" TRAN
```

When the simulation is run, the transient analysis bias point information is saved to the file SAVEFILE.TRN in the form of a .NODESET command. This .NODESET command provides the simulator with a starting solution for determining the bias point calculation for future simulations. To use this file, replace the .SAVEBIAS command in the circuit file using the following .LOADBIAS command.

```
.LOADBIAS "SAVEFILE.TRN"
```

Note *A .SAVEBIAS and .LOADBIAS command should not refer to the same file during the same simulation run. Use the .SAVEBIAS during the first simulation and the .LOADBIAS for subsequent ones.*

The simulator algorithms have been changed to provide an automatic saving and loading of bias point information under certain conditions. This automatic feature is used in parametric analyses (.STEP), DC sweeps (.DC), worst-case analyses (.WCASE), Monte Carlo analyses (.MC), and temperature analyses (.TEMP).

A typical application is a transient analysis where the bias point is calculated at several temperatures (such as .TEMP 0 10 20 30). As each new temperature is processed, the bias point for the previous temperature is used to find the new bias point. Since this process is automatic, the user does not have to change anything in the circuit file. However, there is some memory overhead since the bias point information is saved during the simulation. Disable the automatic saving feature, using the NOREUSE flag option in the .OPTIONS command as follows:

```
.OPTIONS NOREUSE
```

Another application for the .LOADBIAS and .SAVEBIAS command is the handling of convergence problems. Consider a circuit which has difficulty in starting a DC sweep. The designer has added a .NODESET command as shown below to help the simulator determine the bias point solution.

```
.NODESET V(3)=5.0V V(4)=2.75V
```

Even though this helps the simulator determine the bias point, the simulator still has to compute the starting values for each of the other nodes. These values can be saved using the following statement:

```
.SAVEBIAS "DCOP.NOD" DC
```

The next time the simulation is run, the .NODESET and .SAVEBIAS command should be removed and replaced using the following:

```
.LOADBIAS "DCOP.NOD"
```

This provides the starting values for all of the nodes in the circuit, and can assist the simulator in converging to the correct bias point for the start of the sweep. If convergence problems are caused by a change in the circuit topology, the designer can edit the bias point save file to change the values for specific nodes or to add new nodes.

.SENS (Sensitivity Analysis)

Purpose The .SENS command causes a DC sensitivity analysis to be performed.

General Form .SENS <*output variable*>*

Example

```
.SENS V(9) V(4,3) V(17) I(VCC)
```

By linearizing the circuit about the bias point, the sensitivities of each of the output variables to all the device values and model parameters is calculated and output data generated. This can generate large amounts of output data.

<*output variable*> Same format and meaning as in the .PRINT command for DC and transient analyses. However, when <*output variable*> is a current, it is restricted to be the current through a voltage source.

Device sensitivities are only provided for the following device types:

resistors,
independent voltage and current sources,
voltage and current-controlled switches,
diodes, and
bipolar transistors.

Note *The results of the .SENS command are only available in the output file. They cannot be viewed in Probe.*

.STEP (Parametric Analysis)

Purpose

The .STEP command causes a parametric sweep to be performed for all of the analyses of the circuit.

General Form

.STEP LIN <*sweep variable name*>
+ <*start value*> <*end value*> <*increment value*>

.STEP [DEC |OCT] <*sweep variable name*>
+ <*start value*> <*end value*> <*points value*>

.STEP <*sweep variable name*> LIST <*value*>*

The first general form is for doing a linear sweep. The second form is for doing a logarithmic sweep. The third form is for using a list of values for the sweep variable.

Example

```
.STEP VCE 0V 10V .5V
.STEP LIN I2 5mA -2mA 0.1mA
.STEP RES RMOD(R) 0.9 1.1 .001
.STEP DEC NPN QFAST(IS) 1E-18 1E-14 5
.STEP TEMP LIST 0 20 27 50 80 100
.STEP PARAM CenterFreq 9.5kHz 10.5kHz 50Hz
```

The first three examples, are for doing a linear sweep. The fourth example is for doing a logarithmic sweep. The fifth example is for using a list of values for the sweep variable.

The .STEP command is at the same “level” as the .TEMP command: all of the ordinary analyses (e.g., .DC, .AC, and .TRAN) are performed for each step. Once all the runs have finished, the specified .PRINT table or .PLOT plot for each value of the sweep is an output, just as for the .TEMP or .MC command. (Probe allows nested sweeps to be displayed as a family of curves.)

Sweep type

The sweep can be linear, logarithmic, or a list of values. For [*linear sweep type*], the keyword LIN is optional, but either OCT or DEC must be specified for the <*logarithmic sweep type*>. The sweep types are as follows.

Sweep Types	Meaning
LIN	Linear sweep. The sweep variable is swept linearly from the starting to the ending value. The <*increment value*> is the step size
OCT	Sweep by octaves. The sweep variable is swept logarithmically by octaves. The <*points value*> is the number of steps per octave.
DEC	Sweep by decades. The sweep variable is swept logarithmically by decades. The <*points value*> is the number of steps per decade.
LIST	Use a list of values. In this case there are no start and end values. Instead, the numbers that follow the keyword LIST are the values that the sweep variable is set to.

Note *The LIST values must be in either ascending or descending order.*

<sweep variable name>

The <*sweep variable name*> can be one of the following types.

Sweep Variable Name	Meaning
Source	A name of an independent voltage or current source. During the sweep, the source's voltage or current is set to the sweep value.
Model parameter	A model type and model name followed by a model parameter name in parenthesis. The parameter in the model is set to the sweep value.
Temperature	Use the keyword TEMP for <*sweep variable name*>. The temperature is set to the sweep value. For each value in the sweep, all the circuit components have their model parameters updated to that temperature.
Global Parameter	Use the keyword PARAM, followed by the parameter name, for <*sweep variable name*>). During the sweep, the global parameter's value is set to the sweep value and all expressions are reevaluated.

The .STEP command only steps the DC component of an AC source. In order to step the AC component of an AC source, a variable parameter has to be created. For example,

```
Vac  1  0  AC {variable}
.param variable=0
.step param variable 0  5  1
.ac  dec  100  1000  1e6
```

<*start value*> Can be greater or less than <*end value*>: that is, the sweep can go in either direction.

<*increment value*> and <*points value*>
Must be greater than zero.

The following examples illustrate two ways of stepping a resistor from 30 to 50 ohms in steps of 5 ohms.

This example uses a global parameter:

```
.PARAM RVAL = 1
R1 1 2 {RVAL}
.STEP PARAM RVAL 30,50,5
```

The parameter RVAL is global and PARAM is the keyword used by the .STEP command when using a global parameter.

The following example steps the resistor model parameter R:

```
R1 1 2 RMOD 1
.MODEL RMOD RES(R=30)
.STEP RES RMOD(R) 30,50,5
```

(Note**:** Do not use R={30}.)

RMOD is the model name, RES is the sweep variable name (a model type), and R is the parameter within the model to step. To step the value of the resistor, the line value of the resistor is multiplied by the R parameter value to achieve the final resistance value, that is:

final resistor value = line resistor value · R

Therefore, if the line value of the resistor is set to one ohm, the final resistor value is 1 · R or R. Stepping R from 30 to 50 ohms then steps the resistor value from 1 · 30 ohms to 1 · 50 ohms.

In both examples, all of the ordinary analyses (e.g., .DC, .AC, and .TRAN) are run for each step.

The .STEP command is similar to the .DC command and immediately raises the question of what happens if both .STEP and .DC try to set the same value. The same question can come up using the Monte Carlo analysis. The answer is that this is **not** allowed: no two analyses (.STEP, .TEMP, .MC, .WCASE, and .DC) can try to set the same value. This is flagged as an error during read-in and no analyses are performed.

The .STEP command provides the capability to look at the response of a circuit as a parameter varies. For example, how does the center frequency of a filter shift as a capacitor varies? Using .STEP, that capacitor can be varied, yielding a family of AC waveforms showing the variation. Similar comments apply to looking at, for example, propagation delay in transient analysis.

.STIMLIB (Stimulus Library File)

Purpose

Stimulus library files created by StmEd are made available to PSpice by the use of the .STMLIB command.

General Form

.STMLIB <*file name*[.*stl*]>

Example

```
.STMLIB mylib.stl
.STMLIB volts.stl
.STMLIB dgpulse
```

<*file name*>

Specification that identifies a file containing .STIMULUS commands.

.STIMULUS (Stimulus)

Purpose

.STIMULUS commands generally appear within stimulus libraries created by StmEd.

General Form

.STIMULUS <*stimulus name*> <*type*> <*type-specific parameters*>*

Example

```
.STIMULUS  InputPulse PULSE  (-1mv 1mv 2ns 2ns 50ns 100ns)

.STIMULUS  DigitalPulse  STIM  (1,1)
+  0S   1
+  10NS 0
+  20NS 1

.STIMULUS 50KHZSIN  SIN  (0 5 50KHZ 0 0 0)
```

<*stimulus name*>

Is the name by which the stimulus is referred to by the source devices (V or I), or by the digital STIM device.

The .STIMULUS command definition encompasses only the Transient specification portion of what is allowed in the V or I device syntax.

.SUBCKT, .ENDS (Subcircuit and End Subcircuit)

Purpose

The .SUBCKT definition statement starts the definition of a subcircuit by specifying its name, the number and order of its terminals, and the names and default parameters which control its behavior. Subcircuits are instantiated by the X devices in **Chapter 2,Analog Devices**. The .ENDS command marks the end of a subcircuit definition.

General Form

.SUBCKT *<name>* [*node*]*
+ [OPTIONAL: < <*interface node*> = <*default value*> >*]
+ [PARAMS: < <*name*> = <*value*> >*]
+ [TEXT: < <*name*> = <*text value*> >*]
...
.ENDS

Example

```
.SUBCKT OPAMP 1 2 101 102 17
...
.ENDS

.SUBCKT FILTER INPUT, OUTPUT PARAMS: CENTER=100kHz,
+ BANDWIDTH=10kHz
...
.ENDS

.SUBCKT PLD IN1 IN2 IN3 OUT1
+  PARAMS: MNTYMXDLY=0 IO_LEVEL=0
+  TEXT: JEDEC_FILE="PROG.JED"
...
.ENDS

.SUBCKT  74LS00  A B Y
+  OPTIONAL: DPWR=$G_DPWR DGND=$G_DGND
+  PARAMS: MNTYMXDLY=0 IO_LEVEL=0
...
.ENDS
```

The subcircuit definition is ended using a .ENDS command. All of the netlist between .SUBCKT and .ENDS is included in the definition. Whenever the subcircuit is used, by an X device, all of the netlist in the definition replaces the X device.

<*name*> The name is used by an X device to reference the subcircuit.

[*node*]* An optional list of nodes (pins). This is optional because it is possible to specify a subcircuit that has no interface nodes.

There must be the same number of nodes in the subcircuit calling statements as in its definition. When the subcircuit is called, the actual nodes (the ones in the calling statement) replace the argument nodes (the ones in the defining statement).

Note *Do not use 0 ("zero") in this node list: that is reserved for global "ground" node.*

OPTIONAL: The "OPTIONAL:" keyword allows specification of one or more optional nodes (pins) in the subcircuit definition.

The optional nodes are stated as pairs consisting of an interface node and its default value. If an optional node is not specified in an X device, its default value is used inside the subcircuit; otherwise, the value specified in the definition is used.

This feature is particularly useful when specifying power supply nodes, because the same nodes are normally used in every device. This makes the subcircuits easier to use because the same two nodes do not have to be specified in each subcircuit statement. This method is used in the libraries provided with the Digital Simulation feature.

In the example of the 74LS00 subcircuit, the following subcircuit reference uses the default power supply nodes $G_DPWR and $G_DGND:

```
X1 IN1 IN2 OUT 74LS00
```

To specify your own power supply nodes MYPOWER and MYGROUND, use the following subcircuit instantiation:

```
X2 IN1 IN2 OUT MYPOWER MYGROUND 74LS00
```

If wanted, one optional node in the subcircuit instantiation can be provided. In the following subcircuit instantiation, the default $G_DGND would be used:

```
X3 IN1 IN2 OUT MYPOWER 74LS00
```

However, to specify values beyond the first optional node, all nodes previous to that node must be specified. For example, to specify your own ground node, the default power node before it must be explicitly stated:

```
X4 IN1 IN2 OUT $G_DPWR MYGROUND 74LS00
```

The keyword PARAMS: allows values to be passed into subcircuits as arguments and used in expressions inside the subcircuit. The keyword TEXT: allows text values to be passed into subcircuits as arguments and used as expressions inside the subcircuit. Once defined, a text parameter can be used in the following places:

- To specify a JEDEC file name on a PLD device.
- To specify an Intel Hex file name to program a ROM device or initialize a RAM device.
- To specify a stimulus file name or signal name on a FSTIM device.
- To specify a text parameter to a (lower level) subcircuit.
- As part of a text expression used in one of the above.

Note *The text parameters and expressions are currently only used by the Digital Simulation feature.*

Subcircuits can be nested. That is, an X device can appear between a .SUBCKT and a .ENDS command. However, subcircuit definitions *cannot be nested*. That is, a .SUBCKT statement cannot appear in the statements between a .SUBCKT and a .ENDS.

Subcircuit definitions should contain only device instantiations (statements without a leading ".") and possibly .IC, .NODESET, .MODEL, .PARAM, or, .FUNC statements. Models, parameters, and functions defined within a subcircuit definition are *available only within the subcircuit definition* in which they appear. Also, if a .MODEL, .PARAM, or a, .FUNC statement appears in the main circuit, it is available in the main circuit and all subcircuits.

Node, device, and model names are local to the subcircuit in which they are defined. It is acceptable to use a name in a subcircuit which has already been used in the main circuit. When the subcircuit is expanded, all its names are prefixed using the subcircuit instance name: for example, "Q13" becomes "X3.Q13" and node "5" becomes "X3.5" after expansion. After expansion all names are unique. The *only exception* is the use of global node names (refer to your PSpice user's guide) which are not expanded.

.TEMP (Temperature)

Purpose

The .TEMP statement sets the temperature at which all analyses are done.

General Form

.TEMP <*temperature value*>*

Example

```
.TEMP 125
.TEMP 0 27 125
```

The temperatures are in degrees Centigrade. If more than one temperature is given, then all analyses are performed for each temperature.

It is assumed that the model parameters were measured or derived at the nominal temperature, TNOM (27°C by default). See the .OPTIONS (Analysis Options) command (page **1-35**) for setting TNOM. .TEMP behaves similarly to the list variant of the .STEP statement, with the stepped variable being the temperature.

.TEXT (Text Parameter)

Purpose

The command .TEXT is followed by a list of names and text values.

General Form

.TEXT < <*name*> = "<*text value*>" >*
.TEXT < <*name*> = | <*text expression*> | >*

Example

```
.TEXT MYFILE = "FILENAME.EXT"
.TEXT FILE = "ROM.DAT", FILE2 = "ROM2.DAT"
.TEXT PROGDAT = |"ROM"+TEXTINT(RUN_NO)+".DAT"|
.TEXT DATA1 = "PLD.JED", PROGDAT = |"\PROG\DAT\"+FILENAME|
```

The values can be text constants (enclosed in " ") or text expressions (enclosed in |). Text expressions can contain only text constants or previously defined parameters.

<*name*>

Cannot be a .PARAM name, or any of the reserved parameters names.

Once defined, a text parameter can be used in the following places:

- To specify a JEDEC file name on a PLD device.
- To specify an Intel Hex file name to program a ROM device or initialize a RAM device.
- To specify a stimulus file name or signal name on an FSTIM device.
- To specify a text parameter to a subcircuit.
- As part of a text expression used in one of the above.

Note *The text parameters and expressions are currently only used by the digital simulation feature.*

<*text expression*> Text expressions can contain the following.

Text Expressions	Definition
enclosed in “ ”	text constants
text parameters	previously defined parameters
“+”	the operator which concatenates two text values
TEXTINT(<*value or expression*>)	a function which returns a text string which is the integer value closest to the value of the <*value or expression*>; (<*value or expression*> is a floating-point value)

.TF (Transfer)

Purpose

The .TF statement causes the small-signal DC gain to be calculated by linearizing the circuit around the bias point.

General Form

.TF <*output variable*> <*input source name*>

Example

```
.TF V(5) VIN
.TF I(VDRIV) ICNTRL
```

The gain from <*input source name*> to <*output variable*> and the input and output resistances are evaluated and written to the output file. This output does not require a .PRINT, .PLOT, or .PROBE statement.

<*output variable*> This has the same format and meaning as in the .PRINT statement.

When <*output variable*> is a current, it is restricted to be the current through a voltage source.

Note *The results of the .TF command are only available in the output file. They cannot be viewed in Probe.*

.TRAN (Transient Analysis)

Purpose

The .TRAN statement causes a transient analysis to be performed on the circuit.

General Form

.TRAN[/OP] <*print step value*> <*final time value*>
+[*no-print value* [*step ceiling value*]][SKIPBP]

Example

```
.TRAN 1ns 100ns
.TRAN/OP 1ns 100ns 20ns SKIPBP
.TRAN 1ns 100ns 0ns .1ns
```

Prior to doing the transient analysis, PSpice computes a bias point for the circuit separate from the regular bias point. This is performed because the independent sources can have different values at the start of a transient analysis than their DC value.

The transient analysis uses an internal time step which is adjusted as the analysis proceeds. Over intervals where there is little activity, the internal time step is increased and during busy intervals it is decreased.

The default ceiling on the internal time step is <*final time value*>/50. When there are no charge storage elements, inductances, or capacitances in the circuit, the ceiling is <*print step value*>.

[/OP] Normally, only the node voltages are printed for the transient analysis bias point. However, the “/OP” suffix (on .TRAN) has the same detailed printing of the bias point that the .OP command has for the regular bias point.

<*print step value*> The time interval used for printing, plotting (.PRINT or .PLOT), or performing a Fourier integral on the results of the transient analysis.

Since the results are computed at different times than they are printed, a 2nd-order polynomial interpolation is used to obtain the printed values. This applies only to .PRINT, .PLOT, and .FOUR outputs and does not affect Probe.

<*final time value*> The transient analysis calculates the circuit’s behavior over time, starting at TIME=0 and going to the <*final time value*>.

The variables TSTEP and TSTOP, which are used in defaulting some waveform parameters, are set by the .TRAN command. TSTEP is <*print step value*> and TSTOP is <*final time value*>.

The transient analysis always starts at TIME=0. However, it is possible to suppress output of a portion of the analysis.

[*no-print value*] The amount of time from TIME=0 which is not printed, plotted, or given to Probe.

[*step ceiling value*] Overrides the default ceiling on the internal times step with a lower value.

SKIPBP When the SKIPBP is put at the end of the .TRAN statement, the calculation of the bias point is skipped. This option means that the bias conditions are fully determined by the IC= specifications for capacitors and inductors.

Comments

Schematics users should refer to the PSpice user's guide for more information on setting initial conditions. HP and Sun users that have circuit file based designs should also refer to the PSpice user's guide for more information on setting initial conditions.

The .PRINT, .PLOT, .FOUR, or .PROBE statements must be used to get the results of the transient analysis.

.VECTOR (Digital Output)

Purpose

The .VECTOR command is used to create files containing digital simulation results.

General Form

.VECTOR <*number of nodes*> <*node*>*
+ [POS = <*column position*>]
+ [FILE = <*filename*>]
+ [RADIX = "Binary" | "Hex" | "Octal"
+ [BIT = <*bit index*>]]
+ [SIGNAMES = <*signal names*>]

Example

```
.VECTOR 1 CLOCK SIGNAMES=SYSCLK
.VECTOR 4 DATA3 DATA2 DATA1 DATA0
.VECTOR 1 ADDR3 POS=2 RADIX=H BIT=4
.VECTOR 1 ADDR2 POS=2 RADIX=H BIT=3
.VECTOR 1 ADDR1 POS=2 RADIX=H BIT=2
.VECTOR 1 ADDR0 POS=2 RADIX=H BIT=1
```

The resultant file contains time and state values for the circuit nodes specified in the statement. The file format is identical to that used by the digital file stimulus device (FSTIM). Thus, the results of one simulation can be used to drive inputs of a subsequent simulation. See File Stimulus on page **3-85** for more information on the file stimulus file format.

<*filename*> By default, the .VECTOR command creates a file named "<*circuit file name*>.vec".

A different file name can be specified by using the FILE parameter which is described below. A multiple .VECTOR command can be used to specify nodes for the same file.

The optional parameters on the .VECTOR command can be used to control the file name, column order, radix of the state values, and signal names which appear in the file header. Each parameter is described in detail in the following table.

<*number of nodes*> This means the number of nodes in the list.

<*node*> This defines the nodes whose states are to be stored.

<*column position*> Specifies the column position in the file. By default, the column position is determined through the order in which the .VECTOR command appears in the circuit file, and by the order of the signals within a .VECTOR command. Valid values for <*column position*> are 1-255.

<*filename*> Specifies the name of the file to which the simulation results are saved. If left blank, the simulator creates a file named "<*circuit filename*>.vec", where "<circuit filename>.cir" is the name of the netlist file.

RADIX The radix of the values for the specified nodes is defined if <*number of nodes*> is greater than one. Valid values are "BINARY," "OCTAL," or "HEX" (abbreviation to the first letter is allowed). If <*number of nodes*> is one, and a radix of OCTAL or HEX is specified, a bit position within the octal or hex digit via the BIT parameter can also be specified. A separate .VECTOR command can be used to construct multi-bit values out of single signals, provided the same POS value is specified. The default radix is BINARY if <*number of nodes*> is one. Otherwise, the default radix is HEX. If a RADIX of OCTAL or HEX is specified, the simulator creates "dummy" entries in the vector file header to fill out the value if <*number of nodes*> is not an even power of two.

<*bit index*> Defines the bit position within a single hex or octal digit when the VECTOR symbol is attached to a wire. Valid values are one through four if RADIX=HEX, and one through three if RADIX=OCTAL.

<*signal names*> Defines the names of the signals which appear in the header of the vector file. If SIGNAMES is not specified, the <*node*> names are used in the vector file header. If <*number of nodes*> is greater than one, names are defined positionally, msb to lsb. If fewer signal names than <*number of nodes*> are specified, the <*node*> names are used for the remaining unspecified names.

.WATCH (Watch Analysis Results)

Purpose

The .WATCH statement allows results from DC, AC, and transient analyses to be an output to the PSpice display in text format while the simulation is running.

General Form

.WATCH [DC][AC][TRAN]
+ [<*output variable*> [<*lower limit value*>,<*upper limit value*>]]*

Example

```
.WATCH DC V(3) (-1V,4V) V(2,3) V(R1)
.WATCH AC VM(2) VP(2) VMC(Q1)
.WATCH TRAN VBE(Q13) (0V,5V) ID(M2) I(VCC) (0,500mA)
.WATCH DC V([RESET]) (2.5V,10V)
```

The first example displays three output variables on the screen. The first variable, V(3), has an operating range set from minus one volt to four volts. If during the simulation the voltage at node three exceeds four volts, the simulation will pause. If the simulation is allowed to proceed, and node three continues to rise in value, the simulation is then not interrupted. However, if the simulation is allowed to continue and V(3) falls below -1 volt, the simulation would again pause because a new boundary condition was exceeded.

Up to three output variables can be seen on the display at one time. More than three variables can be specified, but they are not all displayed.

DC, AC, and TRAN

These are the analysis types whose results can be displayed by using the .WATCH command.

Only one analysis type must be specified per .WATCH command, but there can be a .WATCH command for each analysis type in the circuit.

<*output variable*> A maximum of eight output variables are allowed on a single .WATCH statement.

The possible output variables are given in .PROBE (Probe) on page 1-46, with the exception that digital nodes cannot be used and group delay is not available.

<lower limit value>,<upper limit value>

The optional value range specifies the normal operating range of that particular output variable. If the range is exceeded during the simulation, the simulator beeps and pauses. At this point, the simulation can be aborted or continued. If continued, the check for that output variable's boundary condition is eliminated. Each output variable can have its own value range.

.WCASE (Sensitivity/Worst-Case Analysis)

Purpose

The .WCASE statement causes a sensitivity and worst-case analysis of the circuit to be performed.

General Form

.WCASE <*analysis*> <*output variable*> <*function*> [*option*]*

Example

```
.WCASE TRAN V(5) YMAX
.WCASE DC IC(Q7) YMAX VARY DEV
.WCASE AC VP(13,5) YMAX DEVICES RQ OUTPUT ALL
.WCASE TRAN V([OUT1],[OUT2]) YMAX RANGE(.4u,.6u)
 + LIST OUTPUT ALL VARY DEV HI
```

Multiple runs of the selected analysis (DC, AC, or transient) are performed while parameters are varied. Unlike .MC, .WCASE varies only one parameter per run. This allows PSpice to calculate the sensitivity of the output waveform to each parameter. Once all the sensitivities are known, one final run is performed using all parameters varied so as to produce the worst-case waveform. The sensitivity and worst-case runs are performed using variations on model parameters as specified by the DEV and LOT tolerances on each .MODEL parameter (see .MODEL (Model) command section for details on the DEV and LOT tolerances). Other specifications on the .WCASE command control the output generated by the analysis.

Note *Either .MC or .WCASE can be run, but not both in the same circuit.*

<*analysis*> Only one of DC, AC, or TRAN must be specified for <*analysis*>. This analysis is repeated in subsequent passes of the worst-case analysis. All requested analyses are performed during the nominal pass. Only the selected analysis is performed during subsequent passes.

<*output variable*> Identical in format to that of a .PRINT output variable; see **1-47** for details.

<*function*> Specifies the operation to be performed on the values of the <*output variable*> to reduce these to a single value.

This value is the basis for the comparisons between the nominal and subsequent runs. The <*function*> must be one of the following.

Function	Meaning
YMAX	Find the absolute value of the *greatest difference* in each waveform from the nominal run.
MAX	Find the *maximum value* of each waveform.
MIN	Find the *minimum value* of each waveform.
RISE_EDGE (<*value*>)	Find the *first occurrence* of the waveform crossing *above* the threshold <*value*>. The waveform must have one or more points at or below <*value*> followed by one above; the output value listed is where the waveform increases above <*value*>.
FALL_EDGE (<*value*>)	Find the *first occurrence* of the waveform crossing *below* the threshold <*value*>. The waveform must have one or more points at or above <*value*> followed by one below; the output value listed is where the waveform decreases below <*value*>.

Note *The <function> and all [option]s do not affect the Probe data saved from the simulation. They are only applicable to the output file.*

[*option*]* Could have none or one or more of the following.

[option]	Meaning
LIST	Prints the updated model parameters for the sensitivity analysis. This does not affect the Probe data generated by the simulation.
OUTPUT ALL	Prints output from the sensitivity runs, after the nominal (first) run. The output from any run is governed by the .PRINT, .PLOT, and .PROBE command in the file. If OUTPUT ALL is omitted, then only the nominal and worst-case runs produce output. OUTPUT ALL ensures that all sensitivity information is saved for Probe.
RANGE* (<*low value*>, <*high value*>)	Restricts the range over which <*function*> can be evaluated. An "*" can be used in place of a <*value*> to show "for all values." For example see the next two rows.
YMAX RANGE(*,.5)	YMAX is evaluated for values of the sweep variable (e.g., time, and frequency) of .5 or less.
MAX RANGE(-1,*)	The maximum of the output variable is found for values of the sweep variable of -1 or more.

[option]	Meaning
HI or LOW	Specify the direction which <*function*> should move for the worst-case run is to go (relative to the nominal). If <*function*> is YMAX or MAX, the default is HI, otherwise the default is LOW.
VARY DEV/ VARY LOT VARY BOTH	By default, any device which has a model parameter specifying *either* a DEV tolerance or a LOT tolerance is included in the analysis. The analysis can be limited to only those devices which have DEV or LOT tolerances by specifying the appropriate option. The default is VARY BOTH. When VARY BOTH is used, sensitivity to parameters using both DEV and LOT specifications is checked only with respect to LOT variations. The parameter is then maximized or minimized using both DEV and LOT tolerances for the worst-case. All devices referencing the model have the same parameter values for the worst-case simulation.
DEVICES (*list of device types*	By default, all devices are included in the sensitivity and worst-case analyses. The devices considered can be limited by listing the device types after the keyword DEVICES. Do not use any spaces or tabs in the devices list. For example, to only perform the analysis on resistors and MOSFETs, enter: `DEVICES RM`

* If RANGE is omitted, then <*function*> is evaluated over the whole sweep range. This is equivalent to RANGE(*,*).

* (Comment)

Purpose

A statement beginning with "*" is a comment line and has no effect.

General Form

* [*any text*]

Example

```
* This is an example of a comment
```

The use of comment statements throughout the input is recommended. It is good practice to place a comment just before a subcircuit definition to identify the nodes, for example

```
*                +IN  -IN  V+  V-  +OUT  -OUT
.SUBCKT  OPAMP   100  101  1   2    200   201
```

or to identify major blocks of circuitry.

; (In-line Comment)

Purpose

A ";" is treated as the end of a line.

General Form

circuit file text ;[*any text*]

The simulator moves on to the next line in the circuit file. The text on the line after the ";" is a comment and has no effect. The use of comments throughout the input is recommended. This type of comment can also replace comment lines, which must start with "*" in the first column.

Trailing in-line comments that extend to more that one line can use a semicolon to mark the beginning of the subsequent comment lines, as shown in the example.

Example

```
R13 6 8 10  ; R13 is a
            ; feedback resistor
C3 15 0 .1U ; decouple supply
```

Analog Devices

2

Overview

This chapter describes the analog devices supported by PSpice A/D and PSpice. The following information is provided:

- device type
- format
- usage
- library location

Analog Devices

This chapter describes the different types of analog devices supported by PSpice and PSpice A / D. These device types include analog primitives, independent and controlled sources, and subcircuit calls. Each device type is described separately, and each description includes the following information as applicable:

- A description, and example of, the proper netlist syntax.
- The corresponding model types and their description.
- The corresponding list of model parameters and their descriptions.
- The equivalent circuit diagram and characteristic equations for the model (as required).
- References to publications on which the model is based.

These analog devices include all of the standard circuit components that normally are not considered part of the two-state (binary) devices that are found in the digital devices.

The model library consists of analog models of off-the-shelf parts that can be used directly in circuits that are being developed. Refer to the *Library Reference Manual* for device models and in which library they can be found. The model library includes models implemented using the .MODEL statement and macromodels implemented as subcircuits with the .SUBCKT statement.

This chapter includes a summary table, Table 2-1, which lists all of the analog device primitives supported by the simulator. Each primitive is described in detail in the sections following the table.

Device Types

PSpice supports many types of analog devices, including sources and general subcircuits. PSpice A / D also supports digital devices. The supported devices are categorized into device types. each of which can have one or more model types. For example, the BJT device type has three model types: NPN, PNP, and LPNP (Lateral PNP). The description of each devices type includes a description of any of the model types it supports.

The device declarations in the netlist always begin with the name of the individual device (instance). The first letter of the name determines the device type. What follows the name depends on the device type and its requested characteristics. Table 2-1 summarizes the device types and the general form of their declaration formats.

Note *The "Device Type" column in the table includes the designator (letter) used in the device modeling.*

Table 2-1 *Analog Device Summary*

Device Type	Letter	Declaration Format	Page
Bipolar Transistor	Q	Q<*name*> <*collector node*> <*base node*> <*emitter node*> + [*substrate node*] <*model name*> [*area value*]	2-83
Capacitor	C	C<*name*> <+ *node*> <- *node*> [*model name*] <*value*> + [IC=<*initial value*>]	2-22
Voltage-Controlled Voltage Source	E	E<*name*> <+ *node*> <- *node*> <+ controlling node> + <- *controlling node*> <*gain*> (additional Analog Behavioral Modeling forms: VALUE, TABLE, LAPLACE, FREQ, and CHEBYSHEV; additional POLY form)	2-29
Voltage-Controlled Current Source	G	G<*name*> <+ *node*> <- *node*> <+ *controlling node*> + <- *controlling node*> <*transconductance*> (additional Analog Behavioral Modeling forms: VALUE, TABLE, LAPLACE, FREQ, and CHEBYSHEV; additional POLY form)	2-29
Current-Controlled Current Source	F	F<*name*> <+ *node*> <- *node*> <*controlling V device name*> + <*gain*> (additional POLY form)	2-32
Current-Controlled Switch	W	W<*name*> <+ *switch node*> <- *switch node*> + <*controlling V device name*> <*model name*>	2-104

Table 2-1 *Analog Device Summary (continued)*

Device Type	Letter	Declaration Format	Page
Current-Controlled Voltage Source	H	H<*name*> <+ *node*> <- *node*> <*controlling V device name*> + <*transresistance*> (additional POLY form)	2-32
Digital Input	N	N<*name*> <*interface node*> <*low level node*> <*high level node*> + <*model name*> <*input specification*>	
Digital Output	O	O<*name*> <*interface node*> <*low level node*> <*high level node*> + <*model name*> <*output specification*>	
Digital Primitive*	U	U<*name*> <*primitive type*> ([*parameter value*]*) + <*digital power node*> <*digital ground node*> <*node*>* + <*timing model name*>	
Digital Stimulus*	U STIM	U<*name*> STIM (<*width value*>, <*format value*>) + <*digital power node*> <*digital ground node*> <*node*>* + <*I/O model name*> [TIMESTEP=<*stepsize value*>] + <*waveform description*>	
Diode	D	D<*name*> <*anode node*> <*cathode node*> <*model name*> [*area value*]	2-24
GaAsFET	B	B<*name*> <*drain node*> <*gate node*> <*source node*> + <*model name*> [*area value*]	2-6
Independent Current Source & Stimulus	I	I<*name*> <+ *node*> <- *node*> [[DC] <*value*>] + [AC <*magnitude value*> [*phase value*]] [*transient specification*]	2-33
Independent Voltage Source & Stimulus	V	V<*name*> <+ *node*> <- *node*> [[DC] <*value*>] + [AC <*magnitude value*> [*phase value*]] [*transient specification*]	2-33
Inductor	L	L<*name*> <+ *node*> <- *node*> [*model name*] <*value*> + [IC=<*initial value*>]	2-61
Inductor Coupling	K	K<*name*> L<*inductor name*> <L<*inductor name*>>* + <*coupling value*> K<*name*> <L<*inductor name*>>* <*coupling value*> + <*model name*> [*size value*]	2-50
Insulated Gate Bipolar Transistor (IGBT)	Z	Z<*name*> <*collector*> <*gate*> <*emitter*> <*model name*> + [AREA=<*value*>] [WB=<*value*>] [AGD=<*value*>] + [KP=<*value*>] [TAU=<*value*>]	2-108
JFET	J	J<*name*> <*drain node*> <*gate node*> <*source node*> + <*model name*> [*area value*]	2-44

Table 2-1 *Analog Device Summary (continued)*

Device Type	Letter	Declaration Format	Page
MOSFET	M	M<*name*> <*drain node*> <*gate node*> <*source node*> + <*bulk/substrate node*> <*model name*> + [*common model parameter*]*	2-63
Resistor	R	R<*name*> <+ *node*> <- *node*> [*model name*] <*value*> + [TC=<*linear temp. coefficient*>[,<*quadratic temp. coefficient*]]	2-94
Subcircuit Call	X	X<*name*> [*node*]* <*subcircuit name*> + [PARAMS: <<*name*>=<*value*>>*] [TEXT:<<name>=<text value>>*]	2-107
Transmission Line	T	T<*name*> <*A port + node*> <*A port - node*> + <*B port + node*> <*B port - node*> <*ideal or lossy specification*>	2-99
Transmission Line Coupling	K	K<*name*> T<*line name*> <T<*line name*>>* + CM=<*coupling capacitance*> LM=<*coupling inductance*>	2-50
Voltage-Controlled Switch	S	S<*name*> <+ *switch node*> <- *switch node*> + <+ *controlling node*> <- *controlling node*> <*model name*>	2-96

*The Digital Primitive and Digital Stimulus device types are generic in form. They have flexible syntax, and can refer to numerous different devices. See Chapter 3, *Digital Devices* for details.

GaAsFET

General Form

B<*name*> <*drain node*> <*gate node*> <*source node*>
+ <*model name*> [*area value*]

Examples

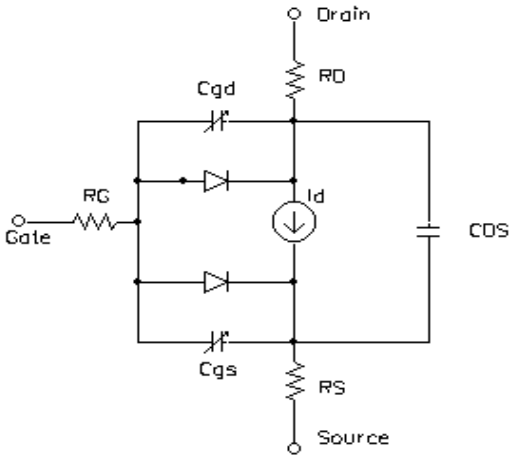
```
BIN 100 10 0  GFAST
B13 22  14 23 GNOM 2.0
```

Model Form

```
.MODEL <model name> GASFET [model parameters]
```

As shown in Figure 2-1, the GaAsFET is modeled as an intrinsic FET using an ohmic resistance (**RD** / *area*) in series with the drain, another ohmic resistance (**RS** / *area*) in series with the source, and another ohmic resistance (**RG**) in series with the gate. The [*area value*] is the relative device area and defaults to 1.

Figure 2-1 *GaAsFET Model*



The **LEVEL** model parameter selects between different models for the intrinsic GaAsFET:

GaAsFET LEVELS	Definition
LEVEL=1	"Curtice" model (see reference [1]).
LEVEL=2	"Raytheon" or "Statz" model (see reference [3]) and is equivalent to the GaAsFET model in SPICE3.
LEVEL=3	"TOM" model by TriQuint (see reference [4]).
LEVEL=4	"Parker-Skellern" model (see reference [5] and [6]).
LEVEL=5	"TOM-2" model by TriQuint (see reference [7].

Note *A support for the improved model of the GaAs MESFET (LEVEL=5) has been added. The TOM-2 model is based on the original TriQuint TOM model. The new model retains the desirable features of the TOM model while improving accuracy in the subthreshold near cutoff, and knee regions (Vds of 1 volt or less). This new model includes additional temperature coefficients that relate to the drain current. It also corrects the major deficiencies in the behavior of the capacitance as a function of temperature. Three "auxiliary" model parameters* **BTRK**, **DVT**, *and* **DVTT** *are added to LEVEL=5 to make the Monte Carlo analysis easier.*

Model Parameters

Table 2-2 *GaAsFET Model Parameters for All Levels*

Model Parameters*	Description	Units	Default
AF	Flicker noise exponent		1
BETA	Transconductance coefficient	amp / volt^2	0.1
BETATCE	**BETA** exponential temperature coefficient	% / °C	0
CDS	Drain-source capacitance	farad	0
CGD	Zero-bias gate-drain p-n capacitance	farad	0
CGS	Zero-bias gate-source p-n capacitance	farad	0
EG	Band gap voltage (barrier height)	eV	1.11
FC	Forward-bias depletion capacitance coefficient		0.5
IS	Gate p-n saturation current	amp	1E-14
KF	Flicker noise coefficient		0
LEVEL	Model index (1, 2, 3, 4, or 5)		1
N	Gate p-n emission coefficient		1
RD	Drain ohmic resistance	ohm	0
RG	Gate ohmic resistance	ohm	0
RS	Source ohmic resistance	ohm	0
TRD1	**RD** temperature coefficient (linear)	°C^{-1}	0
TRG1	**RG** temperature coefficient (linear)	°C^{-1}	0
TRS1	**RS** temperature coefficient (linear)	°C^{-1}	0
T_ABS	Absolute temperature	°C	
T_MEASURED	Measured temperature	°C	
T_REL_GLOBAL	Relative to current temperature	°C	
T_REL_LOCAL	Relative to AKO model temperature	°C	
VBI	Gate p-n potential	volt	1.0

Table 2-2 *GaAsFET Model Parameters for All Levels (continued)*

Model Parameters*	Description	Units	Default
VTO	Pinchoff voltage	volt	-2.5
VTOTC	**VTO** temperature coefficient	volt / °C	0
XTI	**IS** temperature exponent		0

* For information on **T_ABS**, **T_MEASURED**, **T_REL_GLOBAL**, and **T_REL_LOCAL**, see the .MODEL statement.

Table 2-3 *GaAsFET Model Parameters for Level 1*

Model Parameters	Description	Units	Default
ALPHA	Saturation voltage parameter	$volt^{-1}$	2.0
LAMBDA	Channel-length modulation	$volt^{-1}$	0
M	Gate p-n grading coefficient		0.5
TAU	Conduction current delay time	sec	0

Table 2-4 *GaAsFET Model Parameters for Level 2*

Model Parameters	Description	Units	Default
ALPHA	Saturation voltage parameter	$volt^{-1}$	2.0
B	Doping tail extending parameter	$volt^{-1}$	0.3
LAMBDA	Channel-length modulation	$volt^{-1}$	0
M	Gate p-n grading coefficient		0.5
TAU	Conduction current delay time	sec	0
VDELTA	Capacitance transition voltage	volt	0.2
VMAX	Capacitance limiting voltage	volt	0.5

Table 2-5 *GaAsFET Model Parameters for Level 3*

Model Parameters	Description	Units	Default
ALPHA	Saturation voltage parameter	$volt^{-1}$	2.0
BTRK	Auxiliary parameter for Monte Carlo analysis*	$amp/volt^{3}$	0
DELTA	Output feedback parameter	$(amp \cdot volt)^{-1}$	0
DVT	Auxiliary parameter for Monte Carlo analysis*	volt	0
DVTT	Auxiliary parameter for Monte Carlo analysis*	volt	0
GAMMA	Static feedback parameter		0
M	Gate p-n grading coefficient		0.5
Q	Power-law parameter		2
TAU	Conduction current delay time	sec	0
VDELTA	Capacitance transition voltage	volt	0.2
VMAX	Gate diode capacitance limiting voltage	volt	0.5

Table 2-6 *GaAsFET Model Parameters for Level 4*

Model Parameters	Description	Units	Default
ACGAM	Capacitance modulation		0
DELTA	Output feedback parameter	$(amp \cdot volt)^{-1}$	0
HFETA	High-frequency VGS feedback parameter		0
HFE1	**HFGAM** modulation by VGD	$volt^{-1}$	0
HFE2	**HFGAM** modulation by VGS	$volt^{-1}$	0
HFGAM	High-frequency VGD feedback parameter		0
HFG1	**HFGAM** modulation by VSG	$volt^{-1}$	0
HFG2	**HFGAM** modulation by VDG	$volt^{-1}$	0
IBD	Gate junction breakdown current	amp	0
LAMBDA	Channel-length modulation	$volt^{-1}$	0

Table 2-6 *GaAsFET Model Parameters for Level 4 (continued)*

Model Parameters	Description	Units	Default
LFGAM	Low-frequency feedback parameter		0
LFG1	**LFGAM** modulation by VSG	$volt^{-1}$	0
LFG2	**LFGAM** modulation by VDG	$volt^{-1}$	0
MVST	Subthreshold modulation	$volt^{-1}$	0
MXI	Saturation knee-potential modulation		0
P	Linear-region power law exponent		2
Q	Power-law parameter		2
TAUD	Relaxation time for thermal reduction	sec	0
TAUG	Relaxation time for GAM feedback	sec	0
VBD	Gate junction breakdown potential	volt	1
VST	Subthreshold potential	volt	0
XC	Capacitance pinchoff reduction factor		0
XI	Saturation knee potential factor		1000
Z	Knee transition parameter		0.5

Table 2-7 *GaAs FET Model Parameter for Level 5*

Model Parameters	Description	Units	Default
ALPHA	Saturation voltage parameter	$volt^{-1}$	2.0
ALPHATCE	**ALPHA** temperature coefficient	%/°C	0
BTRK	Auxiliary parameter for Monte Carlo analysis*	$amp/volt^3$	0
CGDTCE	**CGD** temperature coefficient	$°C^{-1}$	0
CGSTCE	**CGS** temperature coefficient	$°C^{-1}$	0
DELTA	Output feedback parameter	$(amp \cdot volt)^{-1}$	0
DVT	Auxiliary parameter for Monte Carlo analysis*	volt	0
DVTT	Auxiliary parameter for Monte Carlo analysis*	volt	0
GAMMA	Static feedback parameter		0

Table 2-7 *GaAs FET Model Parameter for Level 5 (continued)*

Model Parameters	Description	Units	Default
GAMMATC	**GAMMA** temperature coefficient	$°C^{-1}$	0
ND	Subthreshold slope drain pull parameter	$volt^{-1}$	0
NG	Subthreshold slope gate parameter		0
Q	Power-law parameter		2
TAU	Conduction current delay time	sec	0
VBITC	**VBI** temperature coefficient	volt / °C	0
VDELTA	Capacitance transition voltage	volt	0.2
VMAX	Gate diode capacitance limiting voltage	volt	0.5

*See Auxiliary model parameters BTRK, DVT, and DVTT

Auxiliary model parameters BTRK, DVT, and DVTT

The parameters **BTRK**, **DVT**, and **DVTT** are "auxiliary" model parameters that are used to make the Monte Carlo analysis easier when using PSpice. In the analysis, these affect the parameters **VTO** and **BETA** as follows:

VTO = **VTO** + **DVT** + **DVTT**

BETA = **BETA** + **BTRK** · (**DVT** + **DVTT**)

In Monte Carlo analysis, DEV tolerances placed on the **DVT** or **DVTT** cause variations in both **VTO** and **BETA**. PSpice does not support correlated DEV variations in Monte Carlo analysis. Without **DVT** and **DVTT**, DEV tolerances placed on **VTO** and **BETA** can result in independent variations, there is a definite correlation between **VTO** and **BETA** on real devices.

The **BTRK**, **DVT**, and **DVTT** parameters are also used to provide tracking between distinct GaAs FETs such as in depletion mode versus enhancement mode. PSpice already provides a limited mechanism for this, but only allows one "DEV" and one "LOT" (or LOT / n and DEV / n) tolerance per model parameter. The added parameters circumvent this restriction by extending the capability of Monte Carlo to model correlation between the critical model parameters.

Equations

In the following equations:

Vgs = intrinsic gate-intrinsic source voltage
Vgd = intrinsic gate-intrinsic drain voltage
Vds = intrinsic drain-intrinsic source voltage
Vt = $k \cdot T / q$ (thermal voltage)
k = Boltzmann constant
q = electron charge
T = analysis temperature (°K)
Tnom = nominal temperature (set using .OPTIONS **TNOM**=)

These equations describe an N-channel GaAsFET.

Positive current is current flowing into a terminal (for example, positive drain current flows from the drain through the channel to the source).

DC Currents [1]

Ig = gate current = *area*·(Igs+Igd)

Igs = gate-source leakage current
Igd = gate-drain leakage current

LEVEL=1, 2, 3, or 5:

$$Igs = \mathbf{IS}\cdot(e^{Vgs/(\mathbf{N}\cdot Vt)}-1)$$
$$Igd = \mathbf{IS}\cdot(e^{Vgd/(\mathbf{N}\cdot Vt)}-1)$$

For **LEVEL**=4:

$$Igs = Igs_f + Igs_r$$

where

$$Igs_f = \mathbf{IS}\cdot\left[e^{\frac{Vgs}{\mathbf{N}\cdot V_t}}-1\right] + Vgs\cdot GMIN$$

and

$$Igs_r = \mathbf{IBD}\cdot\left[1-e^{-\frac{Vgs}{\mathbf{VBD}}}\right]$$

$$Igd = Igd_f + Igd_r$$

where

$$Igd_f = \mathbf{IS}\cdot\left[e^{\frac{Vgd}{\mathbf{N}\cdot V_t}}-1\right] + Vgd\cdot GMIN$$

and

$$Igd_r = \mathbf{IBD}\cdot\left[1-e^{-\frac{Vgd}{\mathbf{VBD}}}\right]$$

Id = drain current = *area*·(Idrain-Igd)

Is = source current = *area*·(-Idrain-Igs)

1. Positive current is current flowing into a terminal

Equations for Idrain: LEVEL=1

For: $Vds \geq 0$ (normal mode)
and: $Vgs - VTO < 0$ (cutoff region)
$Idrain = 0$
and: $Vgs - VTO \geq 0$ (linear & saturation region)
$Idrain = BETA \cdot (1+LAMBDA \cdot Vds) \cdot (Vgs-VTO)^2 \cdot tanh(ALPHA \cdot Vds)$

For: $Vds < 0$ (inverted mode)
Switch the source and drain in equations (above).

Equations for Idrain: LEVEL=2

For: $Vds \geq 0$ (normal mode)
and: $Vgs - VTO < 0$ (cutoff region)
$Idrain = 0$
and: $Vgs - VTO \geq 0$ (linear & saturation region)
$Idrain = BETA \cdot (1+LAMBDA \cdot Vds) \cdot (Vgs-VTO)^2 \cdot Kt/(1+B \cdot (Vgs-VTO))$
where Kt (a polynomial approximation of *tanh*) is:
for: $0 < Vds < 3/ALPHA$ (linear region)
$Kt = 1 - (1 - Vds \cdot ALPHA/3)^3$
for: $Vds \geq 3/ALPHA$ (saturation region)
$Kt = 1$

For: $Vds < 0$ (inverted mode)
Switch the source and drain in equations (above).

Equations for Idrain: LEVEL=3

For: $Vds \geq 0$ (normal mode)
and: $Vgs - V_{to} < 0$ (cutoff region)
$Idrain = 0$
and: $Vgs - V_{to} \geq 0$ (linear & saturation region)
$Idrain = Idso/(1 + DELTA \cdot Vds \cdot Idso)$
where $Idso = BETA \cdot (Vgs-V_{to})^Q \cdot Kt$
where
$V_{to} = VTO - GAMMA \cdot Vds$
Kt is the same as for LEVEL=2

For: $Vds < 0$ (inverted mode)
Switch the source and drain in equations (above).

Equations for Idrain: LEVEL=4

For Vds $\geq$ 0, then:

$$\text{Idrain} = \frac{\text{Ids}}{1 + \textbf{DELTA} \cdot p_{avg}}$$

$$\text{Vgst} = \text{Vgs} - \textbf{VTO} - \gamma_{lf} \cdot \text{Vgd}_{avg} - \gamma_{hf} \cdot (\text{Vgd} - \text{Vgd}_{avg}) - \eta_{hf} \cdot (\text{Vgs} - \text{Vgs}_{avg})$$

$$\text{Vdst} = \text{Vds}$$

For 0 < Vds, then:

$$\text{Idrain} = \frac{-\text{Ids}}{1 + \textbf{DELTA} \cdot p_{avg}}$$

$$\text{Vgst} = \text{Vgd} - \textbf{VTO} - \gamma_{lf} \cdot \text{Vgd}_{avg} - \gamma_{hf} \cdot (\text{Vgs} - \text{Vgd}_{avg}) - \eta_{hf} \cdot (\text{Vgd} - \text{Vgs}_{avg})$$

$$\text{Vdst} = -\text{Vds}$$

where:

$$\text{Ids} = \textbf{BETA} \cdot (1 + \textbf{LAMBDA} \cdot \text{Vdst}) \cdot (\text{Vgt}^{\textbf{Q}} - (\text{Vgt} - \text{Vdt})^{\textbf{Q}})$$

$$\gamma_{lf} = \textbf{LFGAM} - \textbf{LFG1} \cdot \text{Vgs}_{avg} - \textbf{LFG2} \cdot \text{Vgd}_{avg}$$
$$\gamma_{hf} = \textbf{HFGAM} - \textbf{HFG1} \cdot \text{Vgs}_{avg} - \textbf{HFG2} \cdot \text{Vgd}_{avg}$$
$$\eta_{hf} = \textbf{HFETA} + \textbf{HFE1} \cdot \text{Vgd}_{avg} + \textbf{HFE2} \cdot \text{Vgs}_{avg}$$

$$\text{Vdt} = \frac{1}{2} \cdot \sqrt{(\text{Vdp} \cdot \sqrt{1 + \textbf{Z}} + \text{Vsat})^2 + \textbf{Z} \cdot \text{Vsat}^2} - \frac{1}{2} \cdot \sqrt{(\text{Vdp} \cdot \sqrt{1 + \textbf{Z}} - \text{Vsat})^2 + \textbf{Z} \cdot \text{Vsat}^2}$$

$$\text{Vdp} = \text{Vdst} \cdot \frac{\textbf{P}}{\textbf{Q}} \cdot \left(\frac{\text{Vgt}}{\textbf{VBI} - \textbf{VTO}}\right)^{\textbf{P} - \textbf{Q}}$$

$$\text{Vsat} = \frac{\text{Vgt} \cdot (\text{Vgt} \cdot \textbf{MXI} + \textbf{XI} \cdot (\textbf{VBI} - \textbf{VTO}))}{\text{Vgt} + \text{Vgt} \cdot \textbf{MXI} + \textbf{XI} \cdot (\textbf{VBI} - \textbf{VTO})}$$

$$\text{Vgt} = \textbf{VST} \cdot (1 + \textbf{MVST} \cdot \text{Vdst}) \cdot \ln\left(\exp\left(\frac{\text{Vgst}}{\textbf{VST} \cdot (1 + \textbf{MVST} \cdot \text{Vdst})}\right) + 1\right)$$

and,

$$\text{Vgd}_{avg} = \begin{cases} \text{Vgd} - \textbf{TAUG} \cdot d/dt\ \text{Vgd}_{avg} & \text{if Vgd} \leq \text{Vgs} \\ \text{Vgs} - \textbf{TAUG} \cdot d/dt\ \text{Vgd}_{avg} & \text{if Vgs} < \text{Vgd} \end{cases}$$

$$\text{Vgs}_{avg} = \begin{cases} \text{Vgs} - \textbf{TAUG} \cdot d/dt\ \text{Vgs}_{avg} & \text{if Vgd} \leq \text{Vgs} \\ \text{Vgd} - \textbf{TAUG} \cdot d/dt\ \text{Vgs}_{avg} & \text{if Vgs} < \text{Vgd} \end{cases}$$

$$P_{avg} = \text{Vds} \cdot \text{Ids} - \textbf{TAUD} \cdot d/dt\ P_{avg}$$

Equations for Idrain: LEVEL=5

For: $Vds \geq 0$ (normal mode)

and: $Vgs - \mathbf{VTO} + \mathbf{GAMMA} \cdot Vds \leq 0$ and $\mathbf{NG} + \mathbf{ND} \cdot Vds = 0$ (cutoff region)

$Idrain = 0$

and: $Vgs - \mathbf{VTO} + \mathbf{GAMMA} \cdot Vds > 0$ or $\mathbf{NG} + \mathbf{ND} \cdot Vds \neq 0$

(linear and saturation region)

$Idrain = Idso / (1 + \mathbf{DELTA} \cdot Vds \cdot Idso)$

where $Idso = \mathbf{BETA} \cdot (Vg)^{\mathbf{Q}} \cdot \dfrac{\mathbf{ALPHA} \cdot Vds}{\sqrt{1 + (\mathbf{ALPHA} \cdot Vds)^2}}$

and $Vg = \mathbf{Q} \cdot V_{st} \cdot \log\left(\exp\left(\dfrac{Vgs - (\mathbf{VTO} + \mathbf{GAMMA} \cdot Vds)}{\mathbf{Q} \cdot V_{st}}\right) + 1\right)$

$$V_{st} = (\mathbf{NG} + \mathbf{ND} \cdot Vds) \cdot \left(\frac{kT}{q}\right)$$

For: $Vds < 0$ (inverted mode)

Switch the source and drain in the above equations.

Capacitance[1]

Cds = drain-source capacitance = $area \cdot \mathbf{CDS}$

Equations for Cgs and Cgd: LEVEL=1

Cgs = gate-source capacitance

For: $Vgs \leq \mathbf{FC} \cdot \mathbf{VBI}$

$$Cgs = area \cdot \mathbf{CGS} \cdot (1 - Vgs/\mathbf{VBI})^{-\mathbf{M}}$$

For: $Vgs > \mathbf{FC} \cdot \mathbf{VBI}$

$$Cgs = area \cdot \mathbf{CGS} \cdot (1-\mathbf{FC})^{-(1+\mathbf{M})} \cdot (1-\mathbf{FC} \cdot (1+\mathbf{M}) + \mathbf{M} \cdot Vgs/\mathbf{VBI})$$

Cgd = gate-drain capacitance

For: $Vgd \leq \mathbf{FC} \cdot \mathbf{VBI}$

$$Cgd = area \cdot \mathbf{CGD} \cdot (1 - Vgd/\mathbf{VBI})^{-\mathbf{M}}$$

For: $Vgd > \mathbf{FC} \cdot \mathbf{VBI}$

$$Cgd = area \cdot \mathbf{CGD} \cdot (1-\mathbf{FC})^{-(1+\mathbf{M})} \cdot (1-\mathbf{FC} \cdot (1+\mathbf{M}) + \mathbf{M} \cdot Vgd/\mathbf{VBI})$$

Equations for Cgs and Cgd: LEVEL=2, 3, and 5

Cgs = gate-source capacitance = $area \cdot (\mathbf{CGS} \cdot K2 \cdot K1/(1 - Vn/\mathbf{VBI})^{1/2} + \mathbf{CGD} \cdot K3)$

Cgd = gate-drain capacitance = $area \cdot (\mathbf{CGS} \cdot K3 \cdot K1/(1 - Vn/\mathbf{VBI})^{1/2} + \mathbf{CGD} \cdot K2)$

where

$$K1 = (1 + (Ve - \mathbf{VTO})/((Ve - \mathbf{VTO})^2 + \mathbf{VDELTA}^2)^{1/2})/2$$

$$K2 = (1 + (Vgs - Vgd)/((Vgs - Vgd)^2 + (1/\mathbf{ALPHA})^2)^{1/2})/2$$

$$K3 = (1 - (Vgs - Vgd)/((Vgs - Vgd)^2 + (1/\mathbf{ALPHA})^2)^{1/2})/2$$

$$Ve = (Vgs + Vgd + ((Vgs - Vgd)^2 + (1/\mathbf{ALPHA})^2)^{1/2})/2$$

If: $(Ve + \mathbf{VTO} + ((Ve - \mathbf{VTO})^2 + \mathbf{VDELTA}^2)^{1/2})/2 < \mathbf{VMAX}$

$$Vn = (Ve + \mathbf{VTO} + ((Ve - \mathbf{VTO})^2 + \mathbf{VDELTA}^2)^{1/2})/2$$

else: $Vn = \mathbf{VMAX}$

1. All capacitances are between terminals of the intrinsic GaAsFET (that is, to the inside of the ohmic drain, source, and gate resistances).

Equations for Cgs and Cgd: LEVEL=4

Charge storage is implemented using a modified Statz model.

Cgs = gate-source capacitance

$$Cgs = \frac{1}{2}\cdot K1\cdot\left(1+2\mathbf{ACGAM}+\frac{Vds}{\sqrt{Vds^2+\alpha^2}}\right)+\frac{1}{2}\cdot\mathbf{CGD}\cdot area\cdot\left(1+2\mathbf{ACGAM}-\frac{Vds}{\sqrt{Vds^2+\alpha^2}}\right)$$

Cgd = gate-drain capacitance

$$Cgd = \frac{1}{2}\cdot K1\cdot\left(1-2\mathbf{ACGAM}-\frac{Vds}{\sqrt{Vds^2+\alpha^2}}\right)+\frac{1}{2}\cdot\mathbf{CGD}\cdot area\cdot\left(1-2\mathbf{ACGAM}+\frac{Vds}{\sqrt{Vds^2+\alpha^2}}\right)$$

where:

$$K1 = \frac{1}{2}\frac{\mathbf{CGS}}{\sqrt{1-V_{ge}/\mathbf{VBI}}}\left[1+\mathbf{XC}+(1-\mathbf{XC})\frac{V_{gn}}{\sqrt{V_{gn}^2+0.2^2}}\right]$$

$$V_{ge} = \begin{cases} V_x & \text{if } Vx < \mathbf{FC}\cdot\mathbf{VBI} \\ \mathbf{VBI}\left[1-\dfrac{4(1-\mathbf{FC})^3}{\left(2-3\mathbf{FC}+\dfrac{V_x}{\mathbf{VBI}}\right)^2}\right] & \text{if } Vx \geq \mathbf{FC}\cdot\mathbf{VBI} \end{cases}$$

$$V_x = Vgs+\mathbf{ACGAM}\cdot Vds-\frac{1}{2}(V_{gn}-\sqrt{V_{gn}^2+0.2^2})-\frac{1}{2}(V_{gn}-\sqrt{V_{gn}^2+0.2^2})$$

$$V_{gn} = \left[(Vgs+\mathbf{ACGAM})\cdot Vds-\mathbf{VTO}-\frac{1}{2}(Vds-\sqrt{Vds^2+\alpha^2})\right]\cdot(1-\mathbf{XC})$$

and,

$$\alpha = \frac{\mathbf{XI}}{\mathbf{XI}+1}\cdot\frac{\mathbf{VBI}-\mathbf{VTO}}{2}$$

If the source and drain potentials swap, the model reverses over a range set by α. The model maintains a straight line relation between gate-source capacitance and gate bias in the region Vgs > **FC** ·**VBI**.

Temperature Effects

For all levels:

$$\mathbf{VTO}(T) = \mathbf{VTO} + \mathbf{VTOTC} \cdot (T - Tnom)$$

$$\mathbf{BETA}(T) = \mathbf{BETA} \cdot 1.01^{\mathbf{BETATCE} \cdot (T - Tnom)}$$

$$\mathbf{IS}(T) = \mathbf{IS} \cdot e^{(T/Tnom - 1) \cdot \mathbf{EG}/(\mathbf{N} \cdot Vt)} \cdot (T/Tnom)^{\mathbf{XTI}/\mathbf{N}}$$

$$\mathbf{RG}(T) = \mathbf{RG} \cdot (1 + \mathbf{TRG1} \cdot (T - Tnom))$$

$$\mathbf{RD}(T) = \mathbf{RD} \cdot (1 + \mathbf{TRD1} \cdot (T - Tnom))$$

$$\mathbf{RS}(T) = \mathbf{RS} \cdot (1 + \mathbf{TRS1} \cdot (T - Tnom))$$

The following are specific to **LEVEL**=1, 2, 3, and 4:

$$\mathbf{VBI}(T) = \mathbf{VBI} \cdot T/Tnom - 3 \cdot Vt \cdot ln(T/Tnom) - \mathbf{EG}(Tnom) \cdot T/Tnom + \mathbf{EG}(T)$$

where $\mathbf{EG}(T)$ = silicon bandgap energy = $1.16 - .000702 \cdot T^2/(T+1108)$

$$\mathbf{CGS}(T) = \mathbf{CGS} \cdot (1 + \mathbf{M} \cdot (.0004 \cdot (T - Tnom) + (1 - \mathbf{VBI}(T)/\mathbf{VBI})))$$

$$\mathbf{CGD}(T) = \mathbf{CGD} \cdot (1 + \mathbf{M} \cdot (.0004 \cdot (T - Tnom) + (1 - \mathbf{VBI}(T)/\mathbf{VBI})))$$

The following are specific to **LEVEL**=5:

$$\mathbf{ALPHA}(T) = \mathbf{ALPHA} \cdot 1.01^{\mathbf{ALPHATCE} \cdot (T - Tnom)}$$

$$\mathbf{GAMMA}(T) = \mathbf{GAMMA} + \mathbf{GAMMATC} \cdot (T - Tnom)$$

$$\mathbf{VBI}(T) = \mathbf{VBI} + \mathbf{VBITC} \cdot (T - Tnom)$$

$$\mathbf{VMAX}(T) = \mathbf{VMAX} + \mathbf{VBITC} \cdot (T - Tnom)$$

$$\mathbf{CGS}(T) = \mathbf{CGS} \cdot (1 + \mathbf{CGSTCE} \cdot (T - Tnom))$$

$$\mathbf{CGD}(T) = \mathbf{CGD} \cdot (1 + \mathbf{CGDTCE} \cdot (T - Tnom))$$

Noise

Noise is calculated assuming a one hertz bandwidth, using the following spectral power densities (per unit bandwidth):

the parasitic resistances, **RS**, **RD**, and **RG** generate thermal noise ...

$Is^2 = 4 \cdot k \cdot T/(\mathbf{RS}/area)$
$Id^2 = 4 \cdot k \cdot T/(\mathbf{RD}/area)$
$Ig^2 = 4 \cdot k \cdot T/\mathbf{RG}$

the intrinsic GaAsFET generates shot and flicker noise ...

$Id^2 = 4 \cdot k \cdot T \cdot gm \cdot 2/3 + \mathbf{KF} \cdot Id^{\mathbf{AF}}/FREQUENCY$

where $gm = dIdrain/dVgs$ (at the DC bias point)

References

For more information on this GaAsFET model, refer to:

[1] W. R. Curtice, “A MESFET model for use in the design of GaAs integrated circuits,” *IEEE Transactions on Microwave Theory and Techniques*, MTT-28, 448-456 (1980).

[2] S. E. Sussman-Fort, S. Narasimhan, and K. Mayaram, “A complete GaAs MESFET computer model for SPICE,” *IEEE Transactions on Microwave Theory and Techniques*, MTT-32, 471-473 (1984).

[3] H. Statz, P. Newman, I. W. Smith, R. A. Pucel, and H. A. Haus, “GaAs FET Device and Circuit Simulation in SPICE,” *IEEE Transactions on Electron Devices*, **ED-34**, 160-169 (1987).

[4] A. J. McCamant, G. D. McCormack, and D. H. Smith, “An Improved GaAs MESFET Model for SPICE,” *IEEE Transactions on Microwave Theory and Techniques*, June 1990 (est).

[5] A. E. Parker and D. J. Skellern “Improved MESFET Characterization for Analog Circuit Design and Analysis,” 1992 I*EEE GaAs IC Symposium Technical Digest*, pp. 225-228, Miami Beach, October 4-7, 1992.

[6] A. E. Parker, “Device Characterization and Circuit Design for High Performance Microwave Applications,” IEE EEDMO’93, London, October 18, 1993.

[7] D. H. Smith, “An Improved Model for GaAs MESFETs,” Publication forthcoming. (Copies available from TriQuint Semiconductors Corporation or MicroSim.)

Capacitor

General Form

C<*name*> <(+) *node*> <(-) *node*> [*model name*] <*value*>
+ [IC=<*initial value*>]

Examples

```
CLOAD 15 0 20pF
C2 1 2 .2E-12 IC=1.5V
CFDBCK  3 33 CMOD 10pF
```

Model Form

.MODEL <*model name*> CAP [*model parameters*]

Table 2-8 *Capacitor Model Parameters*

Model Parameters*	Description	Units	Default
C	Capacitance multiplier		1
TC1	Linear temperature coefficient	$°C^{-1}$	0
TC2	Quadratic temperature coefficient	$°C^{-2}$	0
T_ABS	Absolute temperature	°C	
T_MEASURED	Measured temperature	°C	
T_REL_GLOBAL	Relative to current temperature	°C	
T_REL_LOCAL	Relative to AKO model temperature	°C	
VC1	Linear voltage coefficient	$volt^{-1}$	0
VC2	Quadratic voltage coefficient	$volt^{-2}$	0

* For information on **T_MEASURED**, **T_ABS**, **T_REL_GLOBAL**, and **T_REL_LOCAL**, see the .MODEL statement.

(+) and (-) *nodes* Define the polarity when the capacitor has a positive voltage across it. The first node listed (or pin one in Schematics), is defined as positive. The voltage across the component is therefore defined as the first node voltage less the second node voltage.

Positive current flows from the (+) node through the capacitor to the (-) node. Current flow from the first node through the component to the second node is considered positive.

[*model name*] If [*model name*]is left out then <*value*> is the *capacitance* in farads. If [*model name*] is specified, then the capacitance is given by the formula

$$<value>\cdot C\cdot(1+\mathbf{VC1}\cdot V+\mathbf{VC2}\cdot V^2)\cdot(1+\mathbf{TC1}\cdot(T-Tnom)+\mathbf{TC2}\cdot(T-Tnom)^2)$$

where <*value*> is normally positive (though it can be negative, but *not* zero). “Tnom” is the nominal temperature (set using TNOM option).

<*initial value*> The initial voltage across the capacitor during the bias point calculation. It can also be specified in a circuit file using a .IC command as follows:

.IC V(+*node*, -*node*) <*initial value*>

For details on using the .IC command in a circuit file, see page 1-16 of this manual, and refer to your PSpice user’s guide, for more information.

The initial voltage across the capacitor can also be set in Schematics by using the IC1 symbol if the capacitor is connected to ground, or the IC2 symbol for setting the initial conditions between two nodes. These symbols can be found in “special.slb”.

If you are using Schematics, for more information about setting initial conditions, refer to your Schematic user’s guide. If you are using PSpice, refer to your PSpice user’s guide for more information on setting initial conditions.

Noise

The capacitor does not have a noise model.

Diode

General Form D<*name*> <(+) *node*> <(-) *node*> <*model name*> [*area value*]

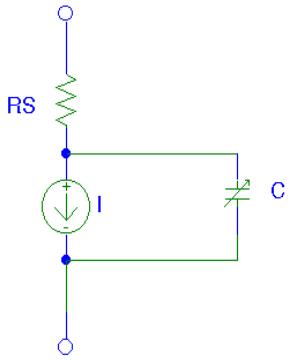
Examples

```
DCLAMP 14  0 DMOD
D13 15 17 SWITCH 1.5
```

Model Form `.MODEL <model name> D [model parameters]`

Figure 2-2 *Diode Model*

l



As shown, the diode is modeled as an ohmic resistance (**RS** / *area*) in series with an intrinsic diode. The <(+) *node*> is the anode and <(-) *node*> is the cathode. Positive current is current flowing from the anode through the diode to the cathode. The [*area value*] scales **IS**, **ISR**, **IKF**,**RS**, **CJO**, and **IBV**, and defaults to 1. **IBV** and **BV** are both specified as positive values.

Model Parameters

Table 2-9 *Diode Model Parameters*

Model Parameters*	Description	Unit	Default
AF	Flicker noise exponent		1
BV	Reverse breakdown “knee” voltage	volt	infinite
CJO	Zero-bias *p-n* capacitance	farad	0
EG	Bandgap voltage (barrier height)	eV	1.11
FC	Forward-bias depletion capacitance coefficient		0.5
IBVL	Low-level reverse breakdown “knee” current	amp	0
IBV	Reverse breakdown “knee” current	amp	1E-10
IKF	High-injection “knee” current	amp	infinite
IS	Saturation current	amp	1E-14
ISR	Recombination current parameter	amp	0
KF	Flicker noise coefficient		0
M	*p-n* grading coefficient		0.5
N	Emission coefficient		1
NBV	Reverse breakdown ideality factor		1
NBVL	Low-level reverse breakdown ideality factor		1
NR	Emission coefficient for ISR		2
RS	Parasitic resistance	ohm	0
TBV1	BV temperature coefficient (linear)	$°C^{-1}$	0
TBV2	BV temperature coefficient (quadratic)	$°C^{-2}$	0
TIKF	IKF temperature coefficient (linear)	$°C^{-1}$	0
TRS1	RS temperature coefficient (linear)	$°C^{-1}$	0
TRS2	RS temperature coefficient (quadratic)	$°C^{-2}$	0
TT	Transit time	sec	0
T_ABS	Absolute temperature	°C	
T_MEASURED	Measured temperature	°C	

Table 2-9 *Diode Model Parameters (continued)*

Model Parameters*	Description	Unit	Default
T_REL_GLOBAL	Relative to current temperature	°C	
T_REL_LOCAL	Relative to AKO model temperature	°C	
VJ	*p-n* potential	volt	1
XTI	IS temperature exponent		3

* For information on **T_MEASURED**, **T_ABS**, **T_REL_GLOBAL**, and **T_REL_LOCAL**, see the .MODEL statement.

Equations

In the following equations:

Vd = voltage across the intrinsic diode only
Vt = $k \cdot T / q$ (thermal voltage)
k = Boltzmann's constant
q = electron charge
T = analysis temperature (°K)
Tnom = nominal temperature (set using TNOM option)

Other variables are from the model parameter list.

DC Current

Id = *area*·(Ifwd - Irev)

Ifwd = forward current = Inrm·Kinj + Irec·Kgen

Inrm = normal current = IS·($e^{Vd/(N\cdot Vt)}$-1)

Kinj = high-injection factor

For: **IKF** > 0

Kinj = (**IKF**/(**IKF**+Inrm))$^{1/2}$

otherwise

Kinj = 1

Irec = recombination current = **ISR**·($e^{Vd/(NR\cdot Vt)}$-1)

Kgen = generation factor = ((1-Vd/VJ)^2+0.005)$^{M/2}$

Irev = reverse current = $Irev_{high}$ + $Irev_{low}$

$Irev_{high}$ = **IBV**·$e^{-(Vd+BV)/(NBV\cdot Vt)}$

$Irev_{low}$ = **IBVL**·$e^{-(Vd+BV)/(NBVL\cdot Vt)}$

Capacitance

Cd = Ct + *area*·Cj

Ct = transit time capacitance = **TT**·Gd

where Gd = DC conductance = area · $\frac{d(Inrm \cdot Kinj + Irec \cdot Kgen)}{dVd}$

Cj = junction capacitance

For: Vd ≤ **FC**·**VJ**

Cj = **CJO**·(1-Vd/VJ)$^{-M}$

For: Vd > FC·VJ

Cj = **CJO**·(1-**FC**)$^{-(1+M)}$·(1-**FC**·(1+**M**)+**M**·Vd/**VJ**)

Temperature Effects

IS(T) = **IS**·$e^{(T/Tnom-1)\cdot EG/(N\cdot Vt)}$·$(T/Tnom)^{XTI/N}$

ISR(T) = **ISR**·$e^{(T/Tnom-1)\cdot EG/(NR\cdot Vt)}$·$(T/Tnom)^{XTI/NR}$

IKF(T) = **IKF**·(1 + **TIKF**·(T-Tnom))

BV(T) = **BV**·(1 + **TBV1**·(T-Tnom) + **TBV2**·$(T-Tnom)^2$)

RS(T) = **RS**·(1 + **TRS1**·(T-Tnom) + **TRS2**·$(T-Tnom)^2$)

VJ(T) = **VJ**·T/Tnom - 3·Vt·*ln*(T/Tnom) - Eg(Tnom)·T/Tnom + Eg(T)
where Eg(T) = silicon bandgap energy = 1.16 - .000702·T^2/(T+1108)

CJO(T) = **CJO**·(1 + **M**·(.0004·(T-Tnom)+(1-**VJ**(T)/**VJ**)))

Noise

Noise is calculated assuming a one hertz bandwidth, using the following spectral power densities (per unit bandwidth):

the parasitic resistance, RS, generates thermal noise ...
$In^2 = 4\cdot k\cdot T/(\mathbf{RS}/area)$

the intrinsic diode generates shot and flicker noise ...
$In^2 = 2\cdot q\cdot Id + \mathbf{KF}\cdot Id^{AF}/FREQUENCY$

References

For a detailed description of *p-n* junction physics refer to:

[1] A. S. Grove, *Physics and Technology of Semiconductor Devices*, John Wiley and Sons, Inc., 1967.

Also, for a generally detailed discussion of the U.C. Berkeley SPICE models, including the diode device, refer to:

[2] P. Antognetti and G. Massobrio, *Semiconductor Device Modeling with SPICE*, McGraw-Hill, 1988.

Voltage-Controlled Voltage Source and Voltage-Controlled Current Source

Note *The Voltage-Controlled Voltage Source (E) and the Voltage-Controlled Current Source (G) devices have the same syntax. For a Voltage-Controlled Current Source just substitute a "G" for the "E". The "G" device generates a current, whereas, the "E" device generates a voltage.*

General Form

E<*name*> <(+) *node*> <(-) *node*> <(+) *controlling node*> <(-) *controlling node*> <*gain*>

E<*name*> <(+) *node*> <(-) *node*> POLY(<*value*>)
+ < <(+) *controlling node*> <(-) *controlling node*> >*
+ < <*polynomial coefficient value*> >*

E<*name*> <(+) <*node*> <(-) *node*> VALUE = { <*expression*> }

E<*name*> <(+) <*node*> <(-) *node*> TABLE { <*expression*> } =
+ < <*input value*>,<*output value*> >*

E<*name*> <(+) *node*> <(-) *node*> LAPLACE { <*expression*> } =
+ { <*transform*> }

E<*name*> <(+) *node*> <(-) *node*> FREQ { <*expression*> } = [KEYWORD]
+ < <*frequency value*>,<*magnitude value*>,<*phase value*> >*
+ [DELAY = <*delay value*>]

E<*name*> <(+) *node*> <(-) *node*> CHEBYSHEV { <*expression*> } =
+ <[LP] [HP] [BP] [BR]>,<*cutoff frequencies*>*,<*attenuation*>*

Examples

```
EBUFF 1 2 10 11 1.0
EAMP 13 0 POLY(1) 26  0 0 500
ENONLIN 100 101 POLY(2) 3 0  4 0  0.0 13.6 0.2 0.005
ESQROOT   5   0 VALUE = {5V*SQRT(V(3,2))}
ET2 2 0 TABLE {V(ANODE,CATHODE)} = (0,0) (30,1)
ERC 5 0 LAPLACE {V(10)} = {1/(1+.001*s)}
ELOWPASS 5 0 FREQ {V(10)}=(0,0,0)(5kHz, 0,0)(6kHz -60, 0)
DELAY=3.2ms
ELOWPASS 5 0 CHEBYSHEV {V(10)} = LP 800 1.2K .1dB 50dB

GBUFF 1 2 10 11  1.0
GAMP 13 0 POLY(1) 26  0 0 500
GNONLIN 100 101 POLY(2) 3 0 4 0 0.0 13.6 0.2 0.005
GPSK 11 6 VALUE = {5MA*SIN(6.28*10kHz*TIME+V(3))}
GT ANODE CATHODE VALUE = {200E-6*PWR(V(1)*V(2),1.5)}
GLOSSY 5 0 LAPLACE {V(10)} = {exp(-sqrt(C*s*(R+L*s)))}
```

The first form and the first two examples apply to the linear case. The second form and the third example are for the nonlinear case.

POLY(<*value*>) Specifies the number of dimensions of the polynomial. The number of pairs of controlling nodes must be equal to the number of dimensions.

The last five forms and examples are analog behavioral modeling (ABM) that have: expression, look up table, Laplace transform, frequency response, and filtering. Refer to your PSpice user's guide for information on the analog behavioral modeling.

(+) and (-) nodes Output nodes. Positive current flows from the (+) node through the source to the (-) node.

The <(+) *controlling node*> and <(-) *controlling node*> are in pairs and define a set of controlling voltages. A particular node can appear more than once, and the output and controlling nodes need not be different. The TABLE form has a maximum size of 2048 input / output value pairs.

FREQ If a DELAY value is specified, the simulator modifies the phases in the FREQ table to incorporate the specified delay value. This is useful for cases of tables which the simulator identifies as being non-causal. When this occurs, the simulator provides a delay value necessary to make the table causal. The new syntax allows this value to be specified in subsequent simulation runs, without requiring the user to modify the table.

If a KEYWORD is specified for FREQ tables, it alters the values in the table. The KEYWORD can be one of the following:

MAG	Causes magnitude of frequency response to be interpreted as a raw value instead of dB
DB	causes magnitude to be interpreted as dB (the default)
RAD	causes phase to be interpreted in radians
DEG	causes phase to be interpreted in degrees (the default)
R_I	causes magnitude and phase values to be interpreted as real and imaginary magnitudes

Chebyshev filters have two attenuation values, given in dB, which specify the pass band ripple and the stop band attenuation. They can be given in either order, but must appear after all of the cutoff frequencies have been given. Low pass (LP) and high pass (HP) have two cutoff frequencies, specifying the pass band and stop band edges, while band pass (BP) and band reject (BR) filters have four. Again, these can be given in any order.

Note *A listing of the filter Laplace coefficients can be obtained for each stage by turning on the LIST option in the* Analysis/Setup/Options *dialog box in Schematics. The output is written to the ".out" file after the simulation is complete.*

For the linear case, there are two controlling nodes and these are followed by the gain. For all cases, including the nonlinear case (POLY), refer to your PSpice user's guide.

Expressions **cannot** be used for linear and polynomial coefficient values in a voltage-controlled voltage source device statement.

Current-Controlled Current Source and Current-Controlled Voltage Source

Note *The Current-Controlled Current Source (F) and the Current-Controlled Voltage Source (H) devices have the same syntax. For a Current-Controlled Voltage Source just substitute a "H" for the "F". The "H" device generates a voltage, whereas, the "F" device generates a current.*

General Form

F<*name*> <(+) *node*> <(-) *node*>
+ <*controlling V device name*> <*gain*>

F<*name*> <(+) *node*> <(-) *node*> POLY(<*value*>)
+ <*controlling V device name*>*
+ < <*polynomial coefficient value*> >*

(+) and (-) These nodes are the output nodes. A positive current flows from the (+) node through the source to the (-) node. The current through the controlling voltage source determines the output current. The controlling source must be an independent voltage source (V device), although it need not have a zero DC value.

For the linear case, there must be one controlling voltage source and its name is followed by the gain. For all cases, including the nonlinear case (POLY), refer to your PSpice user's guide.

Note *Expressions cannot be used for linear and polynomial coefficient values in a current-controlled current source device statement.*

Examples

```
FSENSE 1 2 VSENSE 10.0
FAMP 13 0 POLY(1) VIN 0 500
FNONLIN 100 101 POLY(2) VCNTRL1 VCINTRL2 0.0 13.6 0.2 0.005
```

The first form and the first two examples apply to the linear case. The second form and the last example are for the nonlinear case. POLY(<*value*>) specifies the number of dimensions of the polynomial. The number of controlling voltage sources must be equal to the number of dimensions.

Independent Current Source & Stimulus and Independent Voltage Source & Stimulus

Note *The Independent Current Source & Stimulus (I) and the Independent Voltage Source & Stimulus (V) devices have the same syntax. For an Independent Voltage Source & Stimulus just substitute a "V" for the "I". The "V" device functions identically and has the same syntax as the "I" device, except that it generates voltage instead of current.*

General Form

I<*name*> <(+) *node*> <(-) *node*>
+ [[DC] <*value*>]
+ [AC <*magnitude value*> [*phase value*]]
+ [STIMULUS=<*stimulus name*>]
+ [*transient specification*]

Examples

```
IBIAS 13 0 2.3mA
IAC 2 3 AC .001
IACPHS 2 3 AC .001 90
IPULSE 1 0 PULSE(-1mA 1mA 2ns 2ns 2ns 50ns 100ns)
I3 26 77 DC .002 AC 1 SIN(.002 .002 1.5MEG)
```

This element is a current source. Positive current flows from the (+) node through the source to the (-) node: in the first example, IBIAS drives node 13 to have a *negative* voltage. The default value is zero for the DC, AC, and transient values. None, any, or all of the DC, AC, and transient values can be specified. The AC phase value is in degrees. The pulse and exponential examples are explained later in this section.

<*stimulus name*> References a .STIMULUS definition. See .STIMULUS.

[*transient specification*]

If present, they must be one of:

EXP (<*parameters*>)	for an exponential waveform
PULSE (<*parameters*>)	for a pulse waveform
PWL (<*parameters*>)	for a piecewise linear waveform
SFFM (<*parameters*>)	for a frequency-modulated waveform
SIN (<*parameters*>)	for a sinusoidal waveform

The variables TSTEP and TSTOP, which are used in defaulting some waveform parameters, are set by the .TRAN command. TSTEP is <*print step value*> and TSTOP is <*final time value*>. The .TRAN command can be anywhere in the circuit file; it need not come after the voltage source.

Independent Current Source & Stimulus (EXP)

General Form EXP (<i1> <i2> <td1> <tc1> <td2> <tc2>)

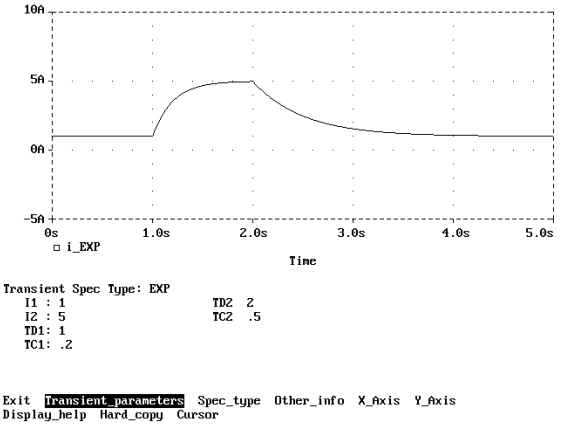
Examples

```
IRAMP 10 5 EXP(1 5 1 .2 2 .5)
```

Table 2-10 *Independent Current Source and Stimulus Exponential Waveform Parameters*

Parameters	Description	Units	Default
<i1>	Initial current	amp	none
<i2>	Peak current	amp	none
<td1>	Rise (fall) delay	sec	0
<tc1>	Rise (fall) time constant	sec	**TSTEP**
<td2>	Fall (rise) delay	sec	<td1>+**TSTEP**
<tc2>	Fall (rise) time constant	sec	**TSTEP**

Figure 2-3 *EXP current waveform created using StmEd (Stimulus Editor)*



The EXP form causes the current to be <i1> for the first <td1> seconds. Then, the current decays exponentially from <i1> to <i2> using a time constant of <tc1>. The decay lasts td2-td1 seconds. Then, the current decays from <i2> back to <i1> using a time constant of <tc2>. This behavior is shown in Figure 2-3. Alternatively, the waveform could be described by the following formulas

Table 2-11 *Independent Current Source and Stimulus Exponential Waveform Formulas*

Time Period	Value
0 to <td1>	i1
<td1> to <td2>	$i1 + (i2-i1)\cdot(1-e^{-(TIME-td1)/tc1})$
<td2> to TSTOP	$i1 + (i2-i1)\cdot((1-e^{-(TIME-td1)/tc1})-(1-e^{-(TIME-td2)/tc2}))$

Independent Current Source & Stimulus (PULSE)

General Form PULSE (<i1> <i2> <td> <tr> <tf> <pw> <per>)

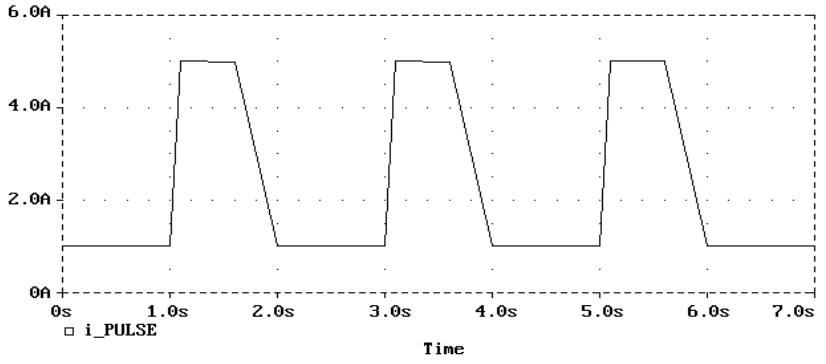
Examples

```
ISW 10 5 PULSE(1A 5A 1sec .1sec .4sec .5sec 2sec)
```

Table 2-12 *Independent Current Source and Stimulus Pulse Waveform Parameters*

Parameters	Description	Units	Default
<i1>	Initial current	amp	none
<i2>	Pulsed current	amp	none
<per>	Period	sec	**TSTOP**
<pw>	Pulse width	sec	**TSTOP**
<td>	Delay	sec	0
<tf>	Fall time	sec	**TSTEP**
<tr>	Rise time	sec	**TSTEP**

Figure 2-4 *PULSE current waveform created using StmEd (Stimulus Editor)*



The PULSE form causes the current to start at <i1>, and stay there for <td> seconds. Then, the current goes linearly from <i1> to <i2> during the next <tr> seconds, and then the current stays at <i2> for <pw> seconds. Then, it goes linearly from <i2> back to <i1> during the next <tf> seconds. It stays at <i1> for per-(tr+pw+tf) seconds, and then the cycle is repeated except for the initial delay of <td> seconds. This behavior is shown in Figure 2-4. Alternatively, the waveform could be described by the following table:

Table 2-13 *Independent Current Source and Stimulus Pulse Waveform Formulas*

Time	Value
0	i1
td	i1
td+tr	i2
td+tr+pw	i2
td+tr+pw+tf	i1
td+per	i1td
td+per+tr	i2
.	.
.	.
.	.

Independent Current Source & Stimulus (PWL)

General Form

PWL
+ [TIME_SCALE_FACTOR=<*value*>]
+ [VALUE_SCALE_FACTOR=<*value*>]
+ (*corner_points*)*

where *corner_points* are:

(<*tn*>, <*in*>)	to specify a point
FILE <*filename*>	to read point values from a file
REPEAT FOR <*n*> (*corner_points*)* ENDREPEAT	to repeat <*n*> times
REPEAT FOREVER (*corner_points*)* ENDREPEAT	to repeat forever

Examples

```
v1     1 2    PWL    (0,1) (1.2,5) (1.4,2) (2,4) (3,1)
v2     3 4    PWL    REPEAT FOR 5 (1,0) (2,1) (3,0)
ENDREPEAT
v3     5,6    PWL    REPEAT FOR 5 FILE DATA1.TAB
+                     ENDREPEAT
v4     7 8    PWL    TIME_SCALE_FACTOR=0.1
+             REPEAT FOREVER
+                     REPEAT FOR 5 (1,0) (2,1) (3,0)
ENDREPEAT
+                     REPEAT FOR 5 FILE DATA1.TAB
+                     ENDREPEAT
+             ENDREPEAT
```

n volt square wave (where *n* is 1, 2, 3, 4, then 5); 75% duty cycle; 10 cycles; 1 microseconds per cycle:

```
.PARAM  N=1
.STEP PARAM  N    1,5,1
V1  1  0  PWL
+      TIME_SCALE_FACTOR=1e-6 ;all time units are scaled to
+      microseconds
+      REPEAT FOR 10
+      (.25, 0)(.26, {N})(.99, {N})(1, 0)
+      ENDREPEAT
```

5 volt square wave; 75% duty cycle; 10 cycles; 10 microseconds per cycle; followed by 50% duty cycle *n* volt square wave (where *n* is 1, 2, 3, 4, then 5) lasting until the end of simulation:

```
.PARAM  N=.2
.STEP PARAM  N    .2,  1.0,  .2
V1  1  0  PWL
+TIME_SCALE_FACTOR=1e-5  ; all time units are scaled +to 10 us
+VALUE_SCALE_FACTOR=5
+REPEAT FOR 10
+(.25, 0)(.26, 1)(.99, 1)(1, 0)
+ENDREPEAT


+REPEAT FOREVER
+(+.50, 0)
+(+.01, {N}); iteration time .51
+(+.48, {N}); iteration time .99
+(1, 0)
+ENDREPEAT
```

Assuming that a PWL specification has been given for a device to generate two triangular waveforms:

```
V3   1 0   PWL
+   (1ms, 1)(2ms, 0)(3ms, 1)(4ms, 0)
```

Or, to replace the above with

```
V3   1   0   PWL   FILE   TRIANGLE.IN
```

where the file “triangle.in” would need to contain:

```
(1ms, 1)(2ms, 0)(3ms, 1)(4ms, 0)
```

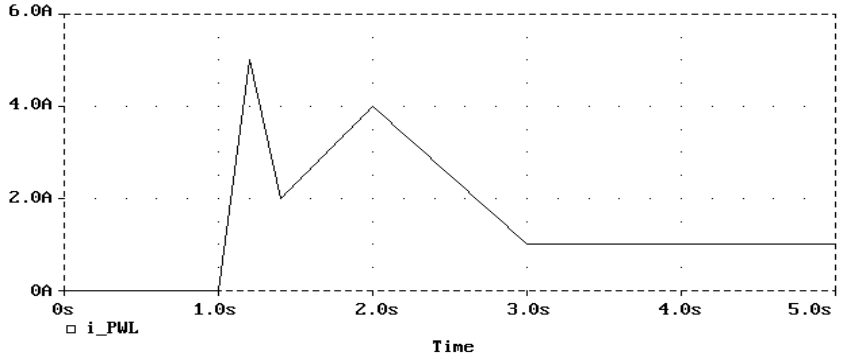
Table 2-14 *Independent Voltage Source and Stimulus PWL Waveform Parameters*

Parameters*	Description	Units	Default
<tn>	Time at corner	seconds	None
<vn>	Voltage at corner	volts	None
<n>	Number of repetitions	positive integer, 0, or -1	None

* <tn> and <n> cannot be expressions; <vn> may be an expression.

The PWL form describes a piecewise linear waveform. Each pair of time-current values specifies a corner of the waveform. The current at times between corners is the linear interpolation of the currents at the corners. This behavior is shown in Figure 2-5

Figure 2-5 *PWL current waveform created using StmEd (Stimulus Editor).*



<*time_scale_factor*> and / or <*value_scale_factor*>

These keywords can be coded immediately after the PWL keyword to show that the time and / or current value pairs are to be multiplied by the appropriate scale factor.

These scale factors can be expressions. If they are expressions, they are evaluated once per outer simulation loop, and thus should be composed of expressions not containing references to voltages or currents.

<*tn*> and <*in*>

The transient specification corner points for the PWL waveform shown in Figure 2-5 are shown in the first example.

The <*in*> can be an expression having the same restrictions as the scaling keywords, but <*tn*> must be a literal.

<*file name*>

The file named <*file name*> can be read to supply the (<*tn*> <*in*>) pairs. The specified file is a text file containing the time-current pairs. The contents of this file are read by the same parser that reads the circuit file. Thus, engineering units (e.g., 10us) are correctly interpreted. Note that the continuation + signs in the first column are unnecessary and are discouraged.

A typical file can be created by editing an existing PWL specification, replacing all + signs with blanks (to avoid

unintentional +time). Only numbers (having units attached) can appear in the file; expressions for <*tn*> and <*n*> values are not allowed. All absolute time points in <*file name*> are with respect to the last (<*tn*> <*in*>) entered. All relative time points are with respect to the last time point.

REPEAT ... ENDREPEAT

These loops permit repetitions.

They can appear anywhere a (<*tn*> <*in*>) pair can appear. Absolute times within REPEAT loops are with respect to the start of the current iteration. The REPEAT ... ENDREPEAT specifications can be nested to any depth. Make sure that the current value associated with the beginning and ending time points (within the same REPEAT loop or between adjacent REPEAT loops), are the same when 0 is specified as the first point in a REPEAT loop.

<*n*> A REPEAT FOR -1 ... ENDREPEAT is treated as if it had been REPEAT FOREVER ... ENDREPEAT. A REPEAT FOR 0 ... ENDREPEAT is ignored (other than syntax checking of the enclosed corner points).

Independent Current Source & Stimulus (SFFM)

General Form SFFM (<ioff> <iampl> <fc> <mod> <fm>)

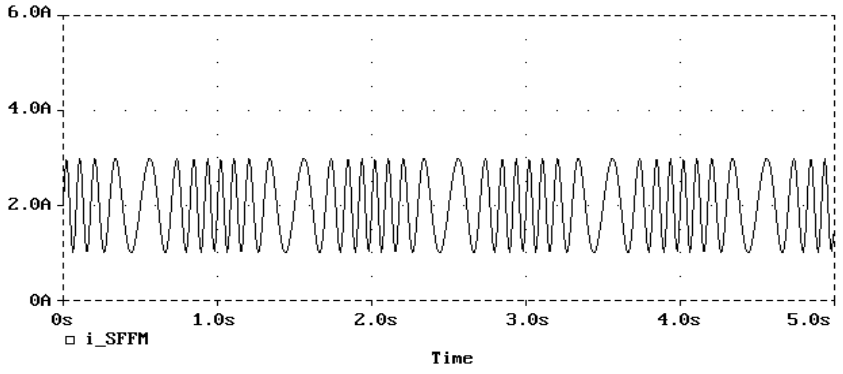
Examples

```
IMOD 10 5 SFFM(2 1 8Hz 4 1Hz)
```

Table 2-15 *Independent Current Source and Stimulus Frequency-Modulated Waveform Parameters*

Parameters	Description	Units	Default
<fc>	Carrier frequency	hertz	1/TSTOP
<fm>	Modulation frequency	hertz	1/**TSTOP**
<iampl>	Peak amplitude of current	amp	none
<ioff>	Offset current	amp	none
<mod>	Modulation index		0

Figure 2-6 *SFFM current waveform created using StmEd (Stimulus Editor)*



The SFFM (Single-Frequency FM) form causes the current, as shown in Figure 2-6, to follow this formula

$$ioff + iampl \cdot sin(2\pi \cdot fc \cdot TIME + mod \cdot sin(2\pi \cdot fm \cdot TIME))$$

Independent Current Source & Stimulus (SIN)

General Form SIN (<ioff> <iampl> <freq> <td> <df> <phase>)

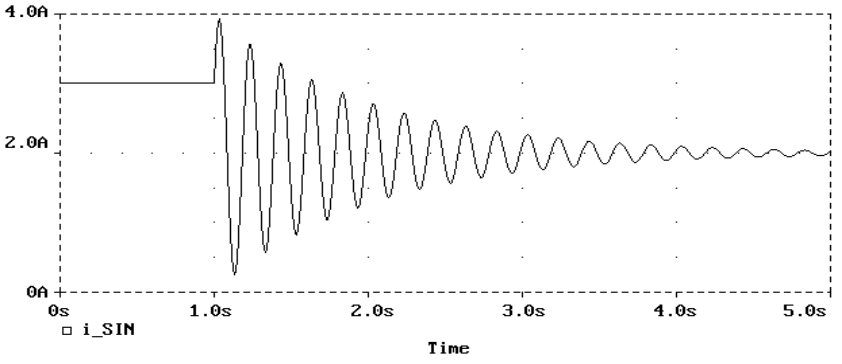
Examples

```
ISIG 10 5 SIN(2 2 5Hz 1sec 1 30)
```

Table 2-16 *Independent Current Source and Stimulus Sinusoidal Waveform Parameters*

Parameters	Description	Units	Default
<df>	Damping factor	sec^{-1}	0
<freq>	Frequency	hertz	1/**TSTOP**
<iampl>	Peak amplitude of current	amp	none
<ioff>	Offset current	amp	none
<phase>	Phase	degree	0
<td>	Delay	sec	0

Figure 2-7 *SIN current waveform created using StmEd (Stimulus Editor)*



The sinusoidal (SIN) waveform causes the current to start at <ioff> and stay there for <td> seconds.

Then, the current becomes an exponentially damped sine wave. This behavior is shown in Figure 2-7. The waveform could be described by the following formulas.

Table 2-17 *Independent Current Source and Stimulus Sinusoidal Waveform Formulas*

Time period	Value
to <td>	$ioff+iampl \cdot sin(2\pi \cdot phase/360°)$
<td> to TSTOP	$ioff+iampl \cdot sin(2\pi \cdot (freq \cdot (TIME-td)+phase/360°)) \cdot e^{-(TIME-td) \cdot df}$

Note *The SIN waveform is for transient analysis only. It does not have any effect during AC analysis. To give a value to a current during AC analysis, use an AC specification, such as*

```
IAC 3 0 AC 1mA
```

where IAC has an amplitude of one milliampere during AC analysis, and can be zero during transient analysis. For transient analysis use (for example)

```
ITRAN 3 0 SIN(0 1mA 1kHz)
```

where ITRAN has an amplitude of one milliampere during transient analysis and is zero during AC analysis. Refer to your PSpice user's guide.

Junction FET

General Form

J<*name*> <*drain node*> <*gate node*> <*source node*>
+ <*model name*> [*area value*]

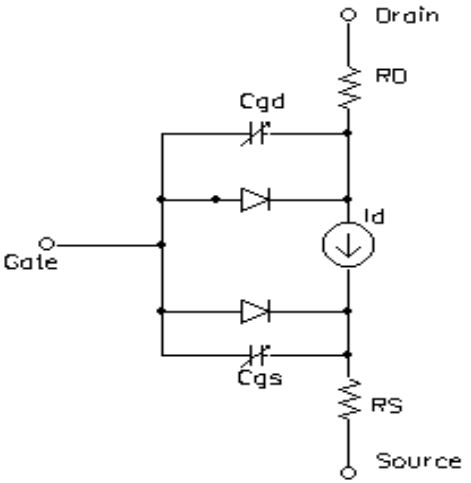
Examples

```
JIN 100 1 0 JFAST
J13  22 14 23 JNOM 2.0
```

Model Form

```
.MODEL <model name> NJF [model parameters]
.MODEL <model name> PJF [model parameters]
```

Figure 2-8 *JFET Model*



As shown, the JFET is modeled as an intrinsic FET using an ohmic resistance (**RD** / *area*) in series with the drain, and using another ohmic resistance (**RS** / *area*) in series with the source. Positive current is current flowing into a terminal. The [*area value*] is the relative device area and defaults to 1.

Table 2-18 *Junction FET Model Parameters*

Model Parameters*	Description	Units	Default
AF	Flicker noise exponent		1
ALPHA	Ionization coefficient	$volt^{-1}$	0
BETA	Transconductance coefficient	$amp/volt^2$	1E-4
BETATCE	BETA exponential temperature coefficient	%/°C	0
CGD	Zero-bias gate-drain *p-n* capacitance	farad	0
CGS	Zero-bias gate-source *p-n* capacitance	farad	0
FC	Forward-bias depletion capacitance coefficient		0.5
IS	Gate *p-n* saturation current	amp	1E-14
ISR	Gate *p-n* recombination current parameter	amp	0
KF	Flicker noise coefficient		0
LAMBDA	Channel-length modulation	$volt^{-1}$	0
M	Gate *p-n* grading coefficient		0.5
N	Gate *p-n* emission coefficient		1
NR	Emission coefficient for ISR		2
PB	Gate *p-n* potential	volt	1.0
RD	Drain ohmic resistance	ohm	0
RS	Source ohmic resistance	ohm	0
T_ABS	Absolute temperature	°C	
T_MEASURED	Measured temperature	°C	
T_REL_GLOBAL	Relative to current temperature	°C	
T_REL_LOCAL	Relative to AKO model temperature	°C	
VK	Ionization “knee” voltage	volt	0
VTO	Threshold voltage	volt	-2.0
VTOTC	VTO temperature coefficient	volt/°C	0
XTI	IS temperature coefficient		3

* For information on **T_MEASURED**, **T_ABS**, **T_REL_GLOBAL**, and **T_REL_LOCAL**, see the .MODEL statement on page 1-25.

Note **VTO** *< 0 means the device is a depletion-mode JFET (for both N-channel and P-channel) and* **VTO** *> 0 means the device is an enhancement-mode JFET. This conforms to U.C. Berkeley SPICE.*

Equations

In the following equations:

Vgs = intrinsic gate-intrinsic source voltage
Vgd = intrinsic gate-intrinsic drain voltage
Vds = intrinsic drain-intrinsic source voltage
Vt = $k \cdot T / q$ (thermal voltage)
k = Boltzmann's constant
q = electron charge
T = analysis temperature (°K)
Tnom = nominal temperature (set using TNOM option)

Other variables are from the model parameter list. These equations describe an N-channel JFET. For P-channel devices, reverse the sign of all voltages and currents. Positive current is current flowing into a terminal (for example, positive drain current flows from the drain through the channel to the source).

DC Currents[1]

Ig = gate current = *area*·(Igs + Igd)

Igs = gate-source leakage current = In + Ir·Kg

In = normal current = **IS**·($e^{Vgs/(N\cdot Vt)}$-1)

Ir = recombination current = **ISR**·($e^{Vgs/(NR\cdot Vt)}$-1)

Kg = generation factor = $((1-Vgs/\mathbf{PB})^2+0.005)^{M/2}$

Igd = gate-drain leakage current = In + Ir·Kg + Ii

In = normal current = **IS**·($e^{Vgd/(N\cdot Vt)}$-1)

Ir = recombination current = **ISR**·($e^{Vgd/(NR\cdot Vt)}$-1)

Kg = generation factor = $((1-Vgd/\mathbf{PB})^2+0.005)^{M/2}$

Ii = impact ionization current

For: 0 < Vgs-**VTO** < Vds forward saturation region)

Ii = Idrain·**ALPHA**·vdif·$e^{-VK/vdif}$

where vdif = Vds - (Vgs-**VTO**)

otherwise

Ii = 0

Id = drain current = *area*·(Idrain-Igd)

Is = source current = *area*·(-Idrain-Igs)

Equation for Idrain

For: Vds ≥ 0 (normal mode)

and: Vgs-**VTO** ≤ 0 (cutoff region)

Idrain = 0

and: Vds ≤ Vgs-**VTO** (linear region)

Idrain = **BETA**·(1+**LAMBDA**·Vds)·Vds·(2·(Vgs-**VTO**)-Vds)

and: 0 < Vgs-**VTO** < Vds(saturation region)

Idrain = **BETA**·(1+**LAMBDA**·Vds)·$(Vgs-VTO)^2$

For: Vds < 0 (inverted mode)

Switch the source and drain in equations (above).

1. Positive current is current flowing into a terminal.

Capacitance[1]

Cgs = gate-source depletion capacitance

For: $Vgs \leq \mathbf{FC} \cdot \mathbf{PB}$

$Cgs = area \cdot \mathbf{CGS} \cdot (1 - Vgs/\mathbf{PB})^{-M}$

For: $Vgs > \mathbf{FC} \cdot \mathbf{PB}$

$Cgs = area \cdot \mathbf{CGS} \cdot (1-\mathbf{FC})^{-(1+M)} \cdot (1-\mathbf{FC} \cdot (1+\mathbf{M}) + \mathbf{M} \cdot Vgs/\mathbf{PB})$

Cgd = gate-drain depletion capacitance

For: $Vgd \leq \mathbf{FC} \cdot \mathbf{PB}$

$Cgd = area \cdot \mathbf{CGD} \cdot (1 - Vgd/\mathbf{PB})^{-M}$

For: $Vgd > \mathbf{FC} \cdot \mathbf{PB}$

$Cgd = area \cdot \mathbf{CGD} \cdot (1-\mathbf{FC})^{-(1+M)} \cdot (1-\mathbf{FC} \cdot (1+\mathbf{M}) + \mathbf{M} \cdot Vgd/\mathbf{PB})$

1. All capacitances are between terminals of the intrinsic JFET (that is, to the inside of the ohmic drain and source resistances).

Temperature Effects

VTO(T) = **VTO**+**VTOTC**·(T-Tnom)

BETA(T) = **BETA**·$1.01^{\text{BETATCE}\cdot(\text{T-Tnom})}$

IS(T) = **IS**·$e^{(\text{T/Tnom-1})\cdot\text{EG}/(\text{N}\cdot\text{Vt})}$·$(\text{T/Tnom})^{\text{XTI/N}}$
where EG = 1.11

ISR(T) = **ISR**·$e^{(\text{T/Tnom-1})\cdot\text{EG}/(\text{NR}\cdot\text{Vt})}$·$(\text{T/Tnom})^{\text{XTI/NR}}$
where EG = 1.11

PB(T) = **PB**·T/Tnom - 3·Vt·*ln*(T/Tnom) - Eg(Tnom)·T/Tnom
+ Eg(T)
where Eg(T) = silicon bandgap energy = 1.16
- .000702·T^2/(T+1108)

CGS(T) = **CGS**·(1+**M**·(.0004·(T-Tnom)+(1-**PB**(T)/**PB**)))

CGD(T) = **CGD**·(1+**M**·(.0004·(T-Tnom)+(1-**PB**(T)/**PB**)))

The drain and source ohmic (parasitic) resistances have no temperature dependence.

Noise

Noise is calculated assuming a one hertz bandwidth, using the following spectral power densities (per unit bandwidth):

the parasitic resistances, Rs and Rd, generate thermal noise ...
$Is^2 = 4\cdot k\cdot T/(\textbf{RS}/area)$
$Id^2 = 4\cdot k\cdot T/(\textbf{RD}/area)$

the intrinsic JFET generates shot and flicker noise ...
$Idrain^2 = 4\cdot k\cdot T\cdot gm\cdot 2/3 + \textbf{KF}\cdot Idrain^{AF}/FREQUENCY$
where gm = *d*Idrain/*d*Vgs (at the DC bias point)

Reference

For a generally detailed discussion of the U.C. Berkeley SPICE models, including the JFET device, refer to:

[1] P. Antognetti and G. Massobrio, *Semiconductor Device Modeling with SPICE*, McGraw-Hill, 1988.

Inductor or Transmission Line Coupling

General Form

K<*name*> L<*inductor name*> < L<*inductor name*> >*
+ <*coupling value*>

K<*name*> < L<*inductor name*> >* <*coupling value*>
+ <*model name*> [*size value*]

K<*name*>T<*transmission line name*>T<*transmission line name*>
+ Cm=<*capacitive coupling*> Lm=<*inductive coupling*>

Examples

```
KTUNED L3OUT  L4IN .8
KTRNSFRM  LPRIMARY  LSECNDRY 1
KXFRM L1 L2  L3  L4 .98 KPOT_3C8
K2LINES  T1  T2  Lm=1m  Cm=.5p
```

Model Form

`.MODEL <model name> CORE [model parameters]`

This device can be used to define coupling between inductors (transformers) or between transmission lines. This device also refers to a nonlinear magnetic core (CORE) model to include magnetic hysteresis effects in the behavior of a single inductor (winding), or in multiple coupled windings.

Table 2-19 *Inductor Coupling Model Parameters*

Model Parameters*	Description	Units	Default
A	Thermal energy parameter	amp / meter	1E+3
AREA	Mean magnetic cross-section	cm^2	0.1
C	Domain flexing parameter		0.2
GAP	Effective air-gap length	cm	0
K	Domain anisotropy parameter	amp / meter	500
LEVEL	Model index		2
MS	Magnetization saturation	amp / meter	1E+6
PACK	Pack** (stacking) factor		1.0
PATH	Mean magnetic path length	cm	1.0

*See .MODEL statement.

**Flux is proportional to PACK.

Inductor Coupling

K<*name*> couples two, or more, inductors. Using the "dot" convention, place a "dot" on the first node of each inductor. In other words, given:

```
I1 1 0 AC 1mA
L1 1  0 10uH
L2 2  0 10uH
R2 2  0 .1
K12 L1 L2 1
```

the current through L2 is in the opposite direction as the current through L1. The polarity is determined by the order of the nodes in the L device(s) and not by the order of inductors in the K statement.

<coupling value> This is the "coefficient of mutual coupling" which must be between 0 and 1.

This coefficient is defined by the equation

$$\text{<coupling value>} = M_{ij} / (L_i \cdot L_j)^{1/2}$$

where

L_i, L_j are a coupled-pair of inductors
M_{ij} is the mutual inductance between L_i and L_j

For transformers of normal geometry, the value one should be used. Values less than one occur in air core transformers when the coils do not completely overlap.

The linear branch relation for transient analysis is

$$V_i = L_i \cdot \frac{dI_i}{dt} + M_{ij} \cdot \frac{dI_j}{dt} + M_{ik} \cdot \frac{dI_k}{dt} + \cdots$$

For U.C. Berkeley SPICE2: if there are several coils on a transformer, then there must be K statements coupling all combinations of inductor pairs. For instance, a transformer using a center-tapped primary and two secondaries would be written:

```
* PRIMARY
L1 1 2 10uH
L2 2 3 10uH
* SECONDARY
L3 11 12 10uH
L4 13 14 10uH
* MAGNETIC COUPLING
K12 L1 L2 1
K13 L1 L3 1
K14 L1 L4 1
K23 L2 L3 1
K24 L2 L4 1
K34 L3 L4 1
```

This "older" technique is still supported, but *not required*, for simulation. The same transformer can now be written:

```
* PRIMARY
L1 1 2 10uH
L2 2 3 10uH
* SECONDARY
L3 11 12 10uH
L4 13 14 10uH
* MAGNETIC COUPLING
KALL L1 L2 L3 L4 1
```

Note *Do not mix the two techniques.*

<*model name*> If <*model name*> is present, four things change:

1. The mutual coupling inductor becomes a nonlinear, magnetic core device. The magnetic core's B-H characteristics are analyzed using the Jiles-Atherton model (see Reference [1] below).
2. The inductors become "windings," so the number specifying inductance now specifies the "number of turns."
3. The list of coupled inductors could be just one inductor.
4. A model statement is required to specify the model parameters.

[*size value*] Defaults to one and scales the magnetic cross-section. It is intended to represent the number of lamination layers, so only one model statement is needed for each lamination type.

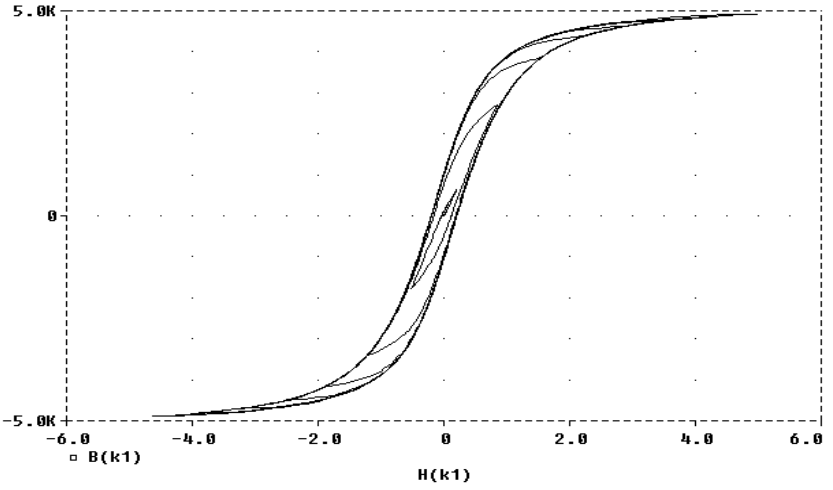
For example

```
L1 5 9 20                 ; inductor having 20 turns
K1 L1 1 K528T500_3C8      ; Ferroxcube toroid core
L2 3 8 15                 ; primary winding having
                            15 turns
L3 4 6 45                 ; secondary winding having
                            45 turns
K2 L2 L3 1 K528T500_3C8   ; another core (not the same
                            as K1)
```

The Jiles-Atherton model is based on existing ideas of domain wall motion, including flexing and translation. The model derives an anhysteric magnetization curve using a mean field technique in which any domain is coupled to the magnetic field (H) and the bulk magnetization (M). This anhysteric value is the magnetization which would be reached in the absence of domain wall pinning. Hysteresis is modeled by the effects of pinning of domain walls on material defect sites. This impedance to motion and flexing due to the differential field exhibits all of the main features of real, nonlinear magnetic devices, such as: the initial magnetization curve (initial permeability), saturation of magnetization, coercivity, remanence, and hysteresis loss.

These features are shown in Figure 2-9.

Figure 2-9 *Probe B-H display of 3C8 ferrite (Ferroxcube)*



The simulator uses the JilesAtherton model to analyze the B-H curve of the magnetic core, and calculate values for inductance and flux for each of the "windings."

The state of the nonlinear core can be viewed in Probe by specifying B(K*xxx*), for the magnetization, or H(K*xxx*), for the magnetizing influence. These values are not available for .PRINT or .PLOT output.

Jiles-Atherton Model

Magnetic material made up of loosely coupled domains have an equilibrium B-H curve, called the “anhysteric”. This curve is the locus of B-H values generated by superimposing a DC magnetic bias and a large AC signal which decays to zero. It is the curve representing minimum energy for the domains, and is modeled, in theory, by

$$M_{an} = \mathbf{MS} \cdot H / (|H| + \mathbf{A})$$

where

M_{an}	is the anhysteric magnetization
MS	is the saturation magnetization
H	is the magnetizing influence (after GAP correction)
A	is a thermal energy parameter

For a given magnetizing influence, H, the anhysteric magnetization is the global flux level the material would attain if the domain walls could move freely. The walls, however, are stopped or pinned on dislocations in the material. The wall remains pinned until enough magnetic potential is available to break free, and travel to the next pinning site. The theory supposes a mean energy required, per volume, to move domain walls. This is analogous to mechanical “drag.” So a (simplified) equation of this is

change-in-magnetization = potential / drag

The irreversible domain wall motion can, therefore, be expressed as

$$dM_{irr} / dH = (M_{an} - M) / \mathbf{K}$$

where

K	is the pinning energy per volume (drag)

Reversible wall motion comes from flexing in the domain walls, especially when it is pinned at a dislocation due to the magnetic potential (that is, the magnetization is not the anhysteric value).

The theory supposes spherical flexure to calculate energy values and arrives at the (simplified) equation

$$dM_{rev} / dH = \mathbf{C} \cdot d(M_{an}\text{-}M) / dH$$

where

C is the domain flexing parameter

The equation for the total magnetization is the sum of these two state equations, and is, therefore:

$$dM / dH = (1 / (1 + \mathbf{C})) \cdot (M_{an} - M) / \mathbf{K}) + (C / (1 + \mathbf{C})) \cdot dM_{an} / dH$$

Including Air-Gap Effects in the Model

If the gap thickness is small compared with the other dimensions of the core, it can be assumed that all of the magnetic flux lines go through the gap directly and that there is little “fringing flux” (having a modest amount of fringing flux only increases the effective air-gap length). Checking the field values around the entire magnetic path, gives the equation

$$Hcore \cdot Lcore + Hgap \cdot Lgap = n \cdot I$$

where n·I is the sum of the amp-turns of the windings on the core. Also, the magnetization in the air-gap is negligible so that Bgap = Hgap, and Bgap = Bcore. These combine in the previous equation to yield

$$Hcore \cdot Lcore + Bcore \cdot Lgap = n \cdot I$$

This is a difficult equation to solve especially for the Jiles-Atherton model, which is a state equation model rather than an explicit function (which one would expect since the B-H curve depends on the history of the material). However, there is a graphical technique which solves for Bcore and Hcore, given n·I, which is to: (i) take the non-gapped B-H curve, (ii) extend a line from the current value of n·I having a slope of -Lcore / Lgap (this would be vertical if Lgap = 0), and (iii) find the intersection of the line using the B-H curve.

The intersection point is the value for Bcore and Hcore for the n·I of the gapped core. The n·I value is the apparent or external value of Hcore, but the real value of Hcore is less. This results in a smaller value for Bcore and the “sheared over” B-H curves of a gapped core. The simulator implements the numerical equivalent

of this graph technique.

The resulting B-H values are recorded in the Probe data file as B_{core} and $H_{apparent}$, since this is what the circuit "sees."

Getting Core Model Values

Characterizing core materials can be performed using Parts, and verified by using PSpice and Probe. The model uses MKS (metric) units, however the results for Probe are converted to Gauss and Oersted, which can be displayed using B(K*xxx*) and H(K*xxx*). The traditional B-H curve is made by a transient run, ramping current through a test inductor, then displaying B(K*xxx*) and setting the X axis to H(K*xxx*).

Reference

For a further description of the Jiles-Atherton model, refer to:

[1] D.C. Jiles, and D.L. Atherton, "Theory of ferromagnetic hysteresis," *Journal of Magnetism and Magnetic Materials***, 61, 48 (1986).**

Transmission Line Coupling

If a K device is used to couple two transmission lines, then two coupling parameters are required.

Table 2-20 *Transmission Line Coupling Device Parameters*

Device	Description	Units	Default
Cm	Capacitive coupling	Farad / Length*	none
Lm	Inductive coupling	Henries / Length*	none

* Length units must be consistent using the LEN parameter for the transmission lines being coupled.

These parameters can be thought of as the off-diagonal terms of a capacitive coupling matrix, [C], and an inductive coupling matrix, [L], respectively. [C] and [L] are both symmetric matrices, and for two coupled lines, the following relationships hold:

$$[C] = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \qquad [L] = \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix}$$

Cm = C12 = C21 **Lm** = L12 = L21

C_{12} represents the charge induced on the first conductor when the second conductor has a potential of one volt. In general, for a system of N coupled lines, C_{ij} is the charge on the i^{th} conductor when the j^{th} conductor is set to one volt, and all other conductors are grounded. The diagonal of the matrix is determined with the understanding that the self-capacitance is really the capacitance between the conductor and ground, so

$$C_{ii} = C_{ig} + \sum |C_{ij}|$$

C_{ig} is equal to the capacitance per unit length for the i^{th} transmission line, and is provided along with the T device that describes the i^{th} line. The simulator computes C_{ii} from this.

The values of C_{ij} in the matrix are negative values. Note that the simulator assigns -|**Cm**| to the appropriate C_{ij}, so the sign used when specifying **Cm** is ignored.

L_{12} is defined in terms of the flux between the 1st conductor and the ground plane when the 2nd conductor carries a current of one ampere. If there are more than two conductors, all other conductors are assumed to be open.

L_{11} is equal to the inductance per unit length for the 1st line, and is obtained directly from the appropriate T device.

The following circuit fragment shows an example using two coupled lines:

```
T1 1 0 2 0 R=.31 L=.38u G=6.3u C=70p LEN=1
T2 3 0 4 0 R=.29 L=.33u G=6.0u C=65p LEN=1
K12 T1 T2 Lm=.04u Cm=6p
```

This fragment leads to the following [C] and [L]:

$$[C] = \begin{bmatrix} 76p & -6p \\ -6p & 71p \end{bmatrix} \qquad [L] = \begin{bmatrix} 0.38u & 0.04u \\ 0.04u & 0.33u \end{bmatrix}$$

The model used to simulate this system is based on the approach described by Tripathi and Rettig in reference [1] and extended for lossy lines by Roychowdhury and Pederson in reference [2]. The approach involves computing the system propagation modes by extracting the eigenvalues and eigenvectors of the matrix product [L][C]. The interested reader is referred to the references for the details. However, it is important to note that the model is not general for lossy lines.

For the lossy line case, the matrix product to be decoupled is actually [R+sL][G+sC], where s is the Laplace variable, R is the resistance per unit length matrix, and G is the conductance per unit length matrix. The modes obtained from [L][C] represent a high frequency asymptote for this system. Simulation results should be good approximations for low-loss lines. However, as shown in reference [2], the approximation becomes exact for homogeneous, equally-spaced lossy lines, provided that coupling beyond immediately adjacent lines is negligible (i.e., the coupling matrices are tridiagonal and Toeplitz).

Note *Coupled ideal lines can be modeled by setting R and G to zero. The ZO/TD parameter set is not supported for coupled lines.*

References

[1] Tripathi and Rettig, “A SPICE Model for Multiple Coupled Microstrips and Other Transmission Lines,” *IEEE MTT-S Internal Microwave Symposium Digest*, 1985.

[2] Roychowdhury and Pederson, “Efficient Transient Simulation of Lossy Interconnect,” Design Automation Conference, 1991.

Inductor

General Form

L<*name*> <(+) *node*> <(-) *node*> [*model name*] <*value*>
+ [IC=<*initial value*>]

Examples

```
LLOAD 15 0 20mH
L2 1 2 .2E-6
LCHOKE 3 42 LMOD .03
LSENSE 5 12 2UH IC=2mA
```

Model Form

`.MODEL <model name> IND [model parameters]`

(+) and (-)

The (+) and (-) nodes define the polarity when the inductor has a positive voltage across it.

The first node listed (or pin one in Schematics), is defined as positive. The voltage across the component is therefore defined as the first node voltage less the second node voltage.

Positive current flows from the (+) node through the inductor to the (-) node. Current flow from the first node through the component to the second node is considered positive.

[*model name*]

If [*model name*] is left out, then the effective value is <*value*>.

If [*model name*] is specified, then the effective value is given by the formula

$$<value>\cdot\mathbf{L}\cdot(1+\mathbf{IL1}\cdot I+\mathbf{IL2}\cdot I^2)\cdot(1+\mathbf{TC1}\cdot(T-Tnom)+\mathbf{TC2}\cdot(T-Tnom)^2)$$

where <*value*> is normally positive (though it can be negative, but *not* zero). “Tnom” is the nominal temperature (set using TNOM option).

If the inductor is associated with a Core model, then the effective value is the number of turns on the core. Otherwise, the effective value is the inductance. See the .MODEL statement for the “K” device on page 50 for more information on the Core model.

<*initial value*>

The initial current through the inductor during the bias point calculation.

It can also be specified in a circuit file using a .IC statement as follows:

.IC I(L<*name*>) <*initial value*>

For details on using the .IC statement in a circuit file, see page 1-16 in this manual, and refer to your PSpice user's guide for more information.

Table 2-21 *Inductor Model Parameters*

Model Parameters*	Description	Units	Default
L	Inductance multiplier		1
IL1	Linear current coefficient	amp^{-1}	0
IL2	Quadratic current coefficient	amp^{-2}	0
TC1	Linear temperature coefficient	$°C^{-1}$	0
TC2	Quadratic temperature coefficient	$°C^{-2}$	0
T_ABS	Absolute temperature	°C	
T_MEASURED	Measured temperature	°C	
T_REL_GLOBAL	Relative to current temperature	°C	
T_REL_LOCAL	Relative to AKO model temperature	°C	

* For information on **T_MEASURED**, **T_ABS**, **T_REL_GLOBAL**, and **T_REL_LOCAL**, see the .MODEL statement on page1-25.

Noise

The inductor does not have a noise model.

MOSFET

General Form

M<*name*> <*drain node*> <*gate node*> <*source node*>
+ <*bulk/substrate node*> <*model name*>
+ [L=<*value*>] [W=<*value*>]
+ [AD=<*value*>] [AS=<*value*>]
+ [PD=<*value*>] [PS=<*value*>]
+ [NRD=<*value*>] [NRS=<*value*>]
+ [NRG=<*value*>] [NRB=<*value*>]
+ [M=<*value*>]

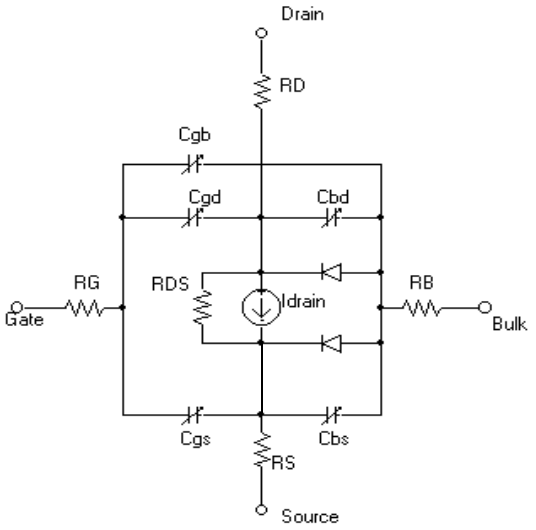
Examples

```
M1 14 2 13 0 PNOM  L=25u W=12u
M13 15 3 0 0 PSTRONG
M16 17 3 0 0 PSTRONG M=2
M28 0 2 100 100 NWEAK L=33u W=12u
+ AD=288p AS=288p PD=60u PS=60u NRD=14 NRS=24 NRG=10
```

Model Form

```
.MODEL <model name> NMOS [model parameters]
.MODEL <model name> PMOS [model parameters]
```

Figure 2-10 *MOSFET Model*



As shown in Figure Model Form, the MOSFET is modeled as an intrinsic MOSFET using ohmic resistances in series with the drain, source, gate, and bulk (substrate). There is also a shunt resistance (RDS) in parallel with the drain-source channel.

Note *[L=<value>] [W=<value>] cannot be used in conjunction with Monte Carlo analysis.*

The simulator provides four MOSFET device models, which differ in the formulation of the I-V characteristic. The **LEVEL** parameter selects between different models:

Table 2-22 *MOSFET Levels*

MOSFET LEVELS	Model Definition
LEVEL=1	Shichman-Hodges model (see reference [1])
LEVEL=2	geometry-based, analytic model (see reference [2])
LEVEL=3	semi-empirical, short-channel model (see reference [2])
LEVEL=4	BSIM model (see reference [3])
LEVEL=5	(No longer supported.)
LEVEL=6	BSIM3 model (see reference [7] Version 2.0)

L and W

These are the channel length and width, and are decreased to get the effective channel length and width.

L and W can be specified in the device, model, or .OPTIONS statements. The value in the device statement supersedes the value in the model statement, which supersedes the value in the .OPTIONS statement.

AD and AS

These are the drain and source diffusion areas.

PD and PS

These are the drain and source diffusion perimeters.

The drain-bulk and source-bulk saturation currents can be specified either by **JS**, which is multiplied by AD and AS, or by IS, which is an absolute value. The zero-bias depletion capacitances can be specified by **CJ**, which is multiplied by AD and AS, and by **CJSW**, which is multiplied by PD and PS. Or they can be set by **CBD** and **CBS**, which are absolute values.

NRD, NRS, NRG, and NRB

Consider a square sheet of resistive material. Analysis shows that the resistance between two parallel edges of such a sheet depends upon its composition and thickness, but is *independent* of its size as long as it is *square*. In other words, the resistance will be the same whether the square's edge is 2 mm, 2 cm, or 2 m. For this reason, the "sheet resistance" of such a layer, abbreviated **RSH** (see Table 2-27), has units of *ohms per square*.

The quantities NRD, NRS, NRG, and NRB are multipliers (in units of squares) that can be multiplied by **RSH** to yield the parasitic (ohmic) resistances of the drain, source, gate, and substrate respectively (designated **RD**, **RS**, **RG**, and **RB**).

PD and PS default to 0, NRD and NRS default to 1, and NRG and NRB default to 0. Defaults for L, W, AD, and AS can be set in the .OPTIONS statement. If AD or AS defaults are not set, they also default to 0. If L or W defaults are not set, they default to 100 u.

M

Device "multiplier" (default = 1), which simulates the effect of multiple devices in parallel.

The effective width, overlap and junction capacitances, and junction currents of the MOSFET are multiplied by M. The parasitic resistance values (e.g., **RD** and **RS**) are divided by M. Note the third example showing a device twice the size of the second example.

Model Levels 1, 2, and 3

The DC characteristics of the first three model levels are defined by the parameters **VTO**, **KP**, **LAMBDA**, **PHI**, and **GAMMA**. These are computed by the simulator if process parameters (e.g., **TOX**, and **NSUB**) are given, but the user-specified values always override (**Note:** The default value for **TOX** is 0.1 µ for model levels two and three, but is unspecified for level one which "turns off" the use of process parameters). **VTO** is positive (negative) for enhancement

mode and negative (positive) for depletion mode of N-channel (P-channel) devices.

Table 2-23 *MOSFET Level 1, 2, and 3 Model Parameters*

Model Parameters*	Description	Units	Default
DELTA	Width effect on threshold		0
ETA	Static feedback (**LEVEL**=3)		0
GAMMA	Bulk threshold parameter	$volt^{1/2}$	calculated
KP	Transconductance coefficient	$amp/volt^2$	2E-5
KAPPA	Saturation field factor (**LEVEL**=3)		0.2
LAMBDA	Channel-length modulation (**LEVEL**=1 or 2)	$volt^{-1}$	0
LD	Lateral diffusion (length)	meter	0
NEFF	Channel charge coefficient (**LEVEL**=2)		1.0
NFS	Fast surface state density	$1/cm^2$	0
NSS	Surface state density	$1/cm^2$	none
NSUB	Substrate doping density	$1/cm^3$	none
PHI	Surface potential	volt	0.6
THETA	Mobility modulation (**LEVEL**=3)	$volt^{-1}$	0
TOX	Oxide thickness	meter	see above
TPG	Gate material type: +1 = opposite of substrate -1 = same as substrate 0 = aluminum		+1
UCRIT	Mobility degradation critical field (**LEVEL**=2)	volt/cm	1E4
UEXP	Mobility degradation exponent (**LEVEL**=2)		0
UTRA	(***not*** *used*) Mobility degradation transverse field coefficient		0
UO	Surface mobility. (The second character is the letter O, not the numeral zero.)	$cm^2/volt \cdot sec$	600

Table 2-23 *MOSFET Level 1, 2, and 3 Model Parameters (continued)*

Model Parameters*	Description	Units	Default
VMAX	Maximum drift velocity	meter/sec	0
VTO	Zero-bias threshold voltage	volt	0
WD	Lateral diffusion (width)	meter	0
XJ	Metallurgical junction depth (**LEVEL**=2 or 3)	meter	0
XQC	Fraction of channel charge attributed to drain		1.0

* See .MODEL statement.

ζ in L&W column indicates that parameter may have corresponding parameters exhibiting length and width dependence. See discussion under *Model Level 4* following.

Model Level 4

The **LEVEL**=4 (BSIM1) model parameters are all values obtained from process characterization, and can be generated automatically. Reference [4] describes a means of generating a "process" file, which *must* then be converted into .MODEL statements for inclusion in the Model Library or circuit file. (The simulator *does not* read process files.)

In the following list, parameters marked using a "ζ" in the L&W column also have corresponding parameters with a length and width dependency. For example, VFB is a basic parameter using units of volts, and LVFB and WVFB also exist and have units of volt·μ. The formula

$$P_i = P_0 + P_L / L_e + P_W / W_e$$

is used to evaluate the parameter for the actual device, where

L_e = effective length = L - DL
W_e = effective width = W - DW

Note *Unlike the other models in PSpice, the BSIM model is designed for use with a process characterization system that provides all parameters: there are no defaults specified for the parameters, and leaving one out can cause problems.*

Table 2-24 *MOSFET Level 4 Model Parameters*

Model Parameters*	Description	Units	L&W
DL	Channel shortening	μ	
DW	Channel narrowing	μ	
ETA	Zero-bias drain-induced barrier lowering coefficient		ζ
K1	Body effect coefficient	$volt^{1/2}$	ζ
K2	Drain / source depletion charge sharing coefficient		ζ
MUS	Mobility at zero substrate bias and Vds=Vdd	$cm^2 / volt^2 \cdot sec$	ζ
MUZ	Zero-bias mobility	$cm^2 / volt \cdot sec$	
N0	Zero-bias subthreshold slope coefficient		ζ
NB	Sens. of subthreshold slope to substrate bias		ζ
ND	Sens. of subthreshold slope to drain bias		ζ
PHI	Surface inversion potential	volt	ζ
TEMP	Temperature at which parameters were measured	°C	
TOX	Gate-oxide thickness	μ	
U0	Zero-bias transverse-field mobility degradation	$volt^{-1}$	ζ
U1	Zero-bias velocity saturation	μ / volt	ζ
VDD	Measurement bias range	volts	
VFB	Flat-band voltage	volt	ζ
WDF	Drain, source junction default width	meter	
X2E	Sens. of drain-induced barrier lowering effect to substrate bias	$volt^{-1}$	ζ
X2MS	Sens. of mobility to substrate bias @ Vds=0	$cm^2 / volt^2 \cdot sec$	ζ
X2MZ	Sens. of mobility to substrate bias @ Vds=0	$cm^2 / volt^2 \cdot sec$	ζ
X2U0	Sens. of transverse-field mobility degradation effect to substrate bias	$volt^{-2}$	ζ
X2U1	Sens. of velocity saturation effect to substrate bias	$\mu / volt^2$	ζ

Table 2-24 *MOSFET Level 4 Model Parameters (continued)*

Model Parameters*	Description	Units	L&W
X3E	Sens. of drain-induced barrier lowering effect to drain bias @ Vds = Vdd	$volt^{-1}$	ζ
X3MS	Sens. of mobility to drain bias @ Vds=Vdd	$cm^2/volt^2 \cdot sec$	ζ
X3U1	Sens. of velocity saturation effect on drain	$\mu/volt^2$	ζ
XPART	Gate-oxide capacitance charge model flag. **XPART**=0 selects a 40/60 drain/source charge partition in saturation, while **XPART**=1 selects a 0/100 drain/source charge partition.		

* See .MODEL statement

ζ in L&W column indicates that parameter may have corresponding parameters exhibiting length and width dependence. See discussion under *Model Level 4* on page 2-67.

Model Level 6 (BSIM3 - Version 2.0)

The BSIM3 model is a physical model using extensive built-in dependencies of important dimensional and processing parameters. It includes the major effects that are important to modeling deep-submicrometer MOSFETs such as threshold voltage reduction, nonuniform doping, mobility reduction due to the vertical field, bulk charge effect, carrier velocity saturation, drain-induced barrier lowering (DIBL), channel length modulation (CLM), hot-carrier-induced output resistance reduction, subthreshold conduction, source/drain parasitic resistance, substrate current induced body effect (SCBE), and drain voltage reduction in LDD structure. More detailed model information is available in reference [7].

Table 2-25 *MOSFET Level 6 Model Parameters*

Model Parameters	Description	Units	Default	Note*
A0	Bulk charge effect coefficient NMOS		1.0	
	Bulk charge effect coefficient PMOS		4.4	
A1	First non-saturation coefficient NMOS	1/V	0.0	
	First non-saturation coefficient PMOS	1/V	0.23	
A2	Second non-saturation coefficient NMOS		1.0	
	Second non-saturation coefficient PMOS		0.08	
AT	Saturation velocity temperature coefficient	m/sec	3.3E4	
BULKMOD	Bulk charge model selector:			
	NMOS		1	
	PMOS		2	
CDSC	Drain/source and channel coupling capacitance	F/m^2	2.4E-4	
CDSCB			0.0	
DL	Channel length reduction on one side	m	0.0	
DROUT	Channel length dependent coefficient of the DIBL effect on Rout		0.56	
DSUB	Subthreshold DIBL coefficient exponent		DROUT	

Table 2-25 *MOSFET Level 6 Model Parameters (continued)*

Model Parameters	Description	Units	Default	Note*
DVT0	First coefficient of short-channel effect on threshold voltage		2.2	
DVT1	Second coefficient of short-channel effect on threshold voltage		0.53	
DVT2	Body bias coefficient of short-channel effect on threshold voltage	1/V	-0.032	
DW	Channel width reduction on one side	m	0.0	
ETA0	DIBL coefficient in subthreshold region		0.08	
ETAB	Body bias coefficient for the subthreshold DIBL coefficient	1/V	-0.07	
K1	First-order body effect coefficient	$\sqrt{V}$	calculated	1
K2	Second-order body effect coefficient		calculated	1
K3	Narrow width effect coefficient		80.0	
K3B			0.0	
KETA	Body bias coefficient of the bulk charge effect.	1/V	-0.047	
KT1	Temperature coefficient for threshold voltage	V	-0.11	
KT1L	Channel length sensitivity of temperature coefficient for threshold voltage.	V-m	0.0	
KT2	Body bias coefficient of the threshold voltage temperature effect		0.022	
NFACTOR	Subthreshold swing coefficient		1.0	
NGATE	Poly gate doping concentration	$1/cm^3$		
NLX	Lateral nonuniform doping coefficient	m	1.74E-7	
NPEAK	Peak doping concentration near interface	$1/cm^3$	1.7E17	
NSUB	Substrate doping concentration	$1/cm^3$	6.0E16	
PCLM	Channel length modulation coefficient		1.3	
PDIBL1	First output resistance DIBL effect coefficient		0.39	

Table 2-25 *MOSFET Level 6 Model Parameters (continued)*

Model Parameters	Description	Units	Default	Note*
PDIBL2	Second output resistance DIBL effect coefficient		0.0086	
PSCBE1	First substrate current body effect coefficient	V / m	4.24E8	
PSCBE2	Second substrate current body effect coefficient	m / V	1.0E-5	
PVAG			0.0	
RDS0	Contact resistance	ohms	0.0	
RDSW	Parasitic resistance per unit width	ohms / µm	0.0	
SATMOD	Saturation model selector: Semi-empirical output resistance model 1 Physical output resistance model 2		2	
SUBTHMOD	Subthreshold model selector: No subthreshold model 0 BSIM1 subthreshold model 1 BSIM3 subthreshold model 2 BSIM3 subthreshold model using log current 3		2	
TNOM	Temperature at which parameters are extracted.	deg. C	27	
TOX	Gate oxide thickness	m	1.5E-8	
UA	First-order mobility degradation coefficient	m / V	2.25E-9	2
UA1	Temperature coefficient for **UA**	m / V	4.31E-9	2
UB	Second-order mobility degradation coefficient	$(m / V)^2$	5.87E-19	2
UB1	Temperature coefficient for **UB**	$(m / V)^2$	-7.61E-18	2
UC	Body effect mobility degradation coefficient	1 / V	0.0465	2
UC1	Temperature coefficient for **UC**	1 / V	-0.056	2
UTE	Mobility temperature exponent		-1.5	
VOFF	Offset voltage in subthreshold region	V	-0.11	

Table 2-25 *MOSFET Level 6 Model Parameters (continued)*

Model Parameters	Description	Units	Default	Note*
VSAT	Saturation velocity at Temp=**TNOM**	cm / sec	8.0E6	
VTH0	Threshold voltage at Vbs=0 for large channel length	V	calculated	1
W0	Narrow width effect parameter	m	2.5E-6	
XJ	Junction depth	m	1.5E-7	
XPART	Charge partitioning coefficient: No charge model < 0.0 40 / 60 partition = 0.0 50 / 50 partition = 0.5 0 / 100 partition = 1.0		0.0	

* See *Notes* on page 2-74 for Note references.

The following parameters presented in Table 2-26 are “expert parameters”. These should not be changed unless the detail structure of the device is known having specified meaningful values.

Table 2-26 *MOSFET Level 6 “Expert Parameters”*

Model Parameters	Description	Units	Default	Note*
CIT	Capacitance due to interface trapped charge	F / m2	0.0	
EM	Critical electrical field in channel	V / m	4.1E7	
ETA	Drain voltage reduction coefficient due to LDD		0.3	
GAMMA1	Body effect coefficient near the interface	$\sqrt{V}$	calculated	1
GAMMA2	Body effect coefficient in the bulk	$\sqrt{V}$	calculated	1
LDD	Total length of the LDD region	m	0.0	
LITL	Characteristic length related to current depth	m	calculated	1
PHI	Surface potential under strong inversion	V	calculated	1
U0	Mobility at Temp=TNOM: NMOS PMOS	 cm2 / V-sec cm2 / V-sec	 670.0 250.0	
VBM	Maximum applied body bias	V	-5.0	

Table 2-26 *MOSFET Level 6 "Expert Parameters" (continued)*

Model Parameters	Description	Units	Default	Note*
VBX	Vbs at which the depletion width equals XT	V	calculated	1
VFB	Flat-band voltage	V	calculated	1
VGHIGH	Voltage shift of the higher bound of the transition region	V	0.12	
VGLOW	Voltage shift of the lower bound of the transition region	V	-0.12	
XT	Doping depth	m	1.55E-7	

* See *Notes* on page 2-74 for Note references.

Notes

1 If any of the following BSIM3 Version 2.0 model parameters are not explicitly specified, they are calculated using the following equations.

$$\mathbf{VTH0} = \mathbf{VFB} + \mathbf{PHI} + \mathbf{K}\sqrt{\mathbf{PHI}}$$

$$\mathbf{K1} = \mathbf{GAMMA2} - 2 \cdot \mathbf{K2}\sqrt{(\mathbf{PHI} - \mathbf{VBM})}$$

$$\mathbf{K2} = \frac{(\mathbf{GAMMA1} - \mathbf{GAMMA2})(\sqrt{\mathbf{PHI} - \mathbf{VBX}} - \sqrt{\mathbf{PHI}})}{2\sqrt{\mathbf{PHI}}(\sqrt{\mathbf{PHI} - \mathbf{VBX}} - \sqrt{\mathbf{PHI}}) + \mathbf{VBM}}$$

$$\mathbf{VBF} = \mathbf{VTH0} - \mathbf{PHI} - \mathbf{K1}\sqrt{\mathbf{PHI}}$$

$$\mathbf{PHI} = 2V_{tm}\ln\left(\frac{\mathbf{NPEAK}}{n_i}\right)$$

$$\mathbf{GAMMA1} = \frac{\sqrt{2q\varepsilon_{si}\mathbf{NPEAK}}}{\mathbf{COX}}$$

$$\mathbf{GAMMA2} = \frac{\sqrt{2q\varepsilon_{si}\mathbf{NSUB}}}{\mathbf{COX}}$$

$$\mathbf{VBX} = \mathbf{PHI} - q \cdot \mathbf{NPEAK} \cdot \mathbf{XT}^2/(2\varepsilon_{si})$$

$$\mathbf{LITL} = \sqrt{\frac{\varepsilon_{si}\mathbf{TOX}X_j}{\varepsilon_{ox}}}$$

2 Default values listed in Table 2-25 for the parameters **UA**, **UB**, **UC** **UA1**, **AB1**, and **UC1** are used for simplified mobility modeling.

For All Model Levels

The following list describes the parameters common to all model levels, which are primarily parasitic element values such as series resistance, overlap and junction capacitance, and so on.

Table 2-27 *MOSFET Model Parameters for All Levels*

Model Parameters*	Description	Units	Default
AF	Flicker noise exponent		1
CBD	Zero-bias bulk-drain *p-n* capacitance	farad	0
CBS	Zero-bias bulk-source *p-n* capacitance	farad	0
CGBO	Gate-bulk overlap capacitance / channel length	farad / meter	0
CGDO	Gate-drain overlap capacitance / channel width	farad / meter	0
CGSO	Gate-source overlap capacitance / channel width	farad / meter	0
CJ	Bulk *p-n* zero-bias bottom capacitance / area	farad / meter^2	0
CJSW	Bulk *p-n* zero-bias sidewall capacitance / length	farad / meter	0
FC	Bulk *p-n* forward-bias capacitance coefficient		0.5
IS	Bulk *p-n* saturation current	amp	1E-14
JS	Bulk *p-n* saturation current / area	amp / meter^2	0
JSSW	Bulk *p-n* saturation sidewall current / length	amp / meter	0
KF	Flicker noise coefficient		0
L	Channel length	meter	**DEFL**
LEVEL	Model index		1
MJ	Bulk *p-n* bottom grading coefficient		0.5
MJSW	Bulk *p-n* sidewall grading coefficient		0.33
N	Bulk *p-n* emission coefficient		1
PB	Bulk *p-n* bottom potential	volt	0.8

Table 2-27 *MOSFET Model Parameters for All Levels (continued)*

Model Parameters*	Description	Units	Default
PBSW	Bulk *p-n* sidewall potential	volt	**PB**
RB	Bulk ohmic resistance	ohm	0
RD	Drain ohmic resistance	ohm	0
RDS	Drain-source shunt resistance	ohm	infinite
RG	Gate ohmic resistance	ohm	0
RS	Source ohmic resistance	ohm	0
RSH	Drain, source diffusion sheet resistance	ohm / square	0
TT	Bulk *p-n* transit time	sec	0
T_ABS	Absolute temperature	°C	
T_MEASURED	Measured temperature	°C	
T_REL_GLOBAL	Relative to current temperature	°C	
T_REL_LOCAL	Relative to AKO model temperature	°C	
W	Channel width	meter	**DEFW**

* For information on **T_MEASURED**, **T_ABS**, **T_REL_GLOBAL**, and **T_REL_LOCAL**, see the .MODEL statement on page 1-25.

Equations

In the following equations:

Vgs = intrinsic gate-intrinsic source voltage
Vgd = intrinsic gate-intrinsic drain voltage
Vds = intrinsic drain-intrinsic source voltage
Vbs = intrinsic substrate-intrinsic source voltage
Vbd = intrinsic substrate-intrinsic drain voltage
Vt = $k \cdot T / q$ (thermal voltage)
k = Boltzmann's constant
q = electron charge
T = analysis temperature (°K)
Tnom = nominal temperature (set using TNOM option)

Other variables are from the model parameter list. These equations describe an N-channel MOSFET. For P-channel devices, reverse the signs of all voltages and currents. Positive current is current flowing into a terminal (for example, positive drain current flows from the drain through the channel to the source).

DC Currents [1]

Ig = gate current = 0

Ib = bulk current = Ibs+Ibd

Ibs = bulk-source leakage current = $Iss \cdot (e^{Vbs/(N \cdot Vt)} - 1)$

Ibd = bulk-drain leakage current = $Ids \cdot (e^{Vbd/(N \cdot Vt)} - 1)$

where if: **JS** = 0, or AS = 0, or AD = 0

Iss = IS

Ids = IS

otherwise:

Iss = AS·**JS** + PS·**JSSW**

Ids = AD·**JS** + PD·**JSSW**

Id = drain current = Idrain-Ibd

Is = source current = -Idrain-Ibs

Equations for Idrain: LEVEL=1

For: $Vds \geq 0$ (normal mode)

and: $Vgs - V_{to} < 0$ (cutoff region)

Idrain = 0

and: $Vds < Vgs - V_{to}$ (linear region)

Idrain =

$(W/L) \cdot (\mathbf{KP}/2) \cdot (1 + \mathbf{LAMBDA} \cdot Vds) \cdot Vds \cdot (2 \cdot (Vgs - V_{to}) - Vds)$

and: $0 \leq Vgs - V_{to} \leq Vds$ (saturation region)

Idrain = $(W/L) \cdot (\mathbf{KP}/2) \cdot (1 + \mathbf{LAMBDA} \cdot Vds) \cdot (Vgs - V_{to})^2$

where $V_{to} = \mathbf{VTO} + \mathbf{GAMMA} \cdot ((\mathbf{PHI} - Vbs)^{1/2} - \mathbf{PHI}^{1/2})$

For: Vds < 0 (inverted mode)

Switch the source and drain in equations (above).

For **LEVEL**=2, or **LEVEL**=3 MOSFET models, see reference [2] on 2-49 for detailed information.

1. Positive current is current flowing into a terminal.

Capacitance [1]

Cbs = bulk-source capacitance = area cap. + sidewall cap. + transit time cap.

Cbd = bulk-drain capacitance = area cap. + sidewall cap. + transit time cap.

For: **CBS** = 0 and **CBD** = 0

Cbs = AS·**CJ**·Cbsj + PS·**CJSW**·Cbss + **TT**·Gbs

Cbd = AD·**CJ**·Cbdj + PD·**CJSW**·Cbds + **TT**·Gds

otherwise

Cbs = **CBS**·Cbsj + PS·**CJSW**·Cbss + **TT**·Gbs

Cbd = **CBD**·Cbdj + PD·**CJSW**·Cbds + **TT**·Gds

where

Gbs = DC bulk-source conductance = *d*Ibs/*d*Vbs

Gbd = DC bulk-drain conductance = *d*Ibd/*d*Vbd

or: Vbs ≤ **FC**·**PB**

$Cbsj = (1-Vbs/\mathbf{PB})^{-MJ}$

$Cbss = (1-Vbs/\mathbf{PBSW})^{-MJSW}$

For: Vbs > **FC**·**PB**

$Cbsj = (1-\mathbf{FC})^{-(1+MJ)}\cdot(1-\mathbf{FC}\cdot(1+\mathbf{MJ})+\mathbf{MJ}\cdot Vbs/\mathbf{PB})$

$Cbss = (1-\mathbf{FC})^{-(1+MJSW)}\cdot(1-\mathbf{FC}\cdot(1+\mathbf{MJSW})+\mathbf{MJSW}\cdot Vbs/\mathbf{PBSW})$

For: Vbd ≤ **FC**·**PB**

$Cbdj = (1-Vbd/\mathbf{PB})^{-MJ}$

$Cbds = (1-Vbd/\mathbf{PBSW})^{-MJSW}$

For: Vbd > **FC**·**PB**

$Cbdj = (1-\mathbf{FC})^{-(1+MJ)}\cdot(1-\mathbf{FC}\cdot(1+\mathbf{MJ})+\mathbf{MJ}\cdot Vbd/\mathbf{PB})$

$Cbds = (1-\mathbf{FC})^{-(1+MJSW)}\cdot(1-\mathbf{FC}\cdot(1+\mathbf{MJSW})$

$+\mathbf{MJSW}\cdot Vbd/\mathbf{PBSW})$

Cgs = gate-source overlap capacitance = **CGSO**·W

Cgd = gate-drain overlap capacitance = **CGDO**·W

Cgb = gate-bulk overlap capacitance = **CGBO**·L

For MOSFETs the capacitance model has been changed to conserve charge. This change affects the level 1, 2, and 3 models. The level 4 (BSIM) and level 6 (BSIM3) models have their own capacitance model, which already conserves charge and remains unchanged. See reference [6] and reference [7] on page 82 for the equations describing the capacitances due to the channel charge.

1. All capacitances are between terminals of the intrinsic MOSFET. That is, to the inside of the ohmic drain and source resistances.

Temperature Effects

$$\mathbf{IS}(T) = \mathbf{IS}\cdot e^{(Eg(Tnom)\cdot T/Tnom - Eg(T))/Vt}$$

$$\mathbf{JS}(T) = \mathbf{JS}\cdot e^{(Eg(Tnom)\cdot T/Tnom - Eg(T))/Vt}$$

$$\mathbf{JSSW}(T) = \mathbf{JSSW}\cdot e^{(Eg(Tnom)\cdot T/Tnom - Eg(T))/Vt}$$

$$\mathbf{PB}(T) = \mathbf{PB}\cdot T/Tnom - 3\cdot Vt\cdot ln(T/Tnom) - Eg(Tnom)\cdot T/Tnom + Eg(T)$$

$$\mathbf{PBSW}(T) = \mathbf{PBSW}\cdot T/Tnom - 3\cdot Vt\cdot ln(T/Tnom) - Eg(Tnom)\cdot T/Tnom + Eg(T)$$

$$\mathbf{PHI}(T) = \mathbf{PHI}\cdot T/Tnom - 3\cdot Vt\cdot ln(T/Tnom) - Eg(Tnom)\cdot T/Tnom + Eg(T)$$

where $Eg(T)$ = silicon bandgap energy = $1.16 - .000702\cdot T^2/(T+1108)$

$$\mathbf{CBD}(T) = \mathbf{CBD}\cdot(1+\mathbf{MJ}\cdot(.0004\cdot(T-Tnom)+(1-\mathbf{PB}(T)/\mathbf{PB})))$$

$$\mathbf{CBS}(T) = \mathbf{CBS}\cdot(1+\mathbf{MJ}\cdot(.0004\cdot(T-Tnom)+(1-\mathbf{PB}(T)/\mathbf{PB})))$$

$$\mathbf{CJ}(T) = \mathbf{CJ}\cdot(1+\mathbf{MJ}\cdot(.0004\cdot(T-Tnom)+(1-\mathbf{PB}(T)/\mathbf{PB})))$$

$$\mathbf{CJSW}(T) = \mathbf{CJSW}\cdot(1+\mathbf{MJSW}\cdot(.0004\cdot(T-Tnom)+(1-\mathbf{PB}(T)/\mathbf{PB})))$$

$$\mathbf{KP}(T) = \mathbf{KP}\cdot(T/Tnom)^{-3/2}$$

$$\mathbf{UO}(T) = \mathbf{UO}\cdot(T/Tnom)^{-3/2}$$

$$\mathbf{MUS}(T) = \mathbf{MUS}\cdot(T/Tnom)^{-3/2}$$

$$\mathbf{MUZ}() = \mathbf{MUZ}\cdot(T/Tnom)^{-3/2}$$

$$\mathbf{X3MS}(T) = \mathbf{X3MS}\cdot(T/Tnom)^{-3/2}$$

The ohmic (parasitic) resistances have no temperature dependence.

Noise

Noise is calculated assuming a one hertz bandwidth, using the following spectral power densities (per unit bandwidth):

the parasitic resistances (Rd, Rg, Rs, and Rb) generate thermal noise ...

$Id^2 = 4 \cdot k \cdot T/Rd$
$Ig^2 = 4 \cdot k \cdot T/Rg$
$Is^2 = 4 \cdot k \cdot T/Rs$
$Ib^2 = 4 \cdot k \cdot T/Rb$

the intrinsic MOSFET generates shot and flicker noise ...

$Idrain^2 = 4 \cdot k \cdot T \cdot gm \cdot 2/3 + \mathbf{KF} \cdot Idrain^{AF}/(FREQUENCY \cdot Kchan)$

where

gm = *d*Idrain/*d*Vgs (at the DC bias point)

Kchan = (effective length)^2·(permittivity of SiO_2)/**TOX**

References

For a more complete description of the MOSFET models, refer to:

[1] H. Shichman and D. A. Hodges, “Modeling and simulation of insulated-gate field-effect transistor switching circuits,” *IEEE Journal of Solid-State Circuits*, SC-3, 285, September 1968.

[2] A. Vladimirescu, and S. Lui, “The Simulation of MOS Integrated Circuits Using SPICE2,” Memorandum No. M80/7, February 1980.

[3] B. J. Sheu, D. L. Scharfetter, P.-K. Ko, and M.-C. Jeng, “BSIM: Berkeley Short-Channel IGFET Model for MOS Transistors,” *IEEE Journal of Solid-State Circuits*, SC-22, 558-566, August 1987.

[4] J. R. Pierret, “A MOS Parameter Extraction Program for the BSIM Model,” Memorandum No. M84/99 and M84/100, November 1984.

[5] P. Antognetti and G. Massobrio, *Semiconductor Device Modeling with SPICE*, McGraw-Hill, 1993.

[6] Ping Yang, Berton Epler, and Pallab K. Chatterjee, “An Investigation of the Charge Conservation Problem for MOSFET Circuit Simulation,” *IEEE Journal of Solid-State Circuits*, Vol. **SC-18**, No.1, February 1983.

[7] J.H. Huang, Z.H. Liu, M.C. Jeng, K. Hui, M. Chan, P.K. KO, and C. Hu, “BSIM3 Manual,” Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720.

References [2] and [4] are available for $10.00 (each) by sending a check payable to *The Regents of the University of California* to this address:

Cindy Manly
EECS/ERL Industrial Support Office
497 Cory Hall
University of California
Berkeley, CA 94720

Bipolar Transistor

General Form

Q<*name*> < *collector node*> <*base node*> <*emitter node*>
+ [*substrate node*] <*model name*> [*area value*]

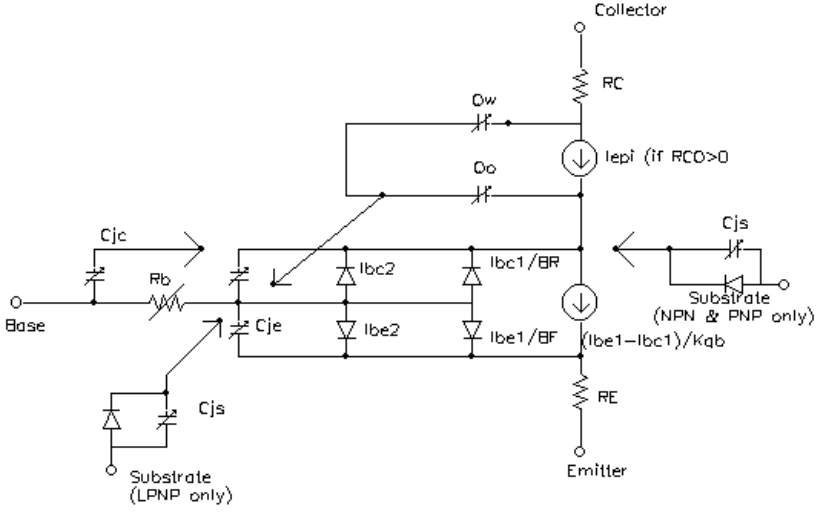
Examples

```
Q1 14 2 13 PNPNOM
Q13 15 3  0 1 NPNSTRONG 1.5
Q7 VC 5 12 [SUB] LATPNP
```

Model Form

```
.MODEL <model name> NPN [model parameters]
.MODEL <model name> PNP [model parameters]
.MODEL <model name> LPNP [model parameters]
```

Figure 2-11 *Bipolar Transistor Model (enhanced Gummel-Poon)*



As shown, the bipolar transistor is modeled as an intrinsic transistor using ohmic resistances in series with the collector (**RC** / *area*), the base (value varies with current, see equations below), and with the emitter (**RE** / *area*). Positive current is current flowing into a terminal. The [*area value*] is the relative device area and defaults to 1. For those model parameters which have alternate names, such as **VAF** and **VA** (the alternate name is shown

by using parentheses), either name can be used.

The substrate node is optional, and if not specified it defaults to ground. Because the simulator allows alphanumeric names for nodes, and because there is no easy way to distinguish these from the model names, it makes it necessary to enclose the name (not a number) used for the substrate node using square brackets "[ ]". Otherwise it is interpreted as a model name. See the third example.

For model types NPN and PNP, the isolation junction capacitance is connected between the intrinsic-collector and substrate nodes. This is the same as in SPICE2, or SPICE3, and works well for vertical IC transistor structures. For lateral IC transistor structures there is a third model, LPNP, where the isolation junction capacitance is connected between the intrinsic-base and substrate nodes.

Table 2-28 *Bipolar Transistor Model Parameters*

Model Parameters*	Description	Units	Default
AF	Flicker noise exponent		1
BF	Ideal maximum forward beta		100
BR	Ideal maximum reverse beta		1
CJC	Base-collector zero-bias *p-n* capacitance	farad	0
CJE	Base-emitter zero-bias *p-n* capacitance	farad	0
CJS (CCS)	Substrate zero-bias *p-n* capacitance	farad	0
EG	Bandgap voltage (barrier height)	eV	1.11
FC	Forward-bias depletion capacitor coefficient		0.5
GAMMA	Epitaxial region doping factor		1E-11
IKF (IK)	Corner for forward-beta high-current roll-off	amp	infinite
IKR	Corner for reverse-beta high-current roll-off	amp	infinite
IRB	Current at which Rb falls halfway to	amp	infinite
IS	Transport saturation current	amp	1E-16
ISC (C4)	Base-collector leakage saturation current	amp	0

Table 2-28 *Bipolar Transistor Model Parameters (continued)*

Model Parameters*	Description	Units	Default
ISE (C2)	Base-emitter leakage saturation current	amp	0
ISS	Substrate *p-n* saturation current	amp	0
ITF	Transit time dependency on Ic	amp	0
KF	Flicker noise coefficient		0
MJC (MC)	Base-collector *p-n* grading factor		0.33
MJE (ME)	Base-emitter *p-n* grading factor		0.33
MJS (MS)	Substrate *p-n* grading factor		0
NC	Base-collector leakage emission coefficient		2
NE	Base-emitter leakage emission coefficient		1.5
NF	Forward current emission coefficient		1
NK	High-current roll-off coefficient		.5
NR	Reverse current emission coefficient		1
NS	Substrate *p-n* emission coefficient		1
PTF	Excess phase @ $1/(2\pi \cdot TF)$Hz	degree	0
QCO	Epitaxial region charge factor	coulomb	0
RB	Zero-bias (maximum) base resistance	ohm	0
RBM	Minimum base resistance	ohm	**RB**
RC	Collector ohmic resistance	ohm	0
RCO	Epitaxial region resistance	ohm	0
RE	Emitter ohmic resistance	ohm	0
TF	Ideal forward transit time	sec	0
TR	Ideal reverse transit time	sec	0
TRB1	RB temperature coefficient (linear)	$°C^{-1}$	0
TRB2	RB temperature coefficient (quadratic)	$°C^{-2}$	0
TRC1	RC temperature coefficient (linear)	$°C^{-1}$	0
TRC2	RC temperature coefficient (quadratic)	$°C^{-2}$	0
TRE1	RE temperature coefficient (linear)	$°C^{-1}$	0

Table 2-28 *Bipolar Transistor Model Parameters (continued)*

Model Parameters*	Description	Units	Default
TRE2	RE temperature coefficient (quadratic)	$°C^{-2}$	0
TRM1	RBM temperature coefficient (linear)	$°C^{-1}$	0
TRM2	RBM temperature coefficient (quadratic)	$°C^{-2}$	0
T_ABS	Absolute temperature	°C	
T_MEASURED	Measured temperature	°C	
T_REL_GLOBAL	Relative to current temperature	°C	
T_REL_LOCAL	Relative to AKO model temperature	°C	
VAF (VA)	Forward Early voltage	volt	infinite
VAR (VB)	Reverse Early voltage	volt	infinite
VJC (PC)	Base-collector built-in potential	volt	0.75
VJE (PE)	Base-emitter built-in potential	volt	0.75
VJS (PS)	Substrate *p-n* built-in potential	volt	0.75
VO	Carrier mobility “knee” voltage	volt	10
VTF	Transit time dependency on Vbc	volt	infinite
XCJC	Fraction of **CJC** connected internally to Rb		1
XCJC2	Fraction of **CJC** connected internally to Rb		1
XTB	Forward and reverse beta temperature coefficient		0
XTF	Transit time bias dependence coefficient		0
XTI (PT)	IS temperature effect exponent		3

* For information on **T_MEASURED**, **T_ABS**, **T_REL_GLOBAL**, and **T_REL_LOCAL**, see the .MODEL statement on page 1-25.

The parameters **ISE** (**C2**) and **ISC** (**C4**) can be set to be greater than one. In this case, they are interpreted as multipliers of **IS** instead of absolute currents: that is, if **ISE** is greater than one then it is replaced by **ISE**·**IS**. Likewise for **ISC**.

If the model parameter **RCO** is specified, then quasi-saturation effects are included.

The model parameter **XCJC2** is used like of **XCJC**. The differences between the two parameters are as follows.

Table 2-29 *How XCJC and XCJC2 Specify the Distribution of the CJC Capacitance*

Branch	XCJC	XCJC2
Intrinsic base to intrinsic collector	**XCJC*CJC**	**XCJC2*CJC**
Extrinsic base to **intrinsic** collector	**(1.0 – XCJC)*CJC**	not applicable
Extrinsic base to **extrinsic** collector	not applicable	**(1.0 – XCJC2)*CJC**

When **XCJC2** is specified in the range 0 < **XCJC2** < 1.0, **XCJC** is ignored. Also, the extrinsic base to extrinsic collector capacitance (Cbx2) and the gain-bandwidth product (Ft2) are included in the operating point information (in the output listing generated during a Bias Point Detail analysis, **.OP**). For backward compatibility, the parameter **XCJC** and the associated calculation of Cbx and Ft remain unchanged. Cbx and Ft appears in the output listing only when **XCJC** is specified.

The use of **XCJC2** produces more accurate results because Cbx2 (the fraction of **CJC** associated with the intrinsic collector node) now equals the ratio of the device's emitter area-to-base area. This results in a better correlation between the measured data and the gain bandwidth product (Ft2) calculated by PSpice.

Equations

In the following equations:

Vbe = intrinsic base-intrinsic emitter voltage
Vbc = intrinsic base-intrinsic collector voltage
Vbs = intrinsic base-substrate voltage
Vbw = intrinsic base-extrinsic collector voltage (quasi-saturation only)
Vbx = extrinsic base-intrinsic collector voltage
Vce = intrinsic collector-intrinsic emitter voltage
Vjs = (NPN) intrinsic collector-substrate voltage
= (PNP) intrinsic substrate-collector voltage
= (LPNP) intrinsic base-substrate voltage
Vt = $k \cdot T / q$ (thermal voltage)
k = Boltzmann's constant
q = electron charge
T = analysis temperature (°K)
Tnom = nominal temperature (set using TNOM option)

Other variables are from the model parameter list. These equations describe an NPN transistor. For the PNP and LPNP devices, reverse the signs of all voltages and currents.

DC Currents [1]

Ib = base current = *area*·(Ibe1/**BF** + Ibe2 + Ibc1/**BR** + Ibc2)

Ic = collector current
= *area*·(Ibe1/Kqb - Ibc1/Kqb - Ibc1/**BR** - Ibc2)
Ibe1 = forward diffusion current = **IS**·($e^{Vbe/(NF\cdot Vt)}$-1)
Ibe2 = non-ideal base-emitter current = **ISE**·($e^{Vbe/(NE\cdot Vt)}$-1)
Ibc1 = reverse diffusion current = **IS**·($e^{Vbc/(NR\cdot Vt)}$-1)
Ibc2 = non-ideal base-collector current
= **ISC**·($e^{Vbc/(NC\cdot Vt)}$-1)
Kqb = base charge factor = Kq1·(1+(1+4·Kq2)^{NK})/2
Kq1 = 1/(1 - Vbc/**VAF** - Vbe/**VAR**)
Kq2 = Ibe1/**IKF** + Ibc1/**IKR**

Is = substrate current = *area*·**ISS**·($e^{Vjs/(NS\cdot Vt)}$-1)

Rb = actual base parasitic resistance
For: **IRB** = infinite (default value)
Rb = (**RBM** + (**RB**-**RBM**)/Kqb)/*area*
For: IRB > 0
Rb = (**RBM** + 3·(**RB**-**RBM**)· $\frac{\tan(x)-x}{x\cdot(\tan(x))^2}$)/*area*

where *x* = $\frac{(1+(144/\pi^2)\cdot Ib/(area\cdot \mathbf{IRB}))^{1/2}-1}{(24/\pi^2)\cdot(Ib/(area\cdot \mathbf{IRB}))^{1/2}}$

1. Positive current is current flowing into a terminal.

Capacitances [1]

Cbe = base-emitter capacitance = Ctbe + *area*·Cjbe

Ctbe = transit time capacitance = tf·Gbe

tf = effective **TF**

= **TF**·(1+**XTF**·(Ibe1/(Ibe1+*area*·**ITF**))^2·$e^{Vbc/(1.44\cdot VTF)}$)

Gbe = DC base-emitter conductance = (*d*Ibe)/(*d*Vb)

Ibe = Ibe1 + Ibe2

For: Vbe ≤ **FC**·**VJE**

Cjbe = **CJE**·(1-Vbe/**VJE**)$^{-MJE}$

For: Vbe > **FC**·**VJE**

Cjbe = **CJE**·(1-**FC**)$^{-(1+MJE)}$·(1-**FC**·(1+**MJE**)
+**MJE**·Vbe/**VJE**)

Cbc = base-collector capacitance = Ctbc + *area*·**XCJC**·Cjbc

Ctbc = transit time capacitance = **TR**·Gbc

Gbc = DC base-collector conductance
= (*d*Ibc)/(*d*Vbc)

For: Vbc < **FC**·**VJC**

Cjbc = **CJC**·(1-Vbc/**VJC**)$^{-MJC}$

For: Vbc > **FC**·**VJC**

Cjbc = **CJC**·(1-**FC**)$^{-(1+MJC)}$·(1-**FC**·(1+**MJC**)+**MJC**·Vbc/**VJC**)

Cbx = extrinsic-base to intrinsic-collector capacitance
= *area*·(1-**XCJC**)·Cjbx

For: Vbx ≤ **FC**·**VJC**

Cjbx = **CJC**·(1-Vbx/**VJC**)$^{-MJC}$

For: Vbx > **FC**·**VJC**

Cjbx = **CJC**·(1-**FC**)$^{-(1+MJC)}$·(1-**FC**·(1+**MJC**)+**MJC**·Vbx/**VJC**)

Cjs = substrate junction capacitance = *area*·Cjjs

For: Vjs ≤ 0

Cjjs = **CJS**·(1-Vjs/**VJS**)$^{-MJS}$(assumes **FC** = 0)

For: Vjs > 0

Cjjs = **CJS**·(1+**MJS**·Vjs/**VJS**)

1. All capacitances, except Cbx, are between terminals of the intrinsic transistor which is inside of the collector, base, and emitter parasitic resistances. Cbx is between the intrinsic collector and the extrinsic base.

Quasi-saturation Effect

Quasi-saturation is an operating region where the internal base-collector metallurgical junction is forward biased, while the external base-collector terminal remains reverse biased.

This effect is modeled by extending the intrinsic Gummel-Poon model, adding a new internal node, a controlled current source, Iepi, and two controlled capacitances, represented by the charges Qo and Qw. These additions are only included if the model parameter **RCO** is specified. See reference [3] for the derivation of this extension.

Iepi = *area*·
(**VO**·(Vt·(K(Vbc)-K(Vbn)-*ln*((1+K(Vbc))/(1+K(Vbn))))
+Vbc-Vbn))/**RCO**·(|Vbc-Vbn|+**VO**)

Qo = *area*·**QCO**·(K(Vbc) - 1 - **GAMMA**/2)

Qw = *area*·**QCO**·(K(Vbn) - 1 - **GAMMA**/2)
where K(v) = $(1 + \mathbf{GAMMA}\cdot e^{(v/Vt)})^{1/2}$

Temperature Effects

$$\mathbf{IS}(T) = \mathbf{IS}\cdot e^{(T/Tnom-1)\cdot EG/(N\cdot Vt)}\cdot(T/Tnom)^{XTI/N}$$

where N = 1

$$\mathbf{ISE}(T) = (\mathbf{ISE}/(T/Tnom)^{XTB})\cdot e^{(T/Tnom-1)\cdot EG/(NE\cdot Vt)}\cdot(T/Tnom)^{XTI/NE}$$

$$\mathbf{ISC}(T) = (\mathbf{ISC}/(T/Tnom)^{XTB})\cdot e^{(T/Tnom-1)\cdot EG/(NC\cdot Vt)}\cdot(T/Tnom)^{XTI/NC}$$

$$\mathbf{ISS}(T) = (\mathbf{ISS}/(T/Tnom)^{XTB})\cdot e^{(T/Tnom-1)\cdot EG/(NS\cdot Vt)}\cdot(T/Tnom)^{XTI/NS}$$

$$\mathbf{BF}(T) = \mathbf{BF}\cdot(T/Tnom)^{XTB}$$

$$\mathbf{BR}(T) = \mathbf{BR}\cdot(T/Tnom)^{XTB}$$

$$\mathbf{RE}(T) = \mathbf{RE}\cdot(1+\mathbf{TRE1}\cdot(T-Tnom)+\mathbf{TRE2}\cdot(T-Tnom)^2)$$

$$\mathbf{RB}(T) = \mathbf{RB}\cdot(1+\mathbf{TRB1}\cdot(T-Tnom)+\mathbf{TRB2}\cdot(T-Tnom)^2)$$

$$\mathbf{RBM}(T) = \mathbf{RBM}\cdot(1+\mathbf{TRM1}\cdot(T-Tnom)+\mathbf{TRM2}\cdot(T-Tnom)^2)$$

$$\mathbf{RC}(T) = \mathbf{RC}\cdot(1+\mathbf{TRC1}\cdot(T-Tnom)+\mathbf{TRC2}\cdot(T-Tnom)^2)$$

$$\mathbf{VJE}(T) = \mathbf{VJE}\cdot T/Tnom - 3\cdot Vt\cdot ln(T/Tnom) - Eg(Tnom)\cdot T/Tnom + Eg(T)$$

$$\mathbf{VJC}(T) = \mathbf{VJC}\cdot T/Tnom - 3\cdot Vt\cdot ln(T/Tnom) - Eg(Tnom)\cdot T/Tnom + Eg(T)$$

$$\mathbf{VJS}(T) = \mathbf{VJS}\cdot T/Tnom - 3\cdot Vt\cdot ln(T/Tnom) - Eg(Tnom)\cdot T/Tnom + Eg(T)$$

where $Eg(T)$ = silicon bandgap energy
$= 1.16 - .000702\cdot T^2/(T+1108)$

$$\mathbf{CJE}(T) = \mathbf{CJE}\cdot(1+\mathbf{MJE}\cdot(.0004\cdot(T-Tnom)+(1-\mathbf{VJE}(T)/\mathbf{VJE})))$$

$$\mathbf{CJC}(T) = \mathbf{CJC}\cdot(1+\mathbf{MJC}\cdot(.0004\cdot(T-Tnom)+(1-\mathbf{VJC}(T)/\mathbf{VJC})))$$

$$\mathbf{CJS}(T) = \mathbf{CJS}\cdot(1+\mathbf{MJS}\cdot(.0004\cdot(T-Tnom)+(1-\mathbf{VJS}(T)/\mathbf{VJS})))$$

Noise

Noise is calculated assuming a one hertz bandwidth, using the following spectral power densities (per unit bandwidth):

the parasitic resistances generate thermal noise ...

$Ic^2 = 4 \cdot k \cdot T/(RC/area)$

$Ib^2 = 4 \cdot k \cdot T/Rb$

$Ie^2 = 4 \cdot k \cdot T/(RE/area)$

the base and collector currents generate shot and flicker noise ...

$Ib^2 = 2 \cdot q \cdot Ib + KF \cdot Ib^{AF}/FREQUENCY$

$Ic^2 = 2 \cdot q \cdot Ic$

References

For a more complete description of bipolar transistor models, refer to

[1] Ian Getreu, *Modeling the Bipolar Transistor*, Tektronix, Inc. part# 062-2841-00.

Also, for a generally detailed discussion of the U.C. Berkeley SPICE models, including the bipolar transistor.

[2] P. Antognetti and G. Massobrio, *Semiconductor Device Modeling with SPICE*, McGraw-Hill, 1988.

For a description of the extension for the quasi-saturation effect, refer to

[3] G. M. Kull, L. W. Nagel, S. W. Lee, P. Lloyd, E. J. Prendergast, and H. K. Dirks, "A Unified Circuit Model for Bipolar Transistors Including Quasi-Saturation Effects," *IEEE Transactions on Electron Devices*, ED-32, 1103-1113 (1985).

Resistor

General Form

R<*name*> <(+) *node*> <(-) *node*> [*model name*] <*value*>
+ [TC = <TC1> [,<TC2>]]

Examples

```
RLOAD 15 0 2K
R2 1 2 2.4E4 TC=.015,-.003
RFDBCK 3 33 RMOD 10K
```

Model Form

`.MODEL <model name> RES [model parameters]`

(+) and (-) *nodes*

Define the polarity when the resistor has a positive voltage across it.

The first node listed (or pin one in Schematics) is defined as positive. The voltage across the component is therefore defined as the first node voltage less the second node voltage.

Positive current flows from the (+) node through the resistor to the (-) node. Current flow from the first node through the component to the second node is considered positive.

Temperature coefficients for the resistor can be specified in-line, as in the second example. If the resistor *has a model specified*, then the coefficients from the model are used for the temperature updates, otherwise the in-line values are used. In both cases the temperature coefficients default to zero. Expressions *cannot be used* for the in-line coefficients.

[*model name*] If this is included and **TCE** (in the model) *is not specified*, then the resistance is given by the formula

$$<value>\cdot R\cdot(1+\mathbf{TC1}\cdot(T-Tnom)+\mathbf{TC2}\cdot(T-Tnom)^2)$$

where <*value*> is normally positive (though it can be negative, but *not* zero). If [*model name*] is included and TCE (in the model) *is specified*, then the resistance is given by the formula

$$<value>\cdot R\cdot 1.01^{TCE\cdot(T-Tnom)}$$

where <*value*> is normally positive (though it can be negative, but *not* zero). “Tnom” is the nominal temperature (set using TNOM option).

Table 2-30 *Resistor Model Parameters*

Model Parameters*	Description	Units	Default
R	Resistance multiplier		1
TC1	Linear temperature coefficient	$°C^{-1}$	0
TC2	Quadratic temperature coefficient	$°C^{-2}$	0
TCE	Exponential temperature coefficient	%/°C	0
T_ABS	Absolute temperature	°C	
T_MEASURED	Measured temperature	°C	
T_REL_GLOBAL	Relative to current temperature	°C	
T_REL_LOCAL	Relative to AKO model temperature	°C	

* For information on **T_MEASURED**, **T_ABS**, **T_REL_GLOBAL**, and **T_REL_LOCAL**, see the .MODEL statement on page 1-25.

Noise

Noise is calculated assuming a one hertz bandwidth. The resistor generates thermal noise using the following spectral power density (per unit bandwidth)

$$i^2 = 4 \cdot k \cdot T / \text{resistance}$$

Voltage-Controlled Switch

General Form

S<*name*> <(+) *switch node*> <(-) *switch node*>
+ <(+) *controlling node*> <(-) *controlling node*>
+ <*model name*>

Examples

```
S12    13 17  2 0 SMOD
SESET  5  0 15 3 RELAY
```

Model Form

`.MODEL <model name> VSWITCH [model parameters]`

The voltage-controlled switch is a special kind of voltage-controlled resistor.

The resistance between the <(+) *switch node*> and <(-) *switch node*> depends on the voltage between the <(+) *controlling node*> and <(–) *controlling node*>. The resistance varies continuously between the **RON** and **ROFF** model parameters.

A resistance of 1/GMIN is connected between the controlling nodes to keep them from floating. See the .OPTIONS statement (page 1-35) for setting GMIN.

Although very little computer time is required to evaluate switches, during transient analysis the simulator must step through the transition region using a fine enough step size to get an accurate waveform. Applying many transitions can produce long run times when evaluating the other devices in the circuit at each time step.

Table 2-31 *Voltage-Controlled Switch Model Parameters*

Model Parameters*	Description	Units	Default
ROFF	“Off” resistance	ohm	1E+6
RON	“On” resistance	ohm	1.0
VOFF	Control voltage for “off” state	volt	0.0
VON	Control voltage for “on” state	volt	1.0

* See .MODEL statement.

RON and **ROFF** must be greater than zero and less than 1/GMIN.

This switch model was designed to minimize numerical problems. However, there are a few things to consider:

Using double precision numbers, the simulator can only handle a dynamic range of about 12 decades. Making the ratio of **ROFF** to **RON** greater than 1E+12 is not recommended.

Also, it not recommend to make the transition region too narrow. Remember that in the transition region the switch has gain. The narrower the region, the higher the gain and the greater the potential for numerical problems. The smallest allowed value for $|$**VON-VOFF**$|$ is **RELTOL**·(**MAX**($|$**VON**$|$, $|$**VOFF**$|$))+ **VNTOL**.

Equations

In the following equations:

Vc	= voltage across control nodes
Lm	= log-mean of resistor values = $ln((\mathbf{RON}\cdot\mathbf{ROFF})^{1/2})$
Lr	= log-ratio of resistor values = $ln(\mathbf{RON}/\mathbf{ROFF})$
Vm	= mean of control voltages = $(\mathbf{VON}+\mathbf{VOFF})/2$
Vd	= difference of control voltages = **VON-VOFF**
k	= Boltzmann's constant
T	= analysis temperature (°K)

Switch Resistance

Rs = switch resistance

If: **VON** > **VOFF**

For: Vc ≥ **VON**

Rs = **RON**

For: Vc ≤ **VOFF**

Rs = **ROFF**

For: **VOFF** < Vc < **VON**

$$Rs = exp(Lm + 3 \cdot Lr \cdot (Vc - Vm) / (2 \cdot Vd) - 2 \cdot Lr \cdot (Vc - Vm)^3 / Vd^3)$$

If: **VON** < **VOFF**

For: Vc < **VON**

Rs = **RON**

For: Vc > **VOFF**

Rs = **ROFF**

For: **VOFF** > Vc > **VON**

$$Rs = exp(Lm - 3 \cdot Lr \cdot (Vc - Vm) / (2 \cdot Vd) + 2 \cdot Lr \cdot (Vc - Vm)^3 / Vd^3)$$

Noise

Noise is calculated assuming a one hertz bandwidth. The voltage-controlled switch generates thermal noise as if it were a resistor having the same resistance that the switch has at the bias point, using the following spectral power density (per unit bandwidth)

$$i^2 = 4 \cdot k \cdot T / Rs$$

Transmission Line

Ideal Line

General Form

T<*name*> <*A port* (+) *node*> <*A port* (-) *node*>
+ <*B port* (+) *node*> <*B port* (-) *node*>
+ [*model name*]
+ Z0=<*value*> [TD=<*value*>] [F=<*value*> [NL=<*value*>]]
+ IC= <*near voltage*> <*near current*> <*far voltage*> <*far current*>

Lossy Line

General Form

T<*name*> <*A port* (+) *node*> <*A port* (-) *node*>
+ <*B port* (+) *node*> <*B port* (-) *node*>
+ [<*model name*> [*electrical length value*]]
+ LEN=<*value*> R=<*value*> L=<*value*>
+ G=<*value*> C=<*value*>

Examples

```
T1 1 2 3 4 Z0=220 TD=115ns
T2 1 2 3 4 Z0=220 F=2.25MEG
T3 1 2 3 4 Z0=220 F=4.5MEG NL=0.5
T4 1 2 3 4 LEN=1 R=.311 L=.378u G=6.27u C=67.3p
T5 1 2 3 4 TMOD 1
```

Model Form

```
.MODEL <model name> TRN [model parameters]
```

Figure 2-12 *Ideal Transmission Line Model*

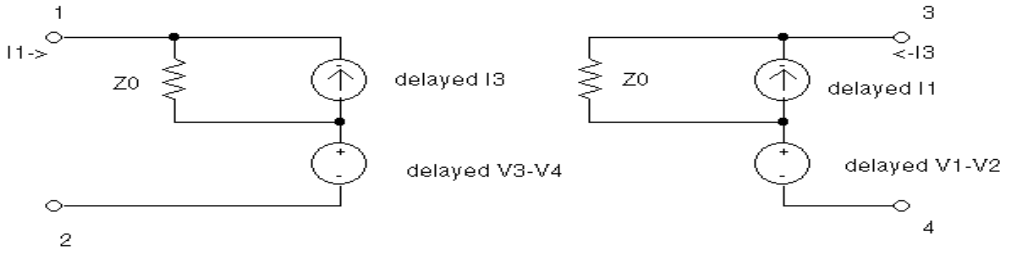


Table 2-32 *Transmission Line Model Parameters*

Model Parameters*	Description	Units**	Default
Model Parameters for Ideal Transmission Lines			
ZO	Characteristic impedance	ohms	none
TD	Transmission delay	seconds	none
F	Frequency for **NL**	Hz	none
NL	Relative wavelength	none	.25
Model Parameters for Lossy Transmission Lines			
R	Per unit length resistance	ohms / unit length	none
L	Per unit length inductance	henries / unit length	none
G	Per unit length conductance	mhos / unit length	none
C	Per unit length capacitance	farads / unit length	none
LEN* **	Physical length	agrees with RLGC *	none
Model Parameters Common to Ideal and Lossy Transmission Lines			
IC	Sets the initial condition and all four values must be entered (This parameter is for both ideal and lossy lines.) (Four values are expected when IC is specified. These are the near-end voltage, the near-end current, the far-end voltage, and the far-end current, given in that order.)		

* See .MODEL statement. The order is, from the most commonly used to the least commonly used parameter.

** Any length units can be used, but they must be consistent. For instance, if LEN is in feet, then the units of R must be in ohms / foot.

*** A lossy line with **R**=**G**=0 and **LEN**=1 is equivalent to an ideal line with $\mathbf{ZO} = \sqrt{\frac{\mathbf{L}}{\mathbf{C}}}$ and $\mathbf{TD} = \mathbf{LEN} \cdot \sqrt{\mathbf{L} \cdot \mathbf{C}}$.

As shown in Figure 2-12, the transmission line device is a bidirectional, delay line. It has two ports, A and B. The (+) and (-) nodes define the polarity of a positive voltage at a port. In Figure 2-12, port A's (+) and (-) nodes are one and two, and port B's (+) and (-) nodes are three and four, respectively.

For the ideal line, IC sets the initial guess for the voltage or current across the ports. The <*near voltage*> value is the voltage across A(+) and A(-) and the <*far voltage*> is the voltage across B(+) and B(-). The <*near current*> is the current through A(+) and A(-) and the <*far current*> is the current through B(+) and B(-).

For the ideal case, Z0 is the characteristic impedance. The transmission line's length can be specified either by TD, a delay in seconds, or by F and NL, a frequency and a relative wavelength at F. NL defaults to 0.25 (F is then the quarter-wave frequency). Although TD and F are both shown as optional, one of the two must be specified. Examples T1, T2, and T3 all specify the same transmission ideal line.

Note *Both Z0 ("zee-zero") and ZO ("zee-oh") are accepted by the simulator.*

During transient (.TRAN) analysis, the internal time step is limited to be no more than one-half the smallest transmission delay, so short transmission lines cause long run times.

For a lossy line, LEN is the electrical length. R, L, G, and C are the per unit length values of resistance, inductance, conductance, and capacitance, respectively. Example T4 specifies a lossy line one meter long. The lossy line model is similar to that shown for the ideal case in Figure 2-13 on page 2-103, except that the delayed voltage and current values include terms which vary with frequency. These terms are computed in transient analysis using an impulse response convolution method, and the internal time step is limited by the time resolution required to accurately model the frequency characteristics of the line. As with ideal lines, short lossy lines cause long run times.

The simulation status window displays the properties of the three shortest transmission lines in a circuit if a transient run's time step ceiling is set more frequently by one of the transmission lines. This is helpful if there is a large number of transmission lines. The properties displayed are:

% loss	Percent attenuation at the characteristic delay (i.e., the degree to which the line is lossy)
time step ceiling	Induced by the line
% of line delay	Time step size at percentage of characteristic delay

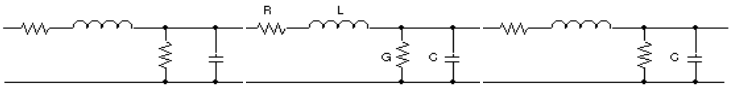
These transmission line properties are displayed only if they are slowing down the simulation.

For a line which uses a model, the electrical length is given after the model name. Example T5 uses TMOD to specify the line parameters and has an electrical length of one unit. All of the transmission line parameters from either the ideal or lossy parameter set can be expressions. In addition, R and G can be general Laplace expressions. This allows the user to model frequency dependent effects, such as skin effect and dielectric loss. However, this adds to the computation time for transient analysis, since the impulse responses must be obtained by an inverse FFT instead of analytically.

The simulator 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 discrete 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 the following figure.

Figure 2-13 *Lossy transmission line lumped line segment*



Lumped Line Segment

This method requires that there is enough lumps to adequately represent the distributed character of the line, and this often results in the need for a large netlist and correspondingly long simulation times. The method also produces spurious oscillations near the natural frequencies of the lumped elements.

An additional extension allows systems of coupled transmission lines to be simulated. Transmission line coupling is specified using the K device. This is done in much the same way that coupling is specified for inductors. See the description of the K device on page 50 for further details.

The distributed model allows freedom from having to determine how many lumps are sufficient, and eliminates the spurious oscillations. It also allows lossy lines to be simulated in a fraction of the time necessary when using the lumped approach, for the same accuracy.

References

For more information on how the lossy transmission line is implemented, refer to:

[1] Roychowdhury and Pederson, “Efficient Transient Simulation of Lossy Interconnect,” Design Automation Conference, 1991.

Current-Controlled Switch

General Form

W<*name*> <(+) *switch node*> <(-) *switch node*>
+ <*controlling V device name*> <*model name*>

Examples

```
W12 13 17 VC WMOD
WRESET  5  0 VRESET RELAY
```

Model Form

.MODEL <*model name*> ISWITCH [*model parameters*]

Table 2-33 *Current-Controlled Switch Model Parameters*

Model Parameters*	Description	Units	Default
IOFF	Control current for “off” state	amp	0.0
ION	Control current for “on” state	amp	1E-3
ROFF	“Off” resistance	ohm	1E+6
RON	“On” resistance	ohm	1.0

* See .MODEL statement.

The current-controlled switch is a special kind of current-controlled resistor.

<*controlling V device name*>

The resistance between the <(+) *switch node* and <(-) *switch node*> depends on the current through <*controlling V device name*>.

The resistance varies continuously between **RON** and **ROFF**.

RON and **ROFF** Must be greater than zero and less than 1/GMIN.

A resistance of 1/GMIN is connected between the controlling nodes to keep them from floating. See the .OPTIONS statement (page 1-35) for setting GMIN.

This model was chosen for a switch to try to minimize numerical problems. However, there are a few things that must be evaluated:

Using double precision numbers, the simulator can handle only a dynamic range of about 12 decades. Therefore, it is not recommended making the ratio of **ROFF** to **RON** greater than 1E+12.

Similarly, it is also not recommended making the transition region too narrow. Remembering that in the transition region the switch has gain. The narrower the region, the higher the gain and the greater the potential for numerical problems. The smallest allowed value for $|\mathbf{ION} - \mathbf{IOFF}|$ is $\mathbf{RELTOL}\cdot(\mathbf{MAX}(|\mathbf{ION}|, |\mathbf{IOFF}|))+\mathbf{ABSTOL}$.

Although very little computer time is required to evaluate switches, during transient analysis the simulator must step through the transition region using a fine enough step size to get an accurate waveform. Having many transitions can produce long run times when evaluating the other devices in the circuit for many times.

In the following equations:

Ic	= controlling current
Lm	= log-mean of resistor values = $ln((\mathbf{RON}\cdot\mathbf{ROFF})^{1/2})$
Lr	= log-ratio of resistor values = $ln(\mathbf{RON}/\mathbf{ROFF})$
Im	= mean of control currents = $(\mathbf{ION}+\mathbf{IOFF})/2$
Id	= difference of control currents = **ION-IOFF**
k	= Boltzmann's constant
T	= analysis temperature (°K)

Switch Resistance

Rs = switch resistance

If: **ION** > **IOFF**

For: Ic > **ION**

Rs = **RON**

For: Ic < **IOFF**

Rs = **ROFF**

For: IOFF < Ic < **ION**

Rs = *exp*(Lm + 3·Lr·(Ic-Im) / (2·Id) - 2·Lr·(Ic-Im)^3 / Id^3)

If: **ION** < **IOFF**

For: Ic < **ION**

Rs = **RON**

For: Ic > **IOFF**

Rs = **ROFF**

For: **IOFF** > Ic > **ION**

Rs = *exp*(Lm - 3·Lr·(Ic-Im) / (2·Id) + 2·Lr·(Ic-Im)^3 / Id^3)

Noise

Noise is calculated assuming a one hertz bandwidth. The current-controlled switch generates thermal noise as if it were a resistor using the same resistance as the switch has at the bias point, using the following spectral power density (per unit bandwidth)

$i^2 = 4 \cdot k \cdot T / Rs$

Subcircuit Instantiation

General Form

X<*name*> [*node*]* <*subcircuit name*> [PARAMS: <<*name*> = <*value*>>*]
+ [TEXT: < <*name*> = <*text value*> >*]

Examples

```
X12   100 101 200 201 DIFFAMP
XBUFF 13  15  UNITAMP
XFOLLOW IN OUT VCC VEE OUT OPAMP
XFELT    1   2   FILTER PARAMS: CENTER=200kHz
X27     A1  A2 A3 Y PLD PARAMS: MNTYMXDLY=1
+ TEXT: JEDEC_FILE=MYJEDEC.JED
XNANDI 25 28 7 MYPWR MYGND PARAMS: IO_LEVEL=2
```

<*subcircuit name*> The <*subcircuit name*> is the name of the subcircuit's definition (see .SUBCKT statement).

There must be the same number of nodes in the call as in the subcircuit's definition. This statement causes the referenced subcircuit to be inserted into the circuit using the given nodes to replace the argument nodes in the definition. It allows a block of circuitry to be defined once and then used in several places.

PARAMS The keyword PARAMS: allows values to be passed into subcircuits as arguments and used in expressions inside the subcircuit.

TEXT The keyword TEXT: allows text values to be passed into subcircuits, and to be used in text expressions inside the subcircuit.

Subcircuit references can be nested. That is, a call can be given to subcircuit A, whose definition contains a call to subcircuit B. The nesting can be to any level, but *must not be circular*: for example, if subcircuit A's definition contains a call to subcircuit B, then subcircuit B's definition must not contain a call to subcircuit A.

IGBT

General Form

Z<*name*> <*collector*> <*gate*> <*emitter*> <*model name*>
+ [AREA=<*value*>] [WB=<*value*>] [AGD=<*value*>]
+ [KP=<*value*>] [TAU=<*value*>]

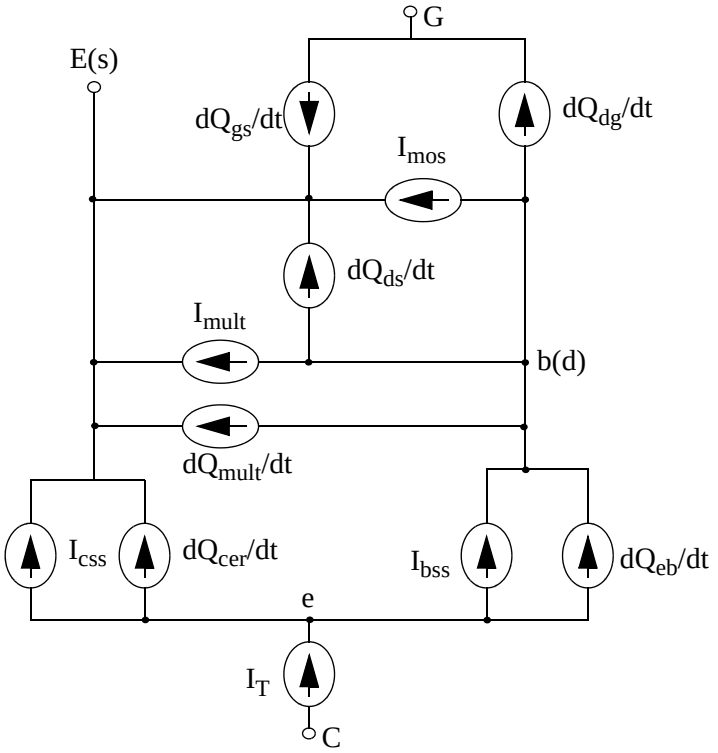
Examples

```
ZDRIVE  1 4 2  IGBTA  AREA=10.1u  WB=91u  AGD=5.1u  KP=0.381
Z231  3 2 9 IGBT27
```

Model Form

`.MODEL <model name> NIGBT [model parameters]`

Figure 2-14 *IGBT Equivalent Circuit*



The equivalent circuit for the IGBT is shown in Figure 2-14. It is modeled as an intrinsic device (not as a subcircuit) and contains five dc current components and six charge (capacitive) components. An overview of the model equations is included below. This overview is not complete. For a more detailed description of the defining equations see references [1] to [4].

Table 2-34 *IGBT Device Parameters*

Device Parameters	Description	Units	Default
AGD	Gate-drain overlap area	m^2	5.0e-6
AREA	Area of the device	m^2	1.0e-5
KP	MOS transconductance	A/V^2	0.38
TAU	Ambipolar recombination lifetime	sec	7.1e-6
WB	Metallurgical base width	m	9.0e-5

The general form of the IGBT syntax allows for the specification of five device parameters.

These device parameters and their associated default values are defined in previous table. The IGBT model parameters and their associated default values are defined in the table that follows. Model parameters can be extracted from data sheet information by using the Parts program. Also, a library of model parameters for commercially available IGBT's is supplied with the software.

The parameters **AGD**, **AREA**, **KP**, **TAU**, and **WB** are specified as both device and model parameters, and they cannot be used in a Monte Carlo analysis.

When specified as device parameters, the assigned values take precedence over those which are specified as model parameters. Also, as device parameters (but not as model parameters), they can be assigned a parameter value and used in conjunction with a .DC or .STEP analysis.

Table 2-35 *IGBT Model Parameters*

Model Parameters*	Description	Units	Default
AGD	Gate-drain overlap area	m^2	5.0e-6
AREA	Area of the device	m^2	1.0e-5
BVF	Avalanche uniformity factor	-	1.0
BVN	Avalanche multiplication exponent	-	4.0
CGS	Gate-source capacitance per unit area	F/cm^2	1.24e-8
COXD	Gate-drain oxide capacitance per unit area	F/cm^2	3.5e-8

Table 2-35 *IGBT Model Parameters*

Model Parameters*	Description	Units	Default
JSNE	Emitter saturation current density	A/cm^2	6.5e-13
KF	Triode region factor	-	1.0
KP	MOS transconductance	A/V^2	0.38
MUN	Electron mobility	$cm^2/(V{\cdot}s)$	1.5e3
MUP	Hole mobility	$cm^2/(V{\cdot}s)$	4.5e2
NB	Base doping	$1/cm^3$	2.e14
TAU	Ambipolar recombination lifetime	sec	7.1e-6
THETA	Transverse field factor	1/V	0.02
VT	Threshold voltage	V	4.7
VTD	Gate-drain overlap depletion threshold	V	1.e-3
WB	Metallurgical base width	m	9.0e-5

* See .MODEL statement.

Equations

In the IGBT equations that follow the values are:

I_{mos}	MOSFET channel current
I_T	anode current
I_{css}	steady-state (bipolar) collector current
I_{bss}	Steady-state base current
I_{mult}	avalanche multiplication current
R_b	conductivity modulated base resistance
b	ambipolar mobility ratio
D_p	diffusion coefficient for holes
W	quasi-neutral base width
Q_{eb}	instantaneous excess carrier base charge
Q_b	background mobile carrier charge
n_i	intrinsic carrier concentration
M	avalanche multiplication factor
I_{gen}	(bipolar) collector-base thermally generated current
ε_{si}	dielectric permittivity of silicon
q	electron charge
W_{bcj}	base (bipolar) to collector depletion width

DC Current

The MOSFET channel current is defined for the three regions of operation as follows:

$$I_{MOS} = \begin{cases} 0 & \text{For } V_{gs} < \mathbf{VT} \\ \dfrac{\mathbf{KF} \cdot \mathbf{KP} \cdot \left((V_{gs} - \mathbf{VT}) \cdot V_{ds} - \dfrac{\mathbf{KF} \cdot V_{ds}^{\,2}}{2} \right)}{1 + \mathbf{THETA} \cdot (V_{gs} - \mathbf{VT})} & \text{For } V_{ds} \le (V_{gs} - \mathbf{VT})/\mathbf{KF} \\ \dfrac{\mathbf{KP} \cdot (V_{gs} - \mathbf{VT})^2}{2 \cdot (1 + \mathbf{THETA} \cdot (V_{gs} - \mathbf{VT}))} & \text{For } V_{ds} > (V_{gs} - \mathbf{VT})/\mathbf{KF} \end{cases}$$

The anode current is the current through the resistor R_b:

$$I_T = \frac{V_{Ce}}{R_b}$$

The steady-state collector current is given by:

$$I_{css} = \begin{cases} 0 & \text{For } V_{eb} \le 0 \\ \left(\dfrac{1}{1+b} \right) \cdot I_T + \left(\dfrac{b}{1+b} \right) \cdot \left(\dfrac{4 \cdot D_p}{W^2} \right) \cdot Q_{eb} & \text{For } V_{eb} > 0 \end{cases}$$

The steady-state base current is defined as follows:

$$I_{bss} = \begin{cases} 0 & \text{For } V_{eb} \le 0 \\ \dfrac{Q_{eb}}{\mathbf{TAU}} + \left(\dfrac{Q_{eb}^{\,2}}{Q_B} \right) \cdot \left(\dfrac{4 \cdot \mathbf{NB}^2}{n_i^{\,2}} \right) \cdot (\mathbf{JSNE} \cdot \mathbf{AREA}) & \text{For } V_{eb} > 0 \end{cases}$$

The avalanche multiplication current is given by:

$$I_{mult} = (M - 1) \cdot (I_{mos} + I_{css}) + M \cdot I_{gen}$$

Capacitance

C_{gs}:

$C_{gs} = \mathbf{CGS}$ $\qquad Q_{gs} = \mathbf{CGS} \cdot V_{gs}$

C_{ds}:

$$C_{ds} = \frac{(\mathbf{AREA} - \mathbf{AGD}) \cdot \varepsilon_{si}}{W_{dsj}} \qquad Q_{ds} = q \cdot (\mathbf{AREA} - \mathbf{AGD}) \cdot \mathbf{NB} \cdot W_{dsj}$$

where $W_{dsj} = \sqrt{\frac{2 \cdot \varepsilon_{si} \cdot (V_{ds} + 0.6)}{q \cdot \mathbf{NB}}}$

C_{dg}:

For $V_{ds} < V_{gs} - \mathbf{VTD}$,

$C_{dg} = \mathbf{COXD}$ $\qquad Q_{dg} = \mathbf{COXD} \cdot V_{dg}$

For $V_{ds} \geq V_{gs} - \mathbf{VTD}$,

$$C_{dg} = \frac{C_{dgj} \cdot \mathbf{COXD}}{C_{dgj} + \mathbf{COXD}}$$

$$Q_{dg} = \frac{q \cdot \mathbf{NB} \cdot \varepsilon_{si} \cdot \mathbf{AGD}^2}{\mathbf{COXD}} \left(\frac{\mathbf{COXD} \cdot W_{dgj}}{\varepsilon_{si} \cdot \mathbf{AGD}} - \log\left(1 + \frac{\mathbf{COXD} \cdot W_{dgj}}{\varepsilon_{si} \cdot \mathbf{AGD}}\right)\right) - \mathbf{COXD} \cdot \mathbf{VTD}$$

where

$$C_{dgj} = \frac{\mathbf{AGD} \cdot \varepsilon_{si}}{W_{dgj}} \qquad W_{dgj} = \sqrt{\frac{2 \cdot \varepsilon_{si} \cdot (V_{dg} + \mathbf{VTD})}{q \cdot \mathbf{NB}}}$$

C_{cer}:

$$C_{cer} = \frac{Q_{eb} \cdot C_{bcj}}{3 \cdot Q_B} \qquad C_{bcj} = \frac{\varepsilon_{si} \cdot \mathbf{AREA}}{W_{bcj}}$$

C_{mult}:

$$C_{mult} = (M-1) \cdot C_{cer} \qquad Q_{mult} = (M-1) \cdot Q_{cer}$$

C_{eb}:

$$C_{eb} = \frac{dQ_{eb}}{dV_{eb}}$$

References

For more information on the IGBT model, refer to:

[1] G.T. Oziemkiewicz, “Implementation and Development of the NIST IGBT Model in a SPICE-based Commercial Circuit Simulator,” Engineer’s Thesis, University of Florida, December 1995.

[2] A.R.Hefner, Jr., “INSTANT - IGBT Network Simulation and Transient Analysis Tool,” National Institute of Standards and Technology Special Publication SP 400-88, June 1992.

[3] A.R.Hefner, Jr., “An Investigation of the Drive Circuit Requirements for the Power Insulated Gate Bipolar Transistor (IGBT),” *IEEE Transactions on Power Electronics* , Vol. 6, No. 2, April 1991, pp. 208-219.

[4] A.R.Hefner, Jr., “Modeling Buffer Layer IGBT’s for Circuit Simulation,” *IEEE Transactions on Power Electronics* , Vol. 10, No. 2, March 1995, pp. 111-123

Digital Devices

3

Overview

This chapter describes the digital devices that are supported by PSpice A/D. These devices include primitives, such as gates and flip-flops, stimulus devices which provide inputs to the simulation, and interface devices that provide A-D and D-A interfaces.

The digital primitives are used to build more complex device models, such as those found in the digital model libraries included with the simulator.

Digital Devices

The digital devices are summarized below.

Table 3-1 *Digital Device Summary*

Device Class	Type	Description
Primitives	U	Low-level digital devices (e.g., gates and flip-flops)
Stimuli	U	Digital stimulus generators
		File-based stimulus
Interface	N	Digital input device
	O	Digital output device

Primitives are primarily used in subcircuits to model complete devices. Stimulus devices are used in the circuit to provide input for other digital devices during the simulation. Interface devices are mainly used inside subcircuits which model analog/digital and digital/analog interfaces.

Note *The digital devices are part of the digital simulation feature of PSpice A/D. For more information on digital simulation and creating models, refer to your PSpice user's guide.*

Digital Primitives

Digital primitives are low-level devices whose main use, often in combinations with each other, is to model off-the-shelf parts for the model library. Digital primitives should not be confused with the subcircuits in the libraries which use them. For instance, the 74LS00 subcircuit in “74ls.lib” uses a NAND digital primitive to model the 74LS00 part, but it also includes timing and interface information that makes the model adapted for use in a circuit simulation. For more information, refer to your PSpice user’s guide.

This section provides a reference for each of the digital primitives supported by the simulator. The purpose of this section is to assist the designer in the creation of digital parts which are not in the model library. The references are grouped as shown in the Digital Primitives Summary, Table 3-2.

Table 3-2 *Digital Primitives Summary*

Primitive Class	Type	Description	Page Number
Standard Gates	BUF	Buffer	3-12
	INV	Inverter	
	AND	AND gate	
	NAND	NAND gate	
	OR	OR gate	
	NOR	NOR gate	
	XOR	Exclusive OR gate	
	NXOR	Exclusive NOR gate	
	BUFA	Buffer array	
	INVA	Inverter array	
	ANDA	AND gate array	
	NANDA	NAND gate array	
	ORA	OR gate array	
	NORA	NOR gate array	
	XORA	Exclusive OR gate array	
	NXORA	Exclusive NOR gate array	
	AO	AND-OR compound gate	
	OA	OR-AND compound gate	
	AOI	AND-NOR compound gate	
	OAI	OR-NAND compound gate	

Table 3-2 *Digital Primitives Summary (continued)*

Primitive Class	Type	Description	Page Number
Tristate Gates	BUF3	Buffer	3-15
	INV3	Inverter	
	AND3	AND gate	
	NAND3	NAND gate	
	OR3	OR gate	
	NOR3	NOR gate	
	XOR3	Exclusive OR gate	
	NXOR3	Exclusive NOR gate	
	BUF3A	Buffer array	
	INV3A	Inverter array	
	AND3A	AND gate array	
	NAND3A	NAND gate array	
	OR3A	OR gate array	
	NOR3A	NOR gate array	
	XOR3A	Exclusive OR gate array	
	NXOR3A	Exclusive NOR gate array	
Bidirectional Transfer Gates	NBTG	N-channel transfer gate	3-18
	PBTG	P-channel transfer gate	
Flip-Flops and Latches	JKFF	J-K, negative-edge triggered	3-19
	DFF	D-type, positive-edge triggered	
	SRFF	S-R gated latch	
	DLTCH	D gated latch	
Pullup and Pulldown Resistors	PULLUP	Pullup resistor array	3-32
	PULLDN	Pulldown resistor array	
Delay Lines	DLYLINE	Delay line	3-33

Table 3-2 *Digital Primitives Summary (continued)*

Primitive Class	Type	Description	Page Number
Programmable Logic Arrays	PLAND	AND array	3-34
	PLOR	OR array	
	PLXOR	Exclusive OR array	
	PLNAND	NAND array	
	PLNOR	NOR array	
	PLNXOR	Exclusive NOR array	
	PLANDC	AND array, true and complement	
	PLORC	OR array, true and complement	
	PLXORC	Exclusive OR array, true and complement	
	PLNANDC	NAND array, true and complement	
	PLNORC	NOR array, true and complement	
	PLNXORC	Exclusive NOR array, true and complement	
Memory	ROM	Read-only memory	3-39
	RAM	Random access read-write memory	3-43
Multi-Bit A/D and D/A Converters	ADC	Multi-bit A/D converter	3-47
	DAC	Multi-bit D/A converter	
Behavioral	LOGICEXP	Logic expression	3-52
	PINDLY	Pin-to-pin delay	
	CONSTRAINT	Constraint checking	

The format for specifying a digital primitive follows the general format described in the next section. Primitive-specific formats are also described which includes parameters and nodes that are specific to the primitive type.

Also listed is the specific timing model format for each primitive, along with the appropriate timing model parameters.

For example, the 74393 part provided in the model library is defined as a subcircuit composed of “U” devices as shown below.

```
subckt 74393   A CLR QA QB QC QD
+  optional: DPWR=$G_DPWR DGND=$G_DGND
+  params: MNTYMXDLY=0 IO_LEVEL=0
UINV inv DPWR DGND
+  CLR   CLRBAR
+  D0_GATE IO_STD IO_LEVEL={IO_LEVEL}
U1 jkff(1) DPWR DGND
+  $D_HI CLRBAR A   $D_HI $D_HI   QA_BUF $D_NC
+  D_393_1 IO_STD MNTYMXDLY={MNTYMXDLY}=
+  IO_LEVEL={IO_LEVEL}
U2 jkff(1) DPWR DGND
+  $D_HI CLRBAR QA_BUF   $D_HI $D_HI   QB_BUF $D_NC
+  D_393_2 IO_STD MNTYMXDLY={MNTYMXDLY}
U3 jkff(1) DPWR DGND
+  $D_HI CLRBAR QB_BUF   $D_HI $D_HI   QC_BUF $D_NC
+  D_393_2 IO_STD MNTYMXDLY={MNTYMXDLY}
U4 jkff(1) DPWR DGND
+  $D_HI CLRBAR QC_BUF   $D_HI $D_HI   QD_BUF $D_NC
+  D_393_3 IO_STD MNTYMXDLY={MNTYMXDLY}
UBUFF bufa(4) DPWR DGND
+  QA_BUF QB_BUF QC_BUF QD_BUF   QA QB QC QD
+  D_393_4 IO_STD MNTYMXDLY={MNTYMXDLY}IO_LEVEL={IO_LEVEL}
.ends
```

When adding digital parts to the Symbol Library, corresponding digital device models can be created by connecting U devices in a subcircuit definition similar to the one shown above. It is recommend that these be saved in a custom model file. The model files can then be configured into the model library or specified for use in a given schematic.

General Digital Primitive Format

The format of digital primitives is similar to that of analog devices. One difference is that most digital primitives use two models instead of one. One of the models is the timing model, which specifies propagation delays and timing constraints, such as setup and hold times. The other model is the I/O model, which specifies information specific to the device's input/output characteristics. The reason for having two models is that, while timing information is specific to a device, the input/output characteristics apply to a whole device family. Thus, many devices in the same family reference the same I/O model, but each device has its own timing model. If wanted, the timing models can be selected among primitives of the same class.

The general digital primitive format is shown below. Each statement can span one or more lines by using the '+' continuation character in the first column position. Comments can be added to each line by first typing the ';' and then adding the comments. For specific information on each primitive type, see the sections that follow.

General Form

U<*name*> <*primitive type*> [(<*parameter value*>*)]
+ <*digital power node*> <*digital ground node*>
+ <*node*>*
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY=<*delay select value*>]
+ [IO_LEVEL=<*interface subckt select value*>]

Example

```
U1 NAND(2) $G_DPWR $G_DGND 1 2 10 D0_GATE IO_DFT
U2 JKFF(1) $G_DPWR $G_DGND 3 5 200 3 3 10 2 D_293ASTD IO_STD
U3 INV $G_DPWR $G_DGND IN OUT D_INV IO_INV MNTYMXDLY=3
IO_LEVEL=2
```

<*primitive type*> [(<*parameter value*>*)]

The type of digital device, such as NAND, JKFF, or INV. It is followed by zero or more parameters specific to the primitive type, such as number of inputs. The number and meaning of the parameters depends on the primitive type. See the sections that follow for a complete description of each primitive type and its parameters.

<digital power node> <digital ground node>

These nodes are used by the interface subcircuits which connect analog nodes to digital nodes or vice versa. Refer to your PSpice user's guide for more information.

*<node>**

One or more input and output nodes. The number of nodes depends on the primitive type and its parameters. Analog devices, digital devices, or both can be connected to a node. If a node has both analog and digital connections, then the simulator automatically inserts an interface subcircuit to translate between logic levels and voltages. Refer to your PSpice user's guide for more information.

<timing model name>

The name of a timing model, which describes the device's timing characteristics, such as propagation delay and setup and hold times. Each timing parameter has a minimum, typical, or maximum value which can be selected using the optional MNTYMXDLY device parameter (described below) or the DIGMNTYMX option (see .OPTIONS (Analysis Options) command on page **1-35**). The type of the timing model and its parameters are specific to each primitive type and are discussed in the following sections. (Note that the PULLUP, PULLDN, and PINDLY primitives do not have timing models.)

<I/O model name>

The name of an I/O model, which describes the device's loading and driving characteristics. I/O models also contain the names of up to four DtoA and AtoD interface subcircuits, which are automatically called by the simulator to handle interface nodes. Refer to your PSpice user's guide for a more detailed description of I/O models.

MNTYMXDLY

An optional device parameter which selects either the minimum, typical, or maximum delay values from the device's timing model. A fourth option operates the primitive in Digital Worst-Case (min/max) mode. If not specified, MNTYMXDLY defaults to 0. Valid values are:

0 = Current value of .OPTIONS DIGMNTYMX (default=2)
1 = Minimum
2 = Typical
3 = Maximum
4 = Worst-case (min/max) timing

IO_LEVEL

An optional device parameter which selects one of the four AtoD or DtoA interface subcircuits from the device's I/O model. The simulator calls the selected subcircuit automatically in the event a node connecting to the primitive also connects to an analog device. If not specified, IO_LEVEL defaults to 0. Valid values are:

0 = the current value of .OPTIONS DIGIOLVL (default=1)
1 = AtoD1/DtoA1
2 = AtoD2/DtoA2
3 = AtoD3/DtoA3
4 = AtoD4/DtoA4

Refer to your PSpice user's guide for more information.

Timing Model Format

`.MODEL <model name> <model type> ( <model parameters>* )`

The <*model type*> is specific to the primitive type. See the specific primitive for the correct <*model type*> and associated <*model parameters*>. General timing model issues are discussed in the next section.

Model Form

`.MODEL <model name> UIO ( <model parameters>* )`

See **Table 3-22 on page 3-91**, for a list of the UIO model parameters.

Timing Models

With the exception of the PULLUP, PULLDN, and PINDLY devices, all digital primitives have a timing model which provides timing parameters to the simulator. Within a timing model, there can be one or more types of parameters: propagation delays (TP), setup times (TSU), hold times (TH), pulse widths (TW), and switching times (TSW). Each parameter is further divided into three values: minimum (MN), typical (TY), and maximum (MX). For example, the typical low-to-high propagation delay on a gate is specified as TPLHTY. The minimum data-to-clock setup time on a flip-flop is specified as TSUDCLKMN.

One or more parameters can be missing from the timing model definition. Data books do not always provide all three (minimum, typical, and maximum) timing specifications. The way the simulator handles missing parameters depends on the type of parameter.

Treatment of Unspecified Propagation Delays

Note that this discussion applies *only* to propagation delay parameters (TP). All other timing parameters, such as setup/hold times and pulse widths are handled differently, and are discussed in the following section.

Often, only the typical and maximum delays are specified in data books. If, in this case, the simulator were to assume that the unspecified minimum delay just defaults to zero, the logic in certain circuits could break down. For this reason, the simulator provides two configurable options, DIGMNTYSCALE and DIGTYMXSCALE (set using the .OPTIONS command), which are used to extrapolate unspecified propagation delays in the timing models.

The first option, DIGMNTYSCALE, is used to compute the minimum delay when a typical delay is known, using the formula

$$\text{TPxxMN} = \text{DIGMNTYSCALE} \cdot \text{TPxxTY}$$

DIGMNTYSCALE defaults to the value 0.4, or 40% of the typical delay. Its value must be between 0.0 and 1.0.

The second option, DIGTYMXSCALE, is used in a similar manner to compute the maximum delay from a typical delay, using the formula

$$TPxxMX = DIGTYMXSCALE \cdot TPxxTY$$

DIGTYMXSCALE defaults to the value 1.6. Its value must be greater than 1.0.

When a typical delay is unspecified, its value is derived from the minimum and/or maximum delays, in one of the following ways. If both the minimum and maximum delays are known, the typical delay is the average of these two values. If only the minimum delay is known, the typical delay is derived using the value of the DIGMNTYSCALE option. Likewise, if only the maximum delay is specified, the typical delay is derived using DIGTYMXSCALE. Obviously, if no values are specified, all three delays default to zero.

Treatment of Unspecified Timing Constraints

The remaining timing constraint parameters are handled differently than the propagation delays. Often, data books state pulse widths, setup times, and hold times as a minimum value. These parameters do not lend themselves to the extrapolation method used for propagation delays.

Instead, when one or more timing constraints are omitted, the simulator uses the following steps to fill in the missing values:

- If the minimum value is omitted, it defaults to zero.
- If the maximum value is omitted, it takes on the typical value if one was specified, otherwise it takes on the minimum value.
- If the typical value is omitted, it is computed as the average of the minimum and maximum values.

Gates

Logic gates come in two types: standard and tristate. Standard gates always have their outputs enabled, whereas tristate gates have an enable control. When the enable control is 0, the output's strength is Z and its level is X.

Logic gates also come in two forms: simple gates and gate arrays. Simple gates have one or more inputs and only one output. Gate arrays contain one or more simple gates in one component. Gate arrays allow one to work directly using parts that have several gates in one package.

The usual Boolean equations apply to these gates having the addition of the X level. The rule for X is: if an input is X, and if changing that input between one and zero would cause the output to change, then the output is also X. In other words, X is only propagated to the output when necessary. For example: 1 AND X = X; 0 AND X = 0; 0 OR X = X; 1 OR X = 1.

Standard Gates

Device Format

U<*name*> <*gate type*> [(<*parameter value*>*)]
+ <*digital power node*> <*digital ground node*>
+ <*input node*>* <*output node*>*
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY=<*delay select value*>]
+ [IO_LEVEL=<*interface subckt select value*>]

The standard gate types and their parameters are listed in Table 3-3.

Example

```
U5  AND(2) $G_DPWR $G_DGND IN0 IN1 OUT; two-input AND gate
+ T_AND2 IO_STD
U2  INV  $G_DPWR $G_DGND 3 5; simple INVerter
+ T_INV IO_STD
U13 NANDA(2,4) $G_DPWR $G_DGND; four two-input NAND gates
+ INA0 INA1 INB0 INB1 INC0 INC1
+ IND0 IND1 OUTA OUTB OUTC OUTD
+ T_NANDA IO_STD

.MODEL T_AND2 UGATE; AND2 Timing Model-see below
+ TPLHMN=15ns TPLHTY=20ns TPLHMX=25ns
+ TPHLMN=10ns TPHLTY=15ns TPHLMX=20ns
+ )
```

Table 3-3 *Standard Gate Types*

Type	Parameters	Nodes	Description
AND	(<*no. of inputs*>)	in*, out	AND gate
ANDA	(<*no. of inputs*>,<*no. of gates*>)	in*, out*	AND gate array
AO	(<*no. of inputs*>,<*no. of gates*>)	in*, out	AND-OR compound gate
AOI	(<*no. of inputs*>,<*no. of gates*>)	in*, out	AND-NOR compound gate
BUF		in, out	Buffer
BUFA	(<*no. of gates*>)	in*, out*	Buffer array
INV		in, out	Inverter
INVA	(<*no. of gates*>)	in*, out*	Inverter array
NAND	(<*no. of inputs*>)	in*, out	NAND gate
NANDA	(<*no. of inputs*>,<*no. of gates*>)	in*, out*	NAND gate array
NOR	(<*no. of inputs*>)	in*, out	NOR gate
NORA	(<*no. of inputs*>,<*no. of gates*>)	in*, out*	NOR gate array
NXOR		in1, in2, out	Exclusive NOR gate
NXORA	(<*no. of gates*>)	in*, out*	Exclusive NOR gate array
OA	(<*no. of inputs*>,<*no. of gates*>)	in*, out	OR-AND compound gate
OAI	(<*no. of inputs*>,<*no. of gates*>)	in*, out	OR-NAND compound gate
OR	(<*no. of inputs*>)	in*, out	OR gate
ORA	(<*no. of inputs*>,<*no. of gates*>)	in*, out*	OR gate array
XOR		in1, in2, out	Exclusive OR gate
XORA	(<*no. of gates*>)	in*, out*	Exclusive OR gate array

<no. of inputs><no. of gates>

The *<no. of inputs>* is the number of inputs per gate and *<no. of gates>* is the number of gates. “in*” and “out*” mean one or more nodes, whereas “in” and “out” refer to only one node.

In gate arrays the order of the nodes is: all inputs for the first gate, all inputs for the second gate, ..., output for the first gate, output for the second gate, ... In other words, all of the input nodes come first, then all of the output nodes. The total number of input nodes is *<no. of inputs>*·*<no. of gates>*; the number of output nodes is *<no. of gates>*.

A compound gate is a set of *<no. of gates>* first-level gates which each have *<no. of inputs>* inputs. Their outputs are connected to a single second-level gate. For example, the AO component has *<no. of gates>* AND gates whose outputs go into one OR gate. The OR gate’s output is the AO device’s output. The order of the nodes is: all inputs for the first, first-level gate; all inputs for the second, first-level gate; ...; the output of the second-level gate. In other words, all of the input nodes followed by the one output node.

Timing Model Format `MODEL <timing model name> UGATE [model parameters]`

Table 3-4 *Standard Gate Timing Model Parameters*

Model Parameters*	Description	Units	Default
TPLHMN	delay: low to high, min	sec	0
TPLHTY	delay: low to high, typ	sec	0
TPLHMX	delay: low to high, max	sec	0
TPHLMN	delay: high to low, min	sec	0
TPHLTY	delay: high to low, typ	sec	0
TPHLMX	delay: high to low, max	sec	0

* See .MODEL statement.

Tristate Gates

Device Format

U<*name*> <*tristate gate type*> [(<*parameter value*>*)]
+ <*digital power node*> <*digital ground node*>
+ <*input node*>* <*enable node*> <*output node*>*
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY=<*delay select value*>]
+ [IO_LEVEL=<*interface subckt select value*>]

Example

```
U5  AND3(2) $G_DPWR $G_DGND IN0 IN1 ENABLE OUT two-input AND
+ T_TRIAND2 IO_STD
U2  INV3 $G_DPWR $G_DGND 3 100 5; INVerter
+ T_TRIINV IO_STD
U13 NAND3A(2,4) $G_DPWR $G_DGND; four two-input NAND
+ INA0 INA1 INB0 INB1 INC0 INC1 IND0 IND1
+ ENABLE OUTA OUTB OUTC OUTD
+ T_TRINAND IO_STD

.MODEL T_TRIAND2 UTGATE     ; TRI-AND2 Timing Model see below
+ TPLHMN=15ns TPLHTY=20ns TPLHMX=25ns ...
+ TPZHMN=10ns TPZHTY=15ns TPZHMX=20ns
+ )
```

<*no. of inputs*> The number of inputs per gate.

<*no. of gates*> The number of gates in model.

In gate arrays the order of the nodes is: all inputs for the first gate, all inputs for the second gate, ..., enable, output for the first gate, output for the second gate, ... In other words, all of the input nodes come first, then the enable, then all of the output nodes. The total number of input nodes is <*no. of inputs*>·<*no. of gates*>+1; the number of output nodes is <*no. of gates*>. If a tristate gate is connected to a net which has at least one device input using an INLD I/O model, or a device output using an OUTLD I/O model where both parameters are greater than zero, then that net is simulated as a charge storage net.

Table 3-5 *Tristate Gate Types*

Type	Parameters	Nodes*	Description
AND3	(<*no. of inputs*>)	in*, en, out	AND gate
AND3A	(<*no. of inputs*>,<*no. of gates*>)	in*, en, out*	AND gate array
BUF3		in, en, out	Buffer
BUF3A	(<*no. of gates*>)	in*, en, out*	Buffer array
INV3		in, en, out	Inverter
INV3A	(<*no. of gates*>)	in*, en, out*	Inverter array
NAND3	(<*no. of inputs*>)	in*, en, out	NAND gate
NAND3A	(<*no. of inputs*>,<*no. of gates*>)	in*, en, out*	NAND gate array
NOR3	(<*no. of inputs*>)	in*, en, out	NOR gate
NOR3A	(<*no. of inputs*>,<*no. of gates*>)	in*, en, out*	NOR gate array
NXOR3		in1, in2, en, out	Exclusive NOR gate
NXOR3A	(<*no. of gates*>)	in*, en, out*	Excl. NOR gate array
OR3	(<*no. of inputs*>)	in*, en, out	OR gate
OR3A	(<*no. of inputs*>,<*no. of gates*>)	in*, en, out*	OR gate array
XOR3		in1, in2, en, out	Exclusive OR gate
XOR3A	(<*no. of gates*>)	in*, en, out*	Excl. OR gate array

* "in*" and "out*": One or more nodes present.
"in" and "out": Refers to only one node.
"en": Refers to the output enable node.

Timing Model Format.MODEL <*timing model name*> UTGATE [*model parameters*]

Table 3-6 *Tristate Gate Timing Model Parameters*

Model Parameters*	Description	Units	Default
TPLHMN	Delay: low to high, min	sec	0
TPLHTY	Delay: low to high, typ	sec	0
TPLHMX	Delay: low to high, max	sec	0
TPHLMN	Delay: high to low, min	sec	0
TPHLTY	Delay: high to low, typ	sec	0
TPHLMX	Delay: high to low, max	sec	0
TPHZMN	Delay: high to Z, min	sec	0
TPHZTY	Delay: high to Z, typ	sec	0
TPHZMX	Delay: high to Z, max	sec	0
TPLZMN	Delay: low to Z, min	sec	0
TPLZTY	Delay: low to Z, typ	sec	0
TPLZMX	Delay: low to Z, max	sec	0
TPZLMN	Delay: Z to low, min	sec	0
TPZLTY	Delay: Z to low, typ	sec	0
TPZLMX	Delay: Z to low, max	sec	0
TPZHMN	Delay: Z to high, min	sec	0
TPZHTY	Delay: Z to high, typ	sec	0
TPZHMX	Delay: Z to high, max	sec	0

* See .MODEL statement.

Bidirectional Transfer Gates

The bidirectional transfer gate is a passive device which connects or disconnects two nodes. The state of the gate input controls whether the gate connects the two digital nets. The device type NBTG connects the nodes if the gate is one, and disconnects the nodes if the gate is zero. Device type PBTG connects the nodes if the gate is zero and disconnects the nodes if the gate is one. The bidirectional transfer gates have no parameters.

The I/O Model DRVH and DRVL parameters are used as a ceiling on the strength of a one or zero which is passed through a bidirectional transfer gate. If a bidirectional transfer gate is connected to a net which has at least one device input using an INLD I/O model parameter greater than zero, or a device output using an OUTLD I/O model parameter greater than zero, then that net is simulated as a charge storage net.

Device Format

U<*name*> NBTG
+ <*digital power node*> <*digital ground node*>
+ <*gate node*> <*channel node 1*> <*channel node 2*>
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY = <*delay select value*>]
+ [IO_LEVEL = <*interface subckt select value*>]

U<*name*> PBTG
+ <*digital power node*> <*digital ground node*>
+ <*gate node*> <*channel node 1*> <*channel node 2*>
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY = <*delay select value*>]
+ [IO_LEVEL = <*interface subckt select value*>]

Example

```
U4 NBTG $G_DPWR $G_DGND GATE SD1 SD2
+ BTG1 IO_BTG
.MODEL BTG1 UBTG
```

Model Form

.MODEL <*timing model name*> UBTG

Flip-Flops and Latches

The simulator supports both edge-triggered and gated flip-flops. Edge-triggered flip-flops change state when the clock changes: on the falling edge for JKFFs, on the rising edge for DFFs. Gated flip-flops are often referred to as latches. The state of gated flip-flops follows the input as long as the clock (gate) is high. The state is “frozen” when the clock (gate) falls. Multiple flip-flops can be specified in each device. This allows direct modeling of parts which contain more than one flip-flop in a package.

Initialization

By default, at the beginning of each simulation, all flip-flops and latches are initialized to the unknown state (that is, they output an X). Each device remains in the unknown state until explicitly set or cleared by an active-low pulse on either the preset or clear pins, or until a known state is clocked in.

The X start-up state can be overridden by setting .OPTIONS DIGINITSTATE to either zero or one. If set to zero, all flip-flops and latches in the circuit are cleared. Likewise, if set to one, all such devices are preset. Any other values produce the default (X) start-up state. The DIGINITSTATE option is useful in situations where the initial state of the flip-flop is unimportant to the function of the circuit, such as a toggle flip-flop in a frequency divider.

It is important to note that if the initial state is set to zero or one, the device still outputs an X at the beginning of the simulation if the inputs would normally produce an X on the output. For example, if the initial state is set to one, but the clock is an X at time zero, Q and QBar both go to X when the simulation begins.

X-Level Handling

The truth-table for each type of flip-flop and latch is given in the sections that follow. However, how the flip-flops treat X levels on the inputs is not depicted in the truth tables because it can depend on the state of the device.

The rule is as follows: if an input is X, and if changing that input between one and zero would cause the output to change, then the output is set to X. In other words, X is only propagated to the output when necessary. For example: if Q = 0 and PresetBar = X, then Q → X; but if Q = 1 and PresetBar = X, then Q → 1.

Timing Violations

The flip-flop and latch primitives have model parameters which specify timing constraints such as setup/hold times and minimum pulse-widths. If these model parameter values are greater than zero, the simulator compares measured times on the inputs against the specified value. See **Table 3-4 on page 3-14** and **Table 3-6 on page 3-17** .

The simulator reports flip-flop timing violations as digital simulation warning messages in the ".out" file. These messages can also be viewed using the Windows version of Probe.

Edge-Triggered Flip-Flops

The simulator supports four types of edge-triggered flip-flops:

- D-type flip-flop (DFF), which is positive-edge triggered
- J-K flip-flop (JKFF), which is negative-edge triggered
- Dual-edge D flip-flop (DFFDE), which is selectively positive and/or negative edge triggered
- Dual-edge J-K flip-flop (JKFFDE), which is selectively positive and/or negative edge triggered

Device Format

U<*name*> DFF (<*no. of flip-flops*>)
+ <*digital power node*> <*digital ground node*>
+ <*presetbar node*> <*clearbar node*> <*clock node*>
+ <*d node 1*> ... <*d node n*>
+ <*q output 1*> ... <*q output n*>
+ <*qbar output 1*> ... <*qbar output n*>
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY=<*delay select value*>]
+ [IO_LEVEL=<*interface subckt select value*>]

U<*name*> JKFF (<*no. of flip-flops*>)
+ <*digital power node*> <*digital ground node*>
+ <*presetbar node*> <*clearbar node*> <*clockbar node*>
+ <*j node 1*> ... <*j node n*>
+ <*k node 1*> ... <*k node n*>
+ <*q output 1*> ... <*q output n*>
+ <*qbar output 1*> ... <*qbar output n*>
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY=<*delay select value*>]
+ [IO_LEVEL=<*interface subckt select value*>]

U<*name*> DFFDE(<*no. of flip-flops*>)
+ <*digital power node*> <*digital ground node*>
+ <*presetbar node*> <*clrbar node*> <*clock node*>
+ <*positive-edge enable node*> <*negative-edge enable node*>
+ <*d node 1*> ... <*d node n*>
+ <*q output 1*> ... <*q output n*>
+ <*qbar output 1*> ... <*qbar output n*>
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY = <*delay select value*>]
+ [IO_LEVEL = <*interface subckt select value*>]

U<*name*> JKFFDE(<*no. of flip-flops*>)
+ <*digital power node*> <*digital ground node*>
+ <*presetbar node*> <*clrbar node*> <*clock node*>
+ <*positive-edge enable node*> <*negative-edge enable node*>
+ <*j node 1*> ... <*j node n*>
+ <*k node 1*> ... <*k node n*>
+ <*q output 1*> ... <*q output n*>
+ <*qbar output 1*> ... <*qbar output n*>
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY = <*delay select value*>]
+ [IO_LEVEL = <*interface subckt select value*>]

Use <*no. of flip-flops*> to specify the number of flip-flops in the device. The three nodes, <*presetbar node*>, <*clearbar node*> and <*clock(bar) node*>, are common to all flip-flops in the device.

The <*positive-edge enable node*> and <*negative-edge enable node*> are common to all flip-flops in the dual-edge flip-flops.

Example

```
U5 JKFF(1) $G_DPWR $G_DGND PREBAR CLRBAR CLKBAR
* one JK flip-flop
+ J K Q QBAR
+ T_JKFF IO_STD
U2 DFF(2)  $G_DPWR $G_DGND PREBAR CLRBAR CLK
* two DFF flip-flops
+ D0 D1 Q0 Q1 QBAR0 QBAR1
+ T_DFF IO_STD

.MODEL T_JKFF UEFF(...) ; JK Timing Model
   ; - see below
```

Timing Model Format

.MODEL <*timing model name*> UEFF [*model parameters*]

Table 3-7 *Edge-Triggered Flip-Flop Timing Model Parameters*

Model Parameters*	Description	Units	Default
THDCLKMN	Hold: j/k/d after clk/clkb edge, min	sec	0
THDCLKTY	Hold: j/k/d after clk/clkb edge, typ	sec	0
THDCLKMX	Hold: j/k/d after clk/clkb edge, max	sec	0
TPCLKQLHMN	Delay: clk/clkb edge to q/qb low to hi, min	sec	0
TPCLKQLHTY	Delay: clk/clkb edge to q/qb low to hi, typ	sec	0
TPCLKQLHMX	Delay: clk/clkb edge to q/qb low to hi, max	sec	0
TPCLKQHLMN	Delay: clk/clkb edge to q/qb hi to low, min	sec	0
TPCLKQHLTY	Delay: clk/clkb edge to q/qb hi to low, typ	sec	0
TPCLKQHLMX	Delay: clk/clkb edge to q/qb hi to low, max	sec	0
TPPCQLHMN	Delay: preb/clrb to q/qb low to hi, min	sec	0
TPPCQLHTY	Delay: preb/clrb to q/qb low to hi, typ	sec	0
TPPCQLHMX	Delay: preb/clrb to q/qb low to hi, max	sec	0
TPPCQHLMN	Delay: preb/clrb to q/qb hi to low, min	sec	0
TPPCQHLTY	Delay: preb/clrb to q/qb hi to low, typ	sec	0
TPPCQHLMX	Delay: preb/clrb to q/qb hi to low, max	sec	0
TSUDCLKMN	Setup: j/k/d to clk/clkb edge, min	sec	0
TSUDCLKTY	Setup: j/k/d to clk/clkb edge, typ	sec	0
TSUDCLKMX	Setup: j/k/d to clk/clkb edge, max	sec	0
TSUPCCLKHMN	Setup: preb/clrb hi to clk/clkb edge, min	sec	0
TSUPCCLKHTY	Setup: preb/clrb hi to clk/clkb edge, typ	sec	0
TSUPCCLKHMX	Setup: preb/clrb hi to clk/clkb edge, max	sec	0
TWPCLMN	Min preb/clrb width low, min	sec	0
TWPCLTY	Min preb/clrb width low, typ	sec	0
TWPCLMX	Min preb/clrb width low, max	sec	0
TWCLKLMN	Min clk/clkb width low, min	sec	0
TWCLKLTY	Min clk/clkb width low, typ	sec	0
TWCLKLMX	Min clk/clkb width low, max	sec	0
TWCLKHMN	Min clk/clkb width hi, min	sec	0

Table 3-7 *Edge-Triggered Flip-Flop Timing Model Parameters*

Model Parameters*	Description	Units	Default
TWCLKHTY	Min clk/clkb width hi, typ	sec	0
TWCLKHMX	Min clk/clkb width hi, max	sec	0
TSUCECLKMN	Setup: clock enable to clk edge, min	sec	0
TSUCECLKTY	Setup: clock enable to clk edge, typ	sec	0
TSUCECLKMX	Setup: clock enable to clk edge, max	sec	0
THCECLKMN	Hold: clock enable after clk edge, min	sec	0
THCECLKTY	Hold: clock enable after clk edge, typ	sec	0
THCECLKMX	Hold: clock enable after clk edge, max	sec	0

* See .MODEL statement.

Edge-Triggered Flip-Flop Truth Tables DFF and JKFF

The function tables for the DFF and JKFF primitives are given in Table 3-8 and Table 3-9.

Table 3-8 *D-Type Flip-Flop (DFF) Truth Table*

Inputs				Outputs	
D	CLK	$\overline{PRE}$	$\overline{CLR}$	Q	$\overline{Q}$
X	X	1	0	0	1
X	X	0	1	1	0
X	X	0	0	1*	1*
X	0	1	1	Q′	$\overline{Q}$′
X	1	1	1	Q′	$\overline{Q}$′
0	↑	1	1	0	1
1	↑	1	1	1	0

* Shows an unstable condition.

Table 3-9 *J-K Flip-Flop (JKFF) Truth Table*

Inputs					Outputs	
J	K	CLK	$\overline{PRE}$	$\overline{CLR}$	Q	Q
X	X	X	1	0	0	1
X	X	X	0	1	1	0
X	X	X	0	0	1*	1*
X	X	0	1	1	Q′	$\overline{Q}$′
X	X	1	1	1	Q′	$\overline{Q}$′
0	0	Ø	1	1	Q′	$\overline{Q}$′
0	1	Ø	1	1	0	1
1	0	↓	1	1	1	0
1	1	↓	1	1	$\overline{Q}$′	Q′

* Shows an unstable condition.

Edge-Triggered Flip-Flop Truth Tables DFFDE and JKFFDE

The function tables for the DFF and JKFF primitives are given in Table 3-10 and Table 3-11.

Table 3-10 *Dual-Edge D Flip-Flop (DFFDE) Truth Table*

Inputs						Outputs	
D	CLK	PENA	NENA	$\overline{PRE}$	$\overline{CLR}$	Q	Q
X	X	X	X	1	0	0	1
X	X	X	X	0	1	1	0
X	X	X	X	0	0	1*	1*
X	0	X	X	1	1	Q'	$\overline{Q'}$
X	1	X	X	1	1	Q'	$\overline{Q'}$
X	X	0	0	1	1	Q'	$\overline{Q'}$
0	↑	1	X	1	1	0	1
1	↑	1	X	1	1	1	0
0	↓	X	1	1	1	0	1
1	↓	X	1	1	1	1	0

* Shows an unstable condition.

Table 3-11 *Dual-edge J-K Flip-Flop (JKFFDE) Truth Table*

Inputs							Outputs	
J	K	CLK	PENA	NENA	$\overline{PRE}$	$\overline{CLR}$	Q	$\overline{Q}$
X	X	X	X	X	1	0	0	1
X	X	X	X	X	0	1	1	0
X	X	X	X	X	0	0	1*	1*
X	X	0	X	X	1	1	Q'	$\overline{Q}$'
X	X	1	X	X	1	1	Q'	$\overline{Q}$'
X	X	X	0	0	1	1	Q'	$\overline{Q}$'
0	0	↑	1	X	1	1	Q'	$\overline{Q}$'
0	1	↑	1	X	1	1	0	1
1	0	↑	1	X	1	1	1	0
1	1	↑	1	X	1	1	$\overline{Q}$'	Q'
0	0	↓	X	1	1	1	Q'	$\overline{Q}$'
0	1	↓	X	1	1	1	0	1
1	0	↓	X	1	1	1	1	0
1	1	↓	X	1	1	1	$\overline{Q}$'	Q'

* Shows an unstable condition.

Gated Latch

The simulator supports two types of gated latches: the S-R flip-flop (SRFF) and the D-type latch (DLTCH).

Device Format

U<*name*> SRFF (<*no. of flip-flops*>)
+ <*digital power node*> <*digital ground node*>
+ <*presetbar node*> <*clearbar node*> <*gate node*>
+ <*s node 1*> ... <*s node n*>
+ <*r node 1*> ... <*r node n*>
+ <*q output 1*> ... <*q output n*>
+ <*qbar output 1*> ... <*qbar output n*>
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY=<*delay select value*>]
+ [IO_LEVEL=<*interface subckt select value*>]

U<*name*> DLTCH (<*no. of latches*>)
+ <*digital power node*> <*digital ground node*>
+ <*presetbar node*> <*clearbar node*> <*gate node*>
+ <*d node 1*> ... <*d node n*>
+ <*q output 1*> ... <*q output n*>
+ <*qbar output 1*> ... <*qbar output n*>
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY=<*delay select value*>]
+ [IO_LEVEL=<*interface subckt select value*>]

Use <*no. of flip-flops*> to specify the number of flip-flops in the device. The three nodes, <*presetbar node*>, <*clearbar node*>, and <*gate node*>, are common to all of the flip-flops in the device.

Example

```
U5 SRFF(4)$G_DPWR $G_DGND PRESET CLEAR GATE
* four S-R latches
+ S0 S1 S2 S3  R0  R1  R2  R3
+ Q0 Q1 Q2 Q3 QB0 QB1 QB2 QB3
+ T_SRFF IO_STD
U2 DLTCH(8) $G_DPWR $G_DGND PRESET CLEAR GATE
* eight D latches
+ D0 D1 D2 D3 D4 D5 D6 D7
+ Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7
+ QB0 QB1 QB2 QB3 QB4 QB5 QB6 QB7
+ T_DLTCH IO_STD

.MODEL T_SRFF UGFF(...)    ; SRFF Timing Model - see below
```

Model Form

```
.MODEL <timing model name> UGFF [model parameters]
```

Table 3-12 *Gated Latch Timing Model Parameters*

Model Parameters*	Description	Units	Default
THDGMN	Hold: s/r/d after gate edge, min	sec	0
THDGTY	Hold: s/r/d after gate edge, typ	sec	0
THDGMX	Hold: s/r/d after gate edge, max	sec	0
TPDQLHMN	Delay: s/r/d to q/qb low to hi, min	sec	0
TPDQLHTY	Delay: s/r/d to q/qb low to hi, typ	sec	0
TPDQLHMX	Delay: s/r/d to q/qb low to hi, max	sec	0
TPDQHLMN	Delay: s/r/d to q/qb hi to low, min	sec	0
TPDQHLTY	Delay: s/r/d to q/qb hi to low, typ	sec	0
TPDQHLMX	Delay: s/r/d to q/qb hi to low, max	sec	0
TPGQLHMN	Delay: gate to q/qb low to hi, min	sec	0
TPGQLHTY	Delay: gate to q/qb low to hi, typ	sec	0
TPGQLHMX	Delay: gate to q/qb low to hi, max	sec	0
TPGQHLMN	Delay: gate to q/qb hi to low, min	sec	0
TPGQHLTY	Delay: gate to q/qb hi to low, typ	sec	0
TPGQHLMX	Delay: gate to q/qb hi to low, max	sec	0
TPPCQLHMN	Delay: preb/clrb to q/qb low to hi, min	sec	0
TPPCQLHTY	Delay: preb/clrb to q/qb low to hi, typ	sec	0
TPPCQLHMX	Delay: preb/clrb to q/qb low to hi, max	sec	0
TPPCQHLMN	Delay: preb/clrb to q/qb hi to low, min	sec	0
TPPCQHLTY	Delay: preb/clrb to q/qb hi to low, typ	sec	0
TPPCQHLMX	Delay: preb/clrb to q/qb hi to low, max	sec	0
TSUDGMN	Setup: s/r/d to gate edge, min	sec	0
TSUDGTY	Setup: s/r/d to gate edge, typ	sec	0
TSUDGMX	Setup: s/r/d to gate edge, max	sec	0
TSUPCGHMN	Setup: preb/clrb hi to gate edge, min	sec	0
TSUPCGHTY	Setup: preb/clrb hi to gate edge, typ	sec	0
TSUPCGHMX	Setup: preb/clrb hi to gate edge, max	sec	0
TWPCLMN	Min preb/clrb width low, min	sec	0

Table 3-12 *Gated Latch Timing Model Parameters (continued)*

Model Parameters*	Description	Units	Default
TWPCLTY	Min preb/clrb width low, typ	sec	0
TWPCLMX	Min preb/clrb width low, max	sec	0
TWGHMN	Min gate width hi, min	sec	0
TWGHTY	Min gate width hi, typ	sec	0
TWGHMX	Min gate width hi, max	sec	0

* See .MODEL statement.

Gated Latch Truth Tables

The function tables for the SRFF and DLTCH primitives are given in Table 3-13 and Table 3-14, below.

Table 3-13 *S-R Flip-Flop (SRFF) Truth Table*

Inputs					Outputs	
S	R	GATE	$\overline{PRE}$	$\overline{CLR}$	Q	$\overline{Q}$
X	X	X	1	0	0	1
X	X	X	0	1	1	0
X	X	X	0	0	1*	1*
X	X	0	1	1	Q′	$\overline{Q}$′
0	0	1	1	1	Q′	$\overline{Q}$′
0	1	1	1	1	0	1
1	0	1	1	1	1	0
1	1	1	1	1	1*	1*

* Shows an unstable condition.

Table 3-14 *D-Type Latch (DLTCH) Truth Table*

Inputs				Outputs	
D	**GATE**	**$\overline{\text{PRE}}$**	**$\overline{\text{CLR}}$**	**Q**	**$\overline{\text{Q}}$**
X	X	1	0	0	1
X	X	0	1	1	0
X	X	0	0	1*	1*
X	0	1	1	Q′	$\overline{\text{Q}}$′
0	1	1	1	0	1
1	1	1	1	1	0

* Shows an unstable condition.

Pullup and Pulldown

The PULLUP and PULLDN primitives function as digital pullup/pulldown resistors. They have no inputs (other than the digital power and ground nodes). Their output is a one level (pullup) or a zero level (pulldown), having a strength determined by the I/O model.

Device Format

U<*name*> <*resistor type*> (<*number of resistors*>)
+ <*digital power node*> <*digital ground node*>
+ <*output node*>*
+ <*I/O model name*>
+ [IO_LEVEL=<*interface subckt select value*>]

Example

```
U5 PULLUP(4) $G_DPWR $G_DGND; four pullup resistors
+ BUS0 BUS1 BUS2 BUS3  R1K
U2 PULLDN(1) $G_DPWR $G_DGND; one pulldown resistor
+ 15 R500
```

<*resistor type*> One of the following:

PULLUP	pullup resistor array
PULLDN	pulldown resistor array

<*number of resistors*>

Specifies the number of resistors in the array.

Notice that PULLUP and PULLDN do not have Timing Models, just I/O models.

Delay Line

The output of a delay line follows the input after the delay specified in the Timing Model. Any width pulse can propagate through a delay line. This behavior is different from gates, which don't propagate a pulse when its width is less than the propagation delay.

The delay line device has no parameters, and only one input and one output node.

Device Format

U<*name*> DLYLINE
+ <*digital power node*> <*digital ground node*>
+ <*input node*> <*output node*>
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY=<*delay select value*>]
+ [IO_LEVEL=<*interface subckt select value*>]

Example

```
U5 DLYLINE $G_DPWR $G_DGND IN OUT; delay line
+ DLY20NS IO_STD
.MODEL DLY20NS UDLY(; delay line Timing Model - see below
+ DLYMN=20ns DLYTY=20ns DLYMX=20ns
+ )
```

Timing Model Format

.MODEL <*timing model name*> UDLY [*model parameters*]

Table 3-15 *Delay Line Timing Model Parameters*

Model Parameters*	Description	Units	Default
DLYMN	Delay: min	sec	0
DLYTY	Delay: typical	sec	0
DLYMX	Delay: max	sec	0

* See .MODEL statement.

Programmable Logic Array

The programmable logic array is made up of a variable number of inputs, which form columns, and a variable number of outputs, which form rows. Each output (row) is driven by one logic gate. The “program” for the device determines which of the inputs (columns) are connected to each gate. All of the gates in the array are the same type (e.g., AND, OR, NAND, and NOR). Commercially available ICs (PALs, GALs, PEELs, and such) can have buffers, registers, more than one array of gates, and so on, all on the same part. These would normally be combined in a library subcircuit to make the part easier to use.

There are two ways to provide the program data for Programmable Logic Arrays. The normal way is to give the name of a JEDEC format file which contains the program data. This file would normally be produced by a PLD design package, or by using PLSyn, which translates logic design information into a program for a specific programmable logic part. The other way to program the logic array is by including the program data, in order, on the device line (using the DATA=... construct).

If one of the PAL or GAL devices are being used in the model library, you will **not** need to use the Programmable Logic Array primitive directly, nor any of the model information below, since the library contains all of the appropriate modeling information. Using a PLD from the library is just like using any other logic device from the library, except that the simulator needs to know the name of the JEDEC file which contains the program for that part. A TEXT parameter name JEDEC_FILE is used to specify the file name, as shown in the following example:

```
X1 IN1 IN2 IN3 IN4 IN5 IN6 IN7 IN8 IN9 IN10 IN11 IN12
+ IN13 IN14
+ OUT1 OUT2 OUT3 OUT4
+ PAL14H4
+ TEXT: JEDEC_FILE = "myprog.jed"
```

This example creates a 14H4 PAL which is programmed by the JEDEC file “myprog.jed.”

Device Format

U<*name*> <*pld type*> (<*no. of inputs*>, <*no. of outputs*>)
+ <*digital power node*> <*digital ground node*>
+ <*input_node*>* <*output_node*>*
+ <*timing model name*> <*I/O model name*>
+ [FILE=<(*file name*) *text value*>]
+ [DATA=<*radix flag*>$<*program data*>$]
+ [MNTYMXDLY=<*delay select value*>]
+ [IO_LEVEL=<*interface subckt select value*>]

Example

```
UDECODE PLANDC(3, 8); 3 inputs, 8 outputs
+ $G_DPWR $G_DGND; digital power supply and ground
+ IN1 IN2 IN3; the inputs
+ OUT0 OUT1 OUT2 OUT3 OUT4 OUT5 OUT6 OUT7  ; the outputs
+ PLD_MDL; the timing model name
+ IO_STD; the I/O model name
+ DATA=B$; the programming data
* IN1 IN2 IN3
* TF  TF  TF
+ 01  01  01; OUT0
+ 01  01  10; OUT1
+ 01  10  01; OUT2
+ 01  10  10; OUT3
+ 10  01  01; OUT4
+ 10  01  10; OUT5
+ 10  10  01; OUT6
+ 10  10  10 $ ; OUT7

.MODEL PLD_MDL UPLD(...) ; PLD timing model definition
                           ;- see below
```

<pld type> One of the following:

Table 3-16 *Programmable Logic Array Types*

PLD Type	Description
PLAND	AND array
PLANDC	AND array using true and complement columns for each input
PLNAND	NAND array
PLNANDC	NAND array using true and complement columns for each input
PLNOR	NOR array
PLNORC	NOR array using true and complement columns for each input
PLNXOR	Exclusive NOR array
PLNXORC	Exclusive NOR array using true and complement columns for each input
PLOR	OR array
PLORC	OR array using true and complement columns for each input
PLXOR	Exclusive OR array
PLXORC	Exclusive OR array using true and complement columns for each input

<file name text value>

The name of a JEDEC format file which specifies the programming data for the array. The file name can be specified as a text constant (enclosed in double quotes “ ”), or as a text expression (enclosed in vertical bars “|”). If a FILE is specified, any programming data specified by a DATA section is ignored. The mapping of addresses in the JEDEC file to locations in the array is controlled by model parameters specified in the timing model.

<radix flag> One of the following:

B	binary data follows
O	octal data follows (most significant bit has the lowest address)
X	hexadecimal data follows (most significant bit has lowest address)

<program data> A string of data values used to program the logic array. The values start at address zero, which programs the array for the connection of the first input pin to the gate which drives the first output. A "0" specifies that the input is not connected to the gate, and a "1" specifies that the input is connected to the gate. (Initially, all inputs are not connected to any gates.) The next value programs the array for the connection of the complement of the first input to the gate which drives the first output (if this is a programmable gate having true and complement inputs) or, the second input connection to the gate which drives the first output. Each additional "1" or "0" programs the connection of the next input or its complement to the gate which drives the first output, until the connection of all inputs (and their complements) to that gate have been programmed. Data values after that, program the connection of inputs to the gate driving the second output, and so on.

The data values must be enclosed in dollar signs ("$"), but can be separated by spaces or continuation lines.

The example below defines a 3-to-8 line decoder. The inputs are IN1 (MSB), IN2, and IN3 (LSB). If the inputs are all low, OUT0 is true. If IN1 and IN2 are low and IN3 is high, then OUT1 is true, and so on. The programming data has been typed in as an array, so that it is easier to read. The comments above the columns identify the true and false (complement) inputs, and the comments at the end of the line identify the output pin which is controlled by that gate. (Note, the simulator does not process any of these comments—they just help make the programming data readable.)

Timing Model Format `.MODEL <timing model name> UPLD [model parameters]`

Table 3-17 *Programmable Logic Array Timing Model Parameters*

Model Parameters*	Description	Units	Default
COMPOFFSET	JEDEC file mapping: address of complement of first input and first gate program		1

Table 3-17 *Programmable Logic Array Timing Model Parameters (continued)*

Model Parameters*	Description	Units	Default
INSCALE	JEDEC file mapping: amount the JEDEC file address changes for each new input pin	std true/cmp	1 2
OFFSET	JEDEC file mapping: address of first input and first gate program		0
OUTSCALE	JEDEC file mapping: amount the JEDEC file address changes for each new output pin (gate)	std true/cmp	*<no. of inputs>* 2**<no. of inputs>*
TPHLMN	Delay: in to out, hi to low, min	sec	0
TPHLTY	Delay: in to out, hi to low, typ	sec	0
TPHLMX	Delay: in to out, hi to low, max	sec	0
TPLHMN	Delay: in to out, low to hi, min	sec	0
TPLHTY	Delay: in to out, low to hi, typ	sec	0
TPLHMX	Delay: in to out, low to hi, max	sec	0

* See .MODEL statement.

Read Only Memory

There are two ways to provide the program data for ROMs. The normal way is to provide the name of an Intel Hex Format file. This file is read before the simulation starts, and the ROM is "programmed" to contain the data in the file. The other way to program the ROM is to include the program data on the device line (with the DATA=... construct).

Device Format

U<*name*> ROM(<*no. of address pins*>, <*no. of output pins*>)
+ <*digital power node*> <*digital ground node*>
+ <*enable_node*> <*address node msb*> ... <*address node lsb*>
+ <*output node msb*> ... <*output node lsb*>
+ <*timing model name*> <*I/O model name*>
+ [FILE=<*file name text value*>]
+ [DATA=<*radix flag*>$<*program data*>$]
+ [MNTYMXDLY=<*delay select value*>]
+ [IO_LEVEL=<*interface subckt select value*>]

Example

The following example defines a 4-bit by 4-bit to 8-bit multiplier ROM.

```
      UMULTIPLY ROM(8, 8)                          ; 8 address bits, 8
      outputs
      + $G_DPWR $G_DGND;digital power supply and ground
      + ENABLE                                     ; enable node
      + AIN3 AIN2 AIN1 AIN0                        ; the first 4 bits of
      address
      + BIN3 BIN2 BIN1 BIN0                        ; the second 4 bits of
      address
      + OUT7 OUT6 OUT5 OUT4 OUT3 OUT2 OUT1 OUT0 ; the outputs
      + ROM_MDL                                    ; the Timing Model name
      + IO_STD                                     ; the I/O MODEL name
      + DATA=X$                                    ; the programming data
      * B input value:
*   0  12   3 4 5   67  8  9A   BC  D   EF
+   00 0000 000000  000000 0000 000000  0000; A = 0
+   00 0102 030405  060708 090A 0B0C0D  0E0F; A = 1
+   00 0204 06080A  0C0E10 1214 16181A  1C1E; A = 2
+   00 0306 090C0F  121518 1B1E 212427  2A2D; A = 3
+   00 0408 0C1014  181C20 2428 2C3034  383C; A = 4
+   00 050A 0F1419  1E2328 2D32 373C41  464B; A = 5
+   00 060C 12181E  242A30 363C 42484E  545A; A = 6
+   00 070E 151C23  2A3138 3F46 4D545B  6269; A = 7
+   00 0810 182028  303840 4850 586068  7078; A = 8
+   00 0812 1B242D  363F48 515A 636C75  7E87; A = 9
+   00 0A14 1E2832  3C4650 5A64 6E7882  8C96; A = A
+   00 0B16 212C37  424D58 636E 79848F  9AA5; A = B
+   00 0C18 24303C  485460 6C78 84909C  A8B4; A = C
+   00 0D1A 273441  4E5B68 7582 8F9CA9  B6C3; A = D
+   00 0E1C 2A3846  546270 7E8C 9AA8B6  C4D2; A = E
+   00 0F1E 2D3C4B  5A6978 8796 A5B4C3  D1E1$; A = F
.MODEL ROM_MDL UROM(...); ROM Timing Model definition - see below
```

<file name text value>

The name of an Intel Hex format file which specifies the programming data for the ROM. The file name can be specified as a text constant (enclosed in double quotes " "), or as a text expression (enclosed in vertical bars "|"). If a FILE is specified, any programming data specified by a DATA section is ignored.

<radix flag> One of the following:

B	binary data follows
O	octal data follows (most significant bit has lowest address)
X	hexadecimal data follows (most significant bit has lowest address)

<program data> The program data is a string of data values used to program the ROM. The values start at address zero, first output bit. The next bit specifies the next output bit, and so on until all of the output bits for that address have been specified. Then the output values for the next address are given, and so on.

The data values must be enclosed in dollar signs ("$"), but can be separated by spaces or continuation lines.

Timing Model Format `.MODEL <timing model name> UROM (<model parameters>*)`

Table 3-18 *Read Only Memory Timing Model Parameters*

Model Parameters*	Description	Units	Default
TPADHMN	Delay: address to data, low to hi, min	sec	0
TPADHTY	Delay: address to data, low to hihi-Z, typ	sec	0
TPADHMX	Delay: address to data, low to hi, max	sec	0
TPADLMN	Delay: address to data, hi to low, min	sec	0
TPADLTY	Delay: address to data, hi to low, typ	sec	0
TPADLMX	Delay: address to data, hi to low, max	sec	0
TPEDHMN	Delay: enable to data, hi-Z to hi, min	sec	0
TPEDHTY	Delay: enable to data, hi-Z to hi, typ	sec	0
TPEDHMX	Delay: enable to data, hi-Z to hi, max	sec	0
TPEDLMN	Delay: enable to data, hi-Z to low, min	sec	0
TPEDLTY	Delay: enable to data, hi-Z to low, typ	sec	0
TPEDLMX	Delay: enable to data, hi-Z to low, max	sec	0
TPEDHZMN	Delay: enable to data, hi to hi-Z, min	sec	0
TPEDHZTY	Delay: enable to data, hi to hi-Z, typ	sec	0
TPEDHZMX	Delay: enable to data, hi to hi-Z, max	sec	0
TPEDLZMN	Delay: enable to data, low to hi-Z, min	sec	0
TPEDLZTY	Delay: enable to data, low to hi-Z, typ	sec	0
TPEDLZMX	Delay: enable to data, low to hi-Z, max	sec	0

* See .MODEL statement.

Random Access Read-Write Memory

The RAM is normally initialized using unknown data at all addresses. There are two ways to provide other initialization data for RAMs. The normal way is to give the name of an Intel Hex Format file. This file is read before the simulation starts, and the RAM is initialized to match the data in the file. The other way to initialize the RAM is to include the initialization data on the device line (using the DATA=... construct).

Device Format

U<*name*> RAM(<*no. of address bits*>, <*no. of output bits*>)
+ <*digital power node*> <*digital ground node*>
+ <*read enable node*> <*write enable node*>
+ <*address msb node*>...<*address lsb node*>
+ <*write-data msb node*>...<*write-data lsb node*>
+ <*read-data msb node*>...<*read-data lsb node*>
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY=<*delay select value*>]
+ [IO_LEVEL=<*interface subckt select value*>]
+ [FILE=<*file name text value*>]
+ [DATA=<*radix flag*>$<*initialization data*>$]

<*file name text value*>

The name of an Intel Hex format file which specifies the initialization data for the RAM. The file name can be specified as a text constant (enclosed in double quotes “ ”), or as a text expression (enclosed in vertical bars “|”). If a FILE is specified, any initialization data specified by a DATA section is ignored.

<*radix flag*> One of the following:

B	binary data follows
O	octal data follows (most significant bit has the lowest address)
X	hexadecimal data follows (most significant bit has the lowest address)

<*initialization data*>

A string of data values used to initialize the RAM. The values start at address zero, first output bit. The next bit specifies the next output bit, and so on until all of the output bits for that address have been specified. Then the output values for the next address are given, and so on.

The data values must be enclosed in dollar signs (“$”), but can be separated by spaces or continuation lines.

The initialization of a RAM using the DATA=... construct is the same as the programming of a ROM. See Read Only Memory section on page 3-39 on the ROM primitive for an example.

Timing Model Format `.MODEL <timing model name> URAM (<model parameters>*)`

The RAM has separate read and write sections, using separate data and enable pins, and shared address pins. To write to the RAM, the address and write data signals must be stable for the appropriate setup times, then write enable is raised. It must stay high for at least the minimum time, then fall. Address and data must remain stable while write enable is high, and for the hold time after it falls. Write enable must remain low for at least the minimum time before changing.

To read from the RAM, raise read enable, and the outputs change from Z (high impedance) to the appropriate value after a delay. The address can change while read enable is high, and if it does, the new data is available at the outputs after the delay.

Nothing prevents both the read and write enable from being true at the same time, although most real devices would not allow this. The new value from the write is sent to the read data outputs on the falling edge of write enable.

Table 3-19 *Random Access Memory Timing Model Parameters*

Model Parameters*	Description	Units	Default
TPADHMN	Delay: address to read data, low to hi, min	sec	0
TPADHTY	Delay: address to read data, low to hi, typ	sec	0
TPADHMX	Delay: address to read data, low to hi, max	sec	0
TPADLMN	Delay: address to read data, hi to low, min	sec	0
TPADLTY	Delay: address to read data, hi to low, typ	sec	0
TPADLMX	Delay: address to read data, hi to low, max	sec	0

Table 3-19 *Random Access Memory Timing Model Parameters*

Model Parameters*	Description	Units	Default
TPERDHMN	Delay: read enable to read data, hi-Z to hi, min	sec	0
TPERDHTY	Delay: read enable to read data, hi-Z to hi, typ	sec	0
TPERDHMX	Delay: read enable to read data, hi-Z to hi, max	sec	0
TPERDLMN	Delay: read enable to read data, hi-Z to low, min	sec	0
TPERDLTY	Delay: read enable to read data, hi-Z to low, typ	sec	0
TPERDLMX	Delay: read enable to read data, hi-Z to low, max	sec	0
TPERDHZMN	Delay: read enable to read data, hi to hi-Z, min	sec	0
TPERDHZTY	Delay: read enable to read data, hi to hi-Z, typ	sec	0
TPERDHZMX	Delay: read enable to read data, hi to hi-Z, max	sec	0
THAEWTY	Min hold time:write enable fall to address change, typ	sec	0
THAEWMX	Min hold time:write enable fall to address change, max	sec	0
THDEWMN	Min hold time:write enable fall to data change, min	sec	0
THDEWTY	Min hold time:write enable fall to data change, typ	sec	0
THDEWMX	Min hold time:write enable fall to data change, max	sec	0
THAEWMN	Min hold time:write enable fall to address change, min	sec	0
TPERDLZMN	Delay: read enable to read data, low to hi-Z, min	sec	0
TPERDLZTY	Delay: read enable to read data, low to hi-Z, typ	sec	0
TPERDLZMX	Delay: read enable to read data, low to hi-Z, max	sec	0

Table 3-19 *Random Access Memory Timing Model Parameters*

Model Parameters*	Description	Units	Default
TSUDEWMN	Min setup time: data to write enable rise, min	sec	0
TSUDEWTY	Min setup time: data to write enable rise, typ	sec	0
TSUDEWMX	Min setup time: data to write enable rise, max	sec	0
TSUAEWMN	Min setup time: address to write enable rise, min	sec	0
TSUAEWTY	Min setup time: address to write enable rise, typ	sec	0
TSUAEWMX	Min setup time: address to write enable rise, max	sec	0
TWEWHMN	Min width: enable write hi, min	sec	0
TWEWHTY	Min width: enable write hi, typ	sec	0
TWEWHMX	Min width: enable write hi, max	sec	0
TWEWLMN	Min width: enable write low, min	sec	0
TWEWLTY	Min width: enable write low, typ	sec	0
TWEWLMX	Min width: enable write low, max	sec	0

* See .MODEL statement.

Multi-Bit A/D and D/A Converter

The simulator provides two primitives to model analog-to-digital converters and digital-to-analog converters: the ADC and the DAC. These two primitives simplify the modeling of these complex mixed-signal devices.

Multi-Bit Analog-to-Digital Converter

Device Format

U<*name*> ADC(<*number of bits*>)
+ <*digital power node*> <*digital ground node*>
+ <*in node*> <*ref node*> <*gnd node*> <*convert node*>
+ <*status node*> <*over-range node*>
+ <*output msb node*> ... <*output lsb node*>
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY=<*delay select value*>]
+ [IO_LEVEL=<*interface subckt select value*>]

Example

```
U5 ADC(4) $G_DPWR $G_DGND; 4-bit ADC
+ Sig Ref 0 Conv Stat OvrRng Out3 Out2 Out1 Out0
+ ADCModel IO_STD

.MODEL ADCModel UADC(...); Timing Model - see below
```

Timing Model Format

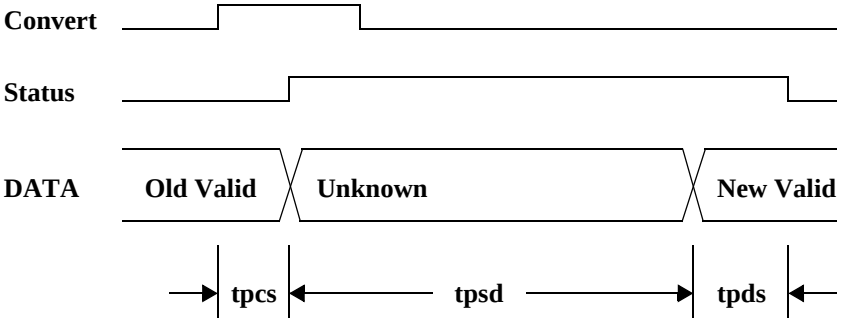
```
.MODEL <timing model name> UADC [model parameters]
```

Table 3-20 *Multi-Bit A/D Converter Timing Model Parameters*

Model Parameters*	Description	Units	Default
TPCSMN	Propagation delay: rising edge of convert to rising edge of status, min	sec	0
TPCSTY	Propagation delay: rising edge of convert to rising edge of status, typ	sec	0
TPCSMX	Propagation delay: rising edge of convert to rising edge of status, max	sec	0
TPDSMN	Propagation delay: data valid to falling edge of status, min	sec	0
TPDSTY	Propagation delay: data valid to falling edge of status, typ	sec	0
TPDSMX	Propagation delay: data valid to falling edge of status, max	sec	0
TPSDMN	Propagation delay: rising edge of status to data valid, min	sec	0
TPSDTY	Propagation delay: rising edge of status to data valid, typ	sec	0
TPSDMX	Propagation delay: rising edge of status to data valid, max	sec	0

* See .MODEL statement.

Figure 3-1 *ADC Primitive Device Timing*



DATA refers to both the data and over-range signals. The Convert pulse can be any width, including zero. If the propagation delay between the rising edge of the Convert signal and the Status signal (tpsd) is zero, the data and over-range do not go to unknown but directly to the new value. There is a resistive load from <*ref node*> to <*gnd node*>, and from <*in node*> to <*gnd node*>, of 1/GMIN.

The voltage at <*in node*> and <*ref node*> with respect to <*gnd node*> is sampled starting at the rising edge of the Convert signal, and ending when the Status signal becomes high. This gives a sample aperture time of tpcs plus any rising time for Convert. If, during the sample aperture, the output calculated having the minimum <*ref node*> voltage and maximum <*in node*> voltage is different from the output calculated having the maximum <*ref node*> voltage and minimum <*in node*> voltage, the appropriate output bits are set to the unknown state and a warning message is printed in the output file.

The output is the binary value of the nearest integer to

$$\frac{V(in, gnd)}{V(ref, gnd)} \cdot 2^{nbits}$$

If this value is greater than 2^{nbits}-1, then all data bits are 1, and over-range is 1. If this value is less than zero, then all data bits are zero, and over-range is 1.

Multi-Bit Digital-to-Analog Converter

Device Format

U<*name*> DAC(<*number of bits*>)
+ <*digital power node*> <*digital ground node*>
+ <*out node*> <*ref node*> <*gnd node*>
+ <*input msb node*> ... <*input lsb node*>
+ <*timing model name*> <*I/O model name*>
+ [MNTYMXDLY=<*delay select value*>]
+ [IO_LEVEL=<*interface subckt select value*>]

Example

```
U7 DAC(4) $G_DPWR $G_DGND; 4-bit DAC
+ Sig Ref 0 In3 In2 In1 In0
+ DACModel IO_STD

.MODEL DACModel UDAC(...); Timing model - see below
```

Timing Model Format

`.MODEL <timing model name> UDAC [model parameters]`

Table 3-21 *Multi-Bit D/A Converter Timing Model Parameters*

Model Parameters*	Description	Units	Default
TSWMN	Switching time: change in data to analog out stable, min	sec	10ns
TSWTY	Switching time: change in data to analog out stable, typ	sec	10ns
TSWMX	Switching time: change in data to analog out stable, max	sec	10ns

* See .MODEL statement.

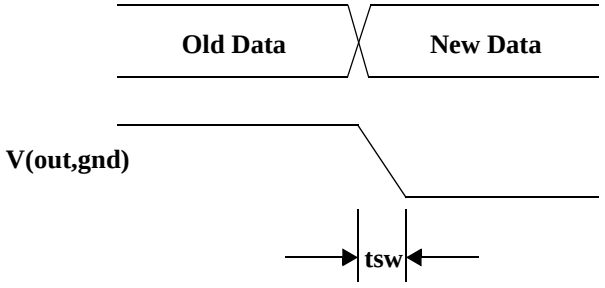
DAC Primitive Device Timing

The DAC is a zero impedance voltage source from <*out node*> to <*gnd node*>. The voltage is

$$V(ref, gnd) \cdot \frac{(binary\ \ value\ \ of\ \ inputs)}{2^{nbits}}$$

There is a resistance of 1/GMIN between <*ref node*> and <*gnd node*>.

If any inputs are unknown (X), the output voltage is halfway between the output voltage if all the “X” bits were “1” and the output voltage if all the “X” bits were “0.” When an input bit changes, the output voltage changes linearly to the new value during the switching time.



Behavioral Primitives

The simulator offers three primitives to aid in the modeling of complex digital devices: the *Logic Expression*, *Pin-to-Pin Delay*, and *Constraint Checker* primitives. These devices are distinct from other primitives in that they allow data-sheet descriptions to be specified more directly, allowing a one-to-one correspondence using the function diagrams and timing specifications.

The *Logic Expression* primitive, LOGICEXP, uses “free-format” logic expressions to describe the functional behavior device.

The *Pin-To-Pin Delay* primitive, PINDLY, describes propagation delays using sets of rules based on the activity on the device inputs.

The *Constraint Checker* primitive, CONSTRAINT allows a listing of timing rules such as setup/hold times, and minimum pulse widths. When a violation occurs, the simulator issues a message indicating the time of the violation and its cause.

Logic Expression

The LOGICEXP primitive allows combinational logic to be expressed in an equation-like style, using standard logic operators, node names, and temporary variables.

Device Format

U<*name*> LOGICEXP (<*no. of inputs*>, <*no. of outputs*>)
+ <*digital power node*> <*digital ground node*>
+ <*input node* 1> ... <*input node* n>
+ <*output node* 1> ... <*output node* n>
+ <*timing model name*>
+ <*I/O model name*>
+ [IO_LEVEL = <*value*>]
+ [MNTYMXDLY = <*value*>]
+ LOGIC:
+ <*logic assignment*>*

LOGIC: Marks the beginning of a sequence of one or more <*logic assignments*>. A <*logic assignment*> can have one of the two following forms:

<*output node*> = { <*logic expression* > }
<*temporary value*> = { <*logic expression*> }

<*output node*> One of the output node names as it appears in the interface list. Assignments to an <*output node*> causes the result of the <*logic expression*> to be scheduled on that output pin. Each <*output node*> must have exactly one assignment.

<*temporary value*> Any target of an assignment which is not specified as one of the nodes attached to the device defines a temporary variable. Once assigned, <*temporary values*> can be used inside subsequent <*logic expressions*>. They are provided to reduce the complexity and improve the readability of the model. The rules for node names apply to <*temporary value*> names.

<*logic expression*> A 'C'-like, infix-notation expression which returns one of the five digital logic levels. Like all other expressions, <*logic expressions*> must be surrounded by curly braces {...}. They can span one or more lines by using the '+' continuation character in the first column position. The logic operators are listed below from highest-to-lowest precedence.

Logic Expression Operators

~	unary not
&	and
^	exclusive or
|	or

The allowed operands are:

<*input nodes*>

Previously assigned <*temporary values*>

Previously assigned <*output nodes*>

<*logic constants*>: 0, 1, X, R, F

As in other expressions, parenthesis (...) can be used to group subexpressions. Note that these logic operators can also be used in Probe trace definitions.

Timing Model Format `.MODEL <timing model name> UGATE [model parameters]`

The LOGICEXP primitive uses the same timing model as the standard gate primitives, UGATE.

See the **Table 3-4 on page 3-14** for the list of UGATE model parameters.

Simulation Behavior

When a LOGICEXP primitive is evaluated during a transient analysis, the assignment statements using in it are evaluated in the order they were specified in the netlist. The logic expressions are evaluated using no delay. When the result is assigned to an output node, it is scheduled on that output pin using the appropriate delay specified in the timing model.

Internal feedback loops are not allowed in expressions. That is, an expression cannot reference a value which has yet to be defined. However, external feedback is allowed if the output node also appears on the list of input nodes.

This example models the functionality of the 74181 Arithmetic/Logic Unit. The logic for the entire part is contained in just one primitive. Timing would be handled by the PINDLY and CONSTRAINT primitives. Refer to any major device manufacturer’s data book for a detailed description of the operation of the 74181.

```
U74181 LOGICEXP( 14, 8 ) DPWR DGND
+ A0BAR A1BAR A2BAR A3BAR B0BAR B1BAR B2BAR B3BAR S0 S1 S2 S3 M CN
+ LF0BAR LF1BAR LF2BAR LF3BAR LAEQUALB LPBAR LGBAR LCN+4
+ D0_GATE IO_STD
+
+ LOGIC:
*
* Intermediate terms:
*
+ I31 = { ~((B3BAR & S3 & A3BAR) | (A3BAR & S2 & ~B3BAR)) }
+ I32 = { ~((~B3BAR & S1) | (S0 & B3BAR) | A3BAR ) }
+
+ I21 = { ~((B2BAR & S3 & A2BAR) | (A2BAR & S2 & ~B2BAR)) }
+ I22 = { ~((~B2BAR & S1) | (S0 & B2BAR) | A2BAR ) }
+
+ I11 = { ~((B1BAR & S3 & A1BAR) | (A1BAR & S2 & ~B1BAR)) }
+ I12 = { ~((~B1BAR & S1) | (S0 & B1BAR) | A1BAR ) }
+
+ I01 = { ~((B0BAR & S3 & A0BAR) | (A0BAR & S2 & ~B0BAR)) }
+ I02 = { ~((~B0BAR & S1) | (S0 & B0BAR) | A0BAR ) }
+
+ MBAR = { ~M }
+ P = { I31 & I21 & I11 & I01 }
*
* Output Assignments
*
+ LF3BAR = {(I31 & ~I32) ^
+    ~( (I21 & I11 & I01 & Cn & MBAR) | (I21 & I11 & I02 & MBAR ) |
+     (I21 & I12 & MBAR) | (I22 & MBAR) )}
+
+ LF2BAR = {(I21 & ~I22) ^
+    ~( (I11 & I01 & Cn & MBAR) | (I11 & I02 & MBAR) |
+     (I12 & MBAR) ) }
+
+ LF1BAR = {(I11 & ~I12) ^ ~( (Cn & I01 & MBAR) |
+     (I02 & MBAR) ) }
+
+ LF0BAR = { (I01 & ~I02) ^ ~(MBAR & Cn) }
+
+ LGBAR = { ~( I32 | (I31 & I22) | (I31 & I21 & I12) |
+     (I31 & I22 & I11 & I02) ) }
+
+ LCN+4 = { ~LGBAR | (P & Cn) }
+ LPBAR = { ~P }
+ LAEQUALB = { LF3BAR & LF2BAR & LF1BAR & LF0BAR }
```

Pin-to-Pin Delay

The pin-to-pin (PINDLY) primitive is a general mechanism which allows the modeling of complex device timing. It can be thought of as a set of delay-lines (paths) and rules describing how to associate specific amounts of delay using each path.

A PINDLY primitive is used in the output path of a device model, typically at the output pins of a subcircuit definition. A single PINDLY primitive can model the timing and output characteristics of an entire part, including tristate behavior.

PINDLY primitives are expressed and evaluated in a manner similar to the LOGICEXP primitive, except in this case a *delay expression* is "assigned" to each output. Whenever an output path undergoes a transition, its delay expression is evaluated to determine the propagation delay which is to be applied to that change.

A delay expression can contain one or more rules that determine which activity on the part's inputs is responsible for the output change, for example "is the output changing because the clock changed or the data changed?" This allows device models to be derived directly from data sheets, which typically specify propagation delays based on which input is changing. The PINDLY primitive uses its *reference* inputs to determine the logic state and recent transitions on nodes which are not in the output path.

Pin-to-pin delay modeling is much simpler compared to earlier methods, in which input-to-output delays had to be distributed among the low-level primitives used to model the device. The latter method can require a great deal of trial and error because manufacturer's data sheets do not provide a one-to-one association between the logic diagram and the timing specifications.

PINDLY primitives can also contain constraints such as setup/hold, width, and frequency specifications, like those supported by the CONSTRAINT primitive. When used in the PINDLY primitive, these constraints allow the simulator to propagate hazard conditions and report violations in subsequent logic.

Device Format

U<*name*> PINDLY (<*no. of paths*>, <*no. of enables*>, <*no. of refs*>)
+ <*digital-power-node*> <*digital-ground-node*>
+ <*input node* 1> ... <*input node* n>
+ [<*enable node* 1> ... <*enable node* n>]
+ [<*reference node* 1> ... <*reference node* n>]
+ <*output node* 1> ... <*output node* n>
+ <*I/O model name*>
+ [MNTYMXDLY = <*delay select value*>]
+ [IO_LEVEL = <*interface subckt select value*>]
+ [BOOLEAN:
+ <*boolean assignment*>*]
+ PINDLY:
+ <*delay assignment*>*
+ [TRISTATE:
+ ENABLE LO | HI <*enable node*>
+ <*delay assignment*>*]
+ [SETUP_HOLD: <*setup-hold-specification*>]
+ [WIDTH: <*width-specification*>]
+ [FREQ: <*frequency-specification*>]
+ [GENERAL: <*general-specification*>]

<*no. of paths*>

Specifies the number of *input*-to-*output* paths represented by the device; the number of inputs *must* be equal to the number of outputs. A *path* is defined as an input-to-output association, having the appropriate delay rules started according to the described conditions.

<*no. of enables*>

Specifies the number of tristate enable nodes used by the primitive. Enable nodes are used in TRISTATE sections. <*no. of enables*> can be zero.

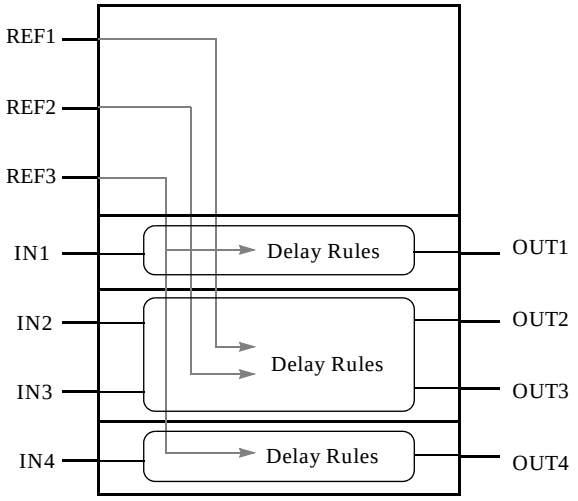
<*no. of refs*>

Specifies the number of reference nodes used by the primitive. Reference nodes are used within delay expressions to get state information about signals which are not in the input-to-output paths. <*no. of refs*> can be zero.

An example is presented to help visualize the relationship and purpose of the different pins on the PINDLY primitive. A primitive which has been specified as the following with the structure depicted

Example

```
U1 PINDLY(4,0,3) $G_DPWR $G_DGND
+ IN1 IN2 IN3 IN4
+ REF1 REF2 REF3
+ OUT1 OUT2 OUT3 OUT4
+ IO_MODEL D0_GATE
+ PINDLY:
+ ...
```



Comments

The PINDLY primitive can be viewed as four buffers, IN1 to OUT1 through IN4 to OUT4, and three reference nodes which are used by the output delay rules. The figure shows how the reference nodes can be used in one or more set of delay rules. In this case, REF1 and REF2 are used by the delay rules for OUT2, and REF3 is used by the delay rules for OUT1 and OUT4. The figure also shows that OUT2 and OUT3 can share the same delay rules. The remainder of the format description describes how to create delay rules.

BOOLEAN: Marks the beginning of a section of one or more <*boolean assignments*>, which define temporary variables that can be used in subsequent <*delay expressions*>. BOOLEAN sections can appear in any order within the PINDLY primitive. A <*boolean assignment*> has the following form:

<*boolean variable*> = { <*boolean-expression*> }

<*boolean variable*>

Can be any name which follows the node name rules.

<*boolean expression*>

A 'C'-like, infix-notation expression which returns the boolean value TRUE or FALSE. Like all other expressions, <*boolean expressions*> must be surrounded by curly braces {...}. They can span one or more lines by using the '+' continuation character in the first column position. The boolean operators are listed below from highest-to-lowest precedence.

Boolean Expression Operators

~	unary not
==	equality
!=	inequality
&	and
^	exclusive or
|	or

All boolean operators take the following boolean values as operands:

- Previously assigned <*boolean variables*>
- *Reference functions* (defined below)
- *Transition functions* (defined below)
- <*boolean constants*>: TRUE, FALSE

In addition, the "==" and "!=" operators take logic values, such as <*input nodes*> and <*logic constants*>. This allows for a check of the values on nodes, for example "CLEAR == '1" returns TRUE if the current level on the node CLEAR is a logic one and FALSE otherwise.

Reference Functions

Reference functions are used to detect changes (transitions) on <*reference nodes*> or <*input nodes*>. All reference functions return boolean values, and therefore can be used within any <*boolean*

expression>. Following is the list of available reference functions and their arguments:

CHANGED(<*node*>, <*delta time*>)
CHANGED_LH(<*node*>, <*delta time*>)
CHANGED_HL(<*node*>, <*delta time*>)

The CHANGED function returns TRUE if the specified <*node*> has undergone any state transition within the past <*delta time*>, prior to the current simulation time; otherwise it returns FALSE.

Similarly, CHANGED_LH returns TRUE if <*node*> has specifically undergone a low-to-high transition within the past <*delta time*>; FALSE otherwise. Note that CHANGED_LH only looks at the *most recent* (or current) transition. It cannot, for example, determine if 0 → 1 occurred two transitions ago.

Finally, CHANGED_HL is similar to CHANGED_LH, but checks for high-to-low transitions.

If a <*delta time*> is specified zero, the reference functions return TRUE if the node has changed at the current simulation time. This allows all of the functionality of a device to be modeled in zero delay so that the total delay through the device can be described using the delay expressions.

Transition Functions

Transition functions are used to determine the state change occurring on the *changing output*, that is, the <*output node*> for which the <*delay expression*> is being evaluated. Like reference functions, transition functions return boolean values. However, they differ from reference functions in that transition functions take no arguments, since they implicitly refer to the *changing output* at the *current time*. The transition functions are of the general form:

TRN_*pn*

where *p* is the "previous" state value and *n* is the "new" state value. State values are taken from the set { L H Z $ }. Where appropriate, the "$" can be used to signify "don't care," e.g., a TRN_H$ matches a transition from "H" to ANY state. Rising states automatically map to High, and Falling states automatically map to Low.

As a term in any boolean expression, for example, TRN_LH takes on a TRUE value if the *changing output* is propagating a change from zero to one.

Following is the complete set of transition functions.

TRN_LH TRN_LZ TRN_L$ TRN_HL TRN_HZ TRN_H$ TRN_ZL TRN_ZH TRN_Z$ TRN_$L TRN _$H TRN_$Z

Note *The TRN_pZ and TRN_Zn functions return true only if it is used within a TRISTATE section, described below. Although open-collector outputs also transition to a high-impedance Z (instead of H), most data books describe propagation times on open-collector outputs as TPLH or TPHL. Therefore, open-collector output devices should use TRN_LH and TRN_HL, and tristate output devices should use TRN_LZ, TRN_HZ, TRN_ZL, and TRN_ZH.*

PINDLY: Marks the beginning of a section of one or more <*delay assignments*>, which are used to associate propagation delays using the PINDLY primitive's outputs. <*delay assignments*> are of the form:

<*output node*>* = { <*delay expression*> }

<*output node*> One of the output node names as it appears in the interface list. Each <*output node*> must have exactly one assignment. Several outputs can "share" the same delay rules by listing them (separated by spaces or commas) on the left-hand side of the <*delay expression*>.

<*delay expression*> An expression which, when evaluated, returns a triplet (min, typ, max) of delay values. Like all other expressions, <*delay expressions*> must be surrounded by curly braces {...}. They can span one or more lines by using the '+' continuation character in the first column position.

The simplest <*delay expression*> is a single <*delay value*>, defined as:

DELAY(<*min*>, <*typ*>, <*max*>)

where <*min*>, <*typ*>, and <*max*> are floating point constants or expressions (involving parameters), expressed in seconds. To specify unknown values, use -1. For example, DELAY(20ns,-1,35ns) specifies a minimum time of 20ns, a default (program-computed) value for typical, and a maximum of 35ns. See Treatment of Unspecified Propagation Delays on page **3-10** for more information on default delays.

Example

```
...
+ PINDLY:
+    Y = { DELAY(2ns, 5ns, 9ns) }
...
```

The delay assignment above specifies the propagation delays through output Y to be: min=2ns, typ=5ns, and max=9ns.

To define more complex, rule-based <*delay expressions*>, use the CASE function, which has the form:

CASE(
<*boolean expression*>, <*delay expression*>, ; Rule 1
<*boolean expression*>, <*delay expression*>, ; Rule 2
... ; ...
<*delay expression*> ; Default delay
)

The arguments to the CASE function are pairs of <*boolean expressions*> and <*delay expressions*>, followed by a final default <*delay expression*>. <*boolean expressions*> (described above) can contain <*boolean values*>, reference functions, and transition functions.

When the CASE function is evaluated, each <*boolean expression*> is evaluated in order of appearance until one produces a TRUE result. When this occurs, the <*delay expression*> it is paired with the result of the CASE function, and the evaluation of the CASE is ended. If none of the <*boolean expressions*> return a TRUE result, the value of the *final* <*delay expression*> is used. Because it is possible for all <*boolean expressions*> to evaluate FALSE, the default delay value *must* be supplied. Note that each argument to the CASE function must be separated by commas.

```
...
+ BOOLEAN:
+    CLOCK = { CHANGED_LH( CLK, 0 ) }
+ PINDLY:
+    QA QB QC QD = {
+       CASE (
+          CLOCK & TRN_LH, DELAY(-1,13ns,24ns),
+          CLOCK & TRN_HL, DELAY(-1,18ns,27ns),
+          CHANGED_HL( CLRBAR,0), DELAY(-1,20ns,28ns),
+          DELAY(-1,20ns,28ns)    ; Default
+          )
+       }
```

This example describes the delays through a four-bit counter. It shows how rules can be defined to precisely isolate the cause of the output change. In this example, the boolean variable CLOCK is being defined. It is TRUE whenever the reference input CLK changes from low-to-high at the current simulation time. This is only true if the device functionality is modeled in zero delay.

The four outputs QA through QD all share the same delay expression. The CASE is used to specify different delays when the device is counting or clearing. The first two rules define delays when the device is counting (CLK changing low-to-high); the first when the output (QA through QD) is going from low-to-high, the second from high-to-low.

The third rule simply uses the CHANGED_HL function directly to determine whether CLRBAR is changing, and in this case the specification applies to any change (low-to-high or high-to-low) on the output. The default delay applies to all other output transitions which are not covered by the first three rules.

TRISTATE: Marks the beginning of a sequence of one or more <*delay assignments*>. The TRISTATE section differs from the PINDLY section in that the outputs are controlled by the specified enable node.

Immediately following the TRISTATE keyword, an enable node must be specified using its polarity and the ENABLE keyword:

ENABLE HI <*enable node*> ; Specifies active HI enable

ENABLE LO <*enable node*> ; Specifies active LO enable

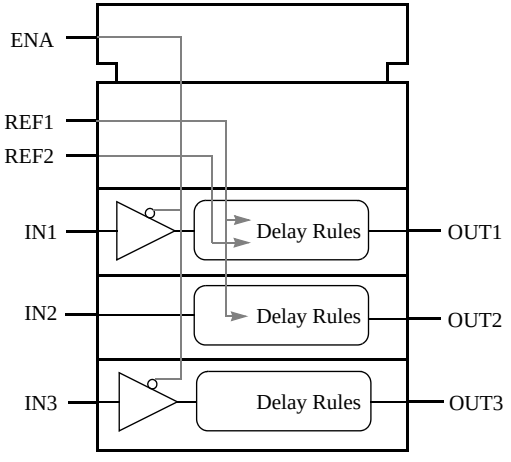
The specified <*enable node*> applies to all <*output node*> assignments in the current section.

Note *Note that <delay expressions> within a TRISTATE section can contain the transition functions pertaining to the Z state, for example TRN_ZL and TRN_HZ.*

The following example demonstrates how an enable node can be used to control more than one output. It also shows that some device outputs can use the standard output (PINDLY) while others use the tristate output. (Delay values have been omitted.)

Example

```
U1 PINDLY(3,1,2) $G_DPWR $G_DGND
+ IN1 IN2 IN3
+ ENA
+ REF1 REF2
+ OUT1 OUT2 OUT3
+ IO_MODEL
+ TRISTATE:
+ENABLE LO = ENA
+OUT1 = {
+CASE(
+CHANGED(REF1, 0) & TRN_LH, DELAY(...),
+CHANGED(REF2, 0), DELAY(...),
+TRN_ZL,    DELAY(...),
+...
+)
+}
+OUT3 = {
+CASE(
+TRN_LZ,DELAY(...),
+TRN_HZ,DELAY(...),
+...
+)
+}
+ PINDLY:
+OUT2 = {
+CASE(
+CHANGED(REF1,0),DELAY(...),
+...
+)
+}
```



1. Each CONSTRAINT clause operates independently of all others within a device.
2. By default, for violations involving <*input node*>, the message tag propagates to the <*output node*> having positional correspondence.
3. By default, for violations involving <*reference node*>, the message tag propagates to ALL <*output node*>s.
4. The default behavior can be overridden by use of one of the following statements, which can appear anywhere within any constraint clause proper:
   AFFECTS (#OUTPUTS) = <*output node*> { ... }
   AFFECTS_ALL
5. AFFECTS_NONE is always the default for the GENERAL constraint.

SETUP-HOLD:
WIDTH:
FREQ:
: Marks the beginning of a constraint specification. These constructs have the same syntax as those used in the CONSTRAINT primitive (see page 3-75).

GENERAL:
: When a PINDLY primitive is used, the constraint specifications allow the simulator to not only report timing violations, but also to track the effects of the violations in downstream logic. This allows more serious persistent hazards to be reported. This behavior differs from the CONSTRAINT primitive, which only reports timing violations.

PINDLY Primitive Simulation Behavior

A PINDLY primitive is evaluated whenever any of its <*input nodes*> or <*enable nodes*> change. The <*input node*> is positionally associated using its corresponding <*output node*>. The BOOLEAN statements up to the output assignment are evaluated first, then the appropriate PINDLY or TRISTATE <*delay expression*> which has been assigned to the changing <*output node*> is evaluated. The changing input's state is then applied to the output, using its delay value.

The following PINDLY primitive models the timing behavior of a 74LS160A counter. This example is derived directly from the device model in the model library.

Example

```
ULS160ADLY PINDLY(5,0,4) DPWR DGND
+ RCO QA QB QC QD; Inputs
+ CLK LOADBAR ENT CLRBAR; Reference nodes
+ RCO_O QA_O QB_O QC_O QD_O; Outputs
+ IO_LS MNTYMXDLY={MNTYMXDLY} IO_LEVEL={IO_LEVEL}
+
+ BOOLEAN:
+   CLOCK = { CHANGED_LH(CLK,0) }
+   CNTENT = { CHANGED(ENT,0) }
+
+ PINDLY:
+   QA_O QB_O QC_O QD_O = {
+     CASE(
+        CLOCK & TRN_LH, DELAY(-1,13NS,24NS),
+        CLOCK & TRN_HL, DELAY(-1,18NS,27NS),
+        CHANGED_HL(CLRBAR,0), DELAY(-1,20NS,28NS),
+        DELAY(-1,20NS,28NS); Default
+        )
+      }
+
+   RCO = {
+     CASE(
+       CNTENT, DELAY(-1,9NS,14NS),
+       CLOCK & TRN_LH, DELAY(-1,18NS,35NS),
+       CLOCK & TRN_HL, DELAY(-1,18NS,35NS),
+       DELAY(-1,20NS,35NS); Default
+       )
+     }
```

Constraint Checker

The CONSTRAINT primitive provides a general “constraint checking” mechanism to the digital device modeler. It performs *setup* and *hold time* checks, *pulse width* checks, *frequency* checks, and includes a *general* mechanism to allow user-defined conditions to be reported.

The CONSTRAINT primitive only reports timing violations. It does not affect propagated or stored logic state or propagation delays.

Timing specifications are usually given at the “device” (i.e., package pin) level. Thus, the “inputs” to the constraint description typically are those of the subcircuit description of the device, after any necessary buffering. CONSTRAINT devices can be used in conjunction with any combination of digital primitives, including gates, logic expressions, and pin-to-pin delay primitives.

Device Format

U<*name*> CONSTRAINT (<*no. of inputs*>)
+ <*digital power node*> <*digital ground node*>
+ <*input node 1*> ... <*input node n*>
+ <*I/O model name*>
+ [IO_LEVEL = <*interface subckt select value*>]
+ [BOOLEAN: <*boolean assignment*>*] ...
+ [SETUP_HOLD: <*setup_hold specification*>] ...
+ [WIDTH: <*width specification*>] ...
+ [FREQ: <*frequency specification*>] ...
+ [GENERAL: <*general specification*>] ...

BOOLEAN:

Marks the beginning of a section containing one or more <*boolean assignments*>, of the form:

<*boolean variable*> = { <*boolean expression*> }

BOOLEAN sections can appear in any order within the CONSTRAINT primitive.

The syntax of the <*boolean expression*> is the same as that defined in the PINDLY primitive reference, having the exception that transition functions have no meaning within the CONSTRAINT primitive.

SETUP_HOLD:

Marks the beginning of a setup/hold constraint specification, which has the following format:

```
+ SETUP_HOLD:
+   CLOCK <assertion edge> = <input node>
+   DATA ( <no. of data inputs> ) = <input node j> ... <input node k>
+   [ SETUPTIME = <time value> ]
+   [ HOLDTIME = <time value> ]
+   [ RELEASETIME = <time value> ]
+   [ WHEN {<boolean expression>} ]
+   [ MESSAGE = "<additional message text>" ]
+   [ ERRORLIMIT = <value> ]
+   [ AFFECTS_ALL | AFFECTS_NONE |
+       AFFECTS (#OUTPUTS) = <output-node-list> ]
```

Note *One or more sections can be specified in any order. Note that AFFECTS clauses are only allowed in PINDLY primitives.*

CLOCK defines the node to be used as the reference for setup/hold/release specification. <*assertion edge*> is one of "LH" or "HL," and specifies which edge of the CLOCK node the setup/hold time is measured against. The CLOCK node must be specified.

DATA defines one or more nodes to be the nodes whose setup/hold time is being measured. At least one DATA node must be specified.

SETUPTIME defines the minimum time that all DATA nodes must be stable prior to the <*assertion edge*> of the clock. The <*time value*> must be a nonnegative constant or expression, expressed in seconds. Some devices have different setup time requirements which depend on whether the data is a low or a high at the time of the clock change. In this case, one or both of the following can be used:

SETUPTIME_LO = <*time value*>
SETUPTIME_HI = <*time value*>

instead of SETUPTIME, which defines both low- and high-level specifications. If one or both SETUPTIME_xx specifications is zero, the simulator does not perform a setup check for that data level.

HOLDTIME defines the minimum time that all DATA nodes must be stable after the <*assertion edge*> of the clock. The <*time value*> must be a nonnegative constant or expression, expressed in seconds. Some devices have different hold time requirements which depend on whether the data is a low or a high at the time of the clock change. In this case, one or both of the following can be used:

HOLDTIME_LO = <*time value*>
HOLDTIME_HI = <*time value*>

instead of HOLDTIME, which defines both low- and high-level specifications. If one or both HOLDTIME_xx specifications is zero, the simulator does not perform a hold check for that data level.

RELEASETIME specifications cause the simulator to perform a special-purpose setup check. In a data sheet, release time (also called recovery time) specifications refer to the minimum time a signal (such as "CLEAR") can go inactive *before* the active CLOCK edge. In other words, release times refer to the position of a specific data *edge* in relation to the clock edge. For this reason, one or both of the following can be used:

RELEASETIME_LH = <*time value*>
RELEASETIME_HL = <*time value*>

instead of RELEASETIME, which defines both LH- and HL-edge specifications. The <*time value*> must be a nonnegative constant or expression, expressed in seconds.

The difference between the release-time checker and the setup-time checker is that simultaneous CLOCK/DATA changes are never allowed in the release-time check. That is, a nonzero hold time is assumed, even though the HOLDTIME is not specified. This feature allows the data

sheet values to be specified for release-times directly in a model. For this reason, release times are usually given alone, and not in conjunction with SETUPTIME or HOLDTIME specifications.

Simulation Behavior

The sequence of setup/hold/release checks begins when the CLOCK node undergoes the specified LH or HL transition. At that time, the WHEN expression is evaluated. If the result is TRUE, all checks using nonzero specifications are performed for during this clock cycle. If the result is FALSE, then no setup, hold, or release checks are performed. The WHEN expression is used in device models to block the reporting of violations when the device is not “listening” to the DATA inputs, such as during a clearing function.

The simulator performs setup-time checks when the CLOCK node undergoes an <*assertion edge*>. If the HOLDTIME specification is zero, simultaneous CLOCK/DATA transitions are allowed, however the previous value of DATA is still checked for setup-time. If the HOLDTIME is not zero, simultaneous CLOCK/DATA transitions are reported as a HOLDTIME violation.

The simulator performs hold-time checks on any DATA node that changes *after* the <*assertion edge*> on the CLOCK node. If the SETUPTIME is zero, simultaneous CLOCK/DATA changes are allowed, and the next transition on DATA which occurs before the non-asserting clock edge is checked for a hold-time violation.

The simulator performs release-time checks when the CLOCK node undergoes an <*assertion edge*>. Simultaneous CLOCK/DATA transitions are not allowed, and is flagged as a violation.

If either the CLOCK or DATA node is unknown (X) at the time of a check, no violation is reported for that node. This reduces the number of unnecessary warning messages: an X being clocked into a device is usually a symptom of another problem which has already been reported.

The sequence ends when the CLOCK node undergoes the “other” (non-asserting) edge. At this time, any violations which occurred during that clock cycle are reported. (This makes it possible for violations to appear out of time-order in the “.out” file.)

WIDTH: Marks the beginning of a minimum pulse-width constraint specification, which has the following format:

```
+ WIDTH:
+   NODE = <input node>
+   [ MIN_HI = <time value> ]
+   [ MIN_LO = <time value> ]
+   [ WHEN {<boolean expression>} ]
+   [ MESSAGE = "<additional message text>" ]
+   [ ERRORLIMIT = <value> ]
+   [ AFFECTS_ALL | AFFECTS_NONE |
+       AFFECTS (#OUTPUTS) = <output-node-list> ]
```

Note *One or more sections can be specified in any order. Note that AFFECTS clauses are only allowed in the PINDLY primitive.*

NODE defines the input node whose pulse width is to be checked.

MIN_HI specifies the minimum time that the <*input node*> can remain at a high (1) logic level. The <*time value*> must be a nonnegative constant or expression, expressed in seconds. If not specified, MIN_HI defaults to 0, meaning that any width HI pulse is allowed.

MIN_LO likewise specifies the minimum time that the <*input node*> can remain at a low (0) logic level. The <*time value*> must be a nonnegative constant or expression, expressed in seconds. If not specified, MIN_LO defaults to 0, meaning that any width LO pulse is allowed.

At least one instance of MIN_HI or MIN_LO must appear within a WIDTH specification.

FREQ: Marks the beginning of a frequency constraint specification, which has the following format:

```
+ FREQ:
+   NODE = <input node>
+   [ MINFREQ = <frequency value> ]
+   [ MAXFREQ = <frequency value>]
+   [ WHEN { <boolean expression> }]
+   [ MESSAGE "<additional message text>" ]
+   [ ERRORLIMIT = <value> ]
+   [ AFFECTS_ALL | AFFECTS_NONE |
+       AFFECTS (#OUTPUTS) = <output-node-list> ]
```

Note *One or more sections can be specified in any order. Note that AFFECTS clauses are only allowed in the PINDLY primitive.*

NODE defines the input node whose frequency is to be checked.

MINFREQ specifies the minimum frequency allowed on <*input node*>. The <*frequency value*> must be a nonnegative floating point constant or expression, expressed in hertz.

MAXFREQ specifies the maximum frequency allowed on <*input node*>. The <*frequency value*> must be a nonnegative floating point constant or expression, expressed in hertz.

At least one of MINFREQ or MAXFREQ must be specified within a FREQ specification.

Simulation Behavior

When performing a MINFREQ check, the simulator reports a violation whenever the duration of a period on the <*input node*> is greater than 1/<*frequency value*>. Likewise, when performing a MAXFREQ check, it reports a violation whenever any period is less than 1/<*frequency value*>. To avoid large numbers of violations, the simulator does not report subsequent violations until *after a valid* cycle occurs.

Note that the use of *maximum* FREQ specifications provides a slightly different functionality from that achieved by use of *minimum* pulse-width checks: in the FREQ specification case, the *duty-cycle* characteristic of the signal is not measured or constrained in any way, whereas the pulse-width check effectively defines the allowable *duty-cycle*.

Some clocked state-storage device specifications include information about maximum clock frequency, but omit *duty-cycle* information.

GENERAL: Marks the beginning of a general condition test. GENERAL constraints have the following form:

```
+ GENERAL:
+   WHEN { <boolean expression> }
+   MESSAGE = "<message text>"
+   [ ERRORLIMIT = <value> ]
+   [ AFFECTS_ALL | AFFECTS_NONE |
+       AFFECTS (#OUTPUTS) = <output-node-list> ]
```

Note *One or more sections can be specified in any order. Note that AFFECTS clauses are only allowed in the PINDLY primitive. The default for the GENERAL constraint is AFFECTS_NONE.*

WHEN is used to define a boolean expression, which can describe arbitrary signal relationships that represent the “error” or condition of interest.

MESSAGE defines the message to be reported by the simulation whenever the WHEN expression evaluates TRUE. The <*message text*> must be a text constant (enclosed by double quotes “ ”) or a text expression.

Simulation Behavior

The <*boolean expression*> is evaluated whenever the CONSTRAINT primitive is evaluated, that is, whenever any of its inputs undergo a transition. If the result is TRUE, the simulator produces a header containing the time of the occurrence, followed by the <*message text*>.

General Notes

Any or all of the constraint specifications (SETUP_HOLD, WIDTH, FREQ, GENERAL) can appear, in any order, within a CONSTRAINT primitive. Further, more than one constraints of the same type can appear (such as two WIDTH specifications). Each of the constraint specifications is evaluated whenever any inputs to the CONSTRAINT primitive instance change.

All constraint specifications can optionally include a WHEN statement, which is interpreted as "only perform the check when result of <*boolean expression*> == TRUE." The WHEN statement is required in the GENERAL constraint.

Each constraint type (SETUP_HOLD, WIDTH, FREQ, and GENERAL) has an associated "built-in" message. In addition, each instance can include a MESSAGE specification, which takes a text constant (enclosed in double quotes " ") or text expression. The <*additional message text*> is appended to the end of the internally-generated, type-specific message which is output whenever a violation occurs. The MESSAGE clause is *required* for the GENERAL constraint device.

All of the constraint specifications can accept an optional ERRORLIMIT specification. The <*value*> must be a nonnegative constant or expression. The default <*value*> is obtained from the value of the DIGERRDEFAULT (set using the .OPTIONS command,) which defaults to 20. A value of zero is interpreted as "infinity", i.e., no limit. When more than <*value*> violations of the associated constraint have occurred, no further message output is generated *for that constraint checker;* other checkers within the CONSTRAINT primitive that have not exceeded their own ERRORLIMITs continue to operate.

During simulation, if the total number of digital violations reported exceeds the value given by DIGERRLIMIT (set using the .OPTIONS command,) the simulation is halted. DIGERRLIMIT defaults to "infinity."

This CONSTRAINT primitive below was derived from the 74LS160A device in the model library. It demonstrates how all of the timing checks can be performed by a single primitive.

Example

```
ULS160ACON CONSTRAINT(10) DPWR DGND
+ CLK ENP ENT CLRBAR LOADBAR A B C D EN
+ IO_LS
+ FREQ:
+   NODE = CLK
+   MAXFREQ = 25MEG
+ WIDTH:
+   NODE = CLK
+   MIN_LO = 25NS
+   MIN_HI = 25NS
+ WIDTH:
+   NODE = CLRBAR
+   MIN_LO = 20NS
+ SETUP_HOLD:
+   DATA(1) = LOADBAR
+   CLOCK LH = CLK
+   SETUPTIME = 20NS
+   HOLDTIME = 3NS
+   WHEN = { CLRBAR!='0 }
+ SETUP_HOLD:
+   DATA(2) = ENP ENT
+   CLOCK LH = CLK
+   SETUPTIME = 20NS
+   HOLDTIME = 3NS
+   WHEN = { CLRBAR!='0 & (LOADBAR!='0 ^
CHANGED(LOADBAR,0))
+     & CHANGED(EN,20NS) }
+ SETUP_HOLD:
+   DATA(4) = A B C D
+   CLOCK LH = CLK
+   SETUPTIME = 20NS
+   HOLDTIME = 3NS
+   WHEN = { CLRBAR!='0 & (LOADBAR!='1 ^
CHANGED(LOADBAR,0)) }
+ SETUP_HOLD:
+   DATA(1) = CLRBAR
+   CLOCK LH = CLK
+   RELEASETIME_LH = 25NS
```

Stimulus Devices

Stimulus devices apply digital waveforms to a node. Their purpose is to provide the input to a digital circuit or a digital portion of a mixed circuit. They play the same role in the digital simulator that the independent voltage and current sources (V and I devices) do in the analog simulator.

There are two types of stimulus devices: the stimulus generator (STIM), which uses a simple command to generate a wide variety of waveforms; and the file stimulus (FSTIM), which obtains the waveforms from an external file.

Unlike digital primitives, stimulus devices do not have a Timing Model. This is similar to the analog V and I devices: the timing characteristics are described by the device itself, not in a separate model.

Stimulus Generator

Device Format

U<*name*> STIM(<*width*>, <*format array*>)
+ <*digital power node*> <*digital ground node*>
+ <*node*>*
+ <*I/O model name*>
+ [STIMULUS=<*stimulus name*>]
+ [IO_LEVEL=<*interface subckt select value*>]
+ [TIMESTEP=<*stepsize*>]
+ <*command*>*

<*width*> Specifies the number of signals (nodes) output by the stimulus generator.

<*format array*> Specifies the format of <*value*>s used in defining the stimulus. <*format array*> is a sequence of digits which specifies the number of signals (nodes) that the corresponding digit in a <*value*> represents. Each digit of <*value*> is assumed to be in base $2^{<m>}$ where <*m*> is the corresponding digit in <*format array*>. Each <*value*> must have the same number of digits as <*format array*>. The sum of the digits in <*format array*> must be <*width*>, and each digit must be either a 1, 3, or 4 (that is, binary, octal, or hexadecimal).

<*digital power node*> <*digital ground node*>

These nodes are used by the interface devices which connect analog nodes to digital nodes or vice versa. Refer to your PSpice user's guide for more information.

<*node*>* One or more node names which are output by the stimulus generator. The number of nodes specified must be the same as <*width*>.

<*I/O model name*> The name of an I/O model, which describes the driving characteristics of the stimulus generator. I/O models also contain the names of up to four DtoA interface subcircuits, which are automatically called by the simulator to handle interface nodes. In most cases, the I/O model named IO_STM can be used from the "dig_io.lib" library file. Refer to your PSpice user's guide for a more detailed description of I/O models.

STIMULUS An optional parameter for referencing a stimulus definition.

IO_LEVEL — An optional device parameter which selects one of the four DtoA interface subcircuits from the I/O model. The simulator calls the selected subcircuit automatically in the event a <*node*> connects to an analog device. If not specified, IO_LEVEL defaults to 0. Valid values are:

0 = the current value of .OPTIONS DIGIOLVL (default=1)
1 = DtoA1
2 = DtoA2
3 = DtoA3
4 = DtoA4

Refer to the *Circuit Analysis User's Guide* for more information.

TIMESTEP — Number of seconds per clock cycle, or step. Transition times that are specified in clock cycles (using the "C" suffix) are multiplied by this amount to determine the actual time of the transition. (See <*time*> below.) If TIMESTEP is not specified, the default is zero seconds. TIMESTEP has no effect on <*time*> values which are specified in seconds (using the "S" suffix).

<*command*>* — A description of the stimuli to be generated, using one or more of the following.

```
<time> <value>
LABEL=<label name>
<time> GOTO <label name> <n> TIMES
<time> GOTO <label name> UNTIL GT <value>
<time> GOTO <label name> UNTIL GE <value>
<time> GOTO <label name> UNTIL LT <value>
<time> GOTO <label name> UNTIL LE <value>
<time> INCR BY <value>
<time> DECR BY <value>
REPEAT FOREVER
REPEAT <n> TIMES
ENDREPEAT
FILE=<file name>
```

<*time*> — Specifies the time for the new <*value*>, GOTO, or INCR/DECR command to occur.

Time Units

Time values can be stated in seconds or in clock cycles (see TIMESTEP above). To specify a time value in clock cycles, use the "C" suffix. Otherwise, the units default to seconds.

Absolute/Relative Times

Times can be *absolute*, such as 45ns or 10c, or *relative* to the previous time. To specify a relative time, prefix the time using a "+" such as +5ns or +2c.

<*value*> Value for each node (0, 1, R, F, X, or Z). <*value*> is interpreted using the <*format array*>.

<*label name*> Name used in GOTO statements. GOTO <*label name*> jumps to the next non-label statement after the <LABEL = <*label name*>> statement.

<*n*> Number of times to repeat a GOTO loop. Use a -1 to specify "forever."

Keep the following in mind when using the stimulus command:

Transitions using absolute times within a GOTO loop are converted to relative times based on the time of the previous command and the current step size.

- GOTO <*label name*> must specify a label that has been defined in a previous LABEL=<*label name*> statement.
- Times must be in strictly ascending order, except that the transition after a GOTO can be at the same time as the GOTO.

A simpler syntax for constructing "counted loops" in digital stimulus is to use the REPEAT/ENDREPEAT construct. Specify the count value, for example:

Example

```
REPEAT 3 TIMES
+ 5ns  0
+ 5ns  1
ENDREPEAT
```

For an infinite loop, use REPEAT FOREVER (equivalent to REPEAT -1 TIMES). All times within REPEAT loops are interpreted as "relative" to the start of the loop.

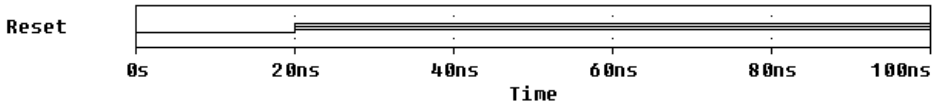
Transition (i.e., time-value pairs) information can be placed in a FILE and accessed one or more times from the STIM device by using the "FILE=" statement. The syntax for the file contents is identical to what can appear directly in the body of the STIM device <*command*> section.

Stimulus Generator Examples

Example One

The first example creates a simple reset signal, which could be used to set or clear a flip-flop at the beginning of a simulation. The node, named Reset, is set to a level zero at time zero nanoseconds, and to a Z (high impedance) at 20 ns.

```
UReset STIM(1,1) $G_DPWR $G_DGND
+ Reset
+ IO_STM
+ 0s 0
+ 20ns Z
```

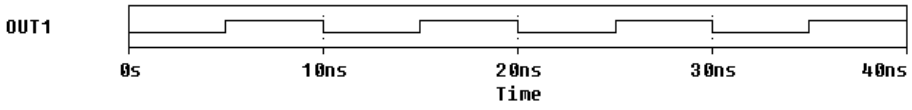


This is useful when the Reset node is being driven by another device which does not reset the flip-flop at time zero. By using the library I/O model named IO_STM, the stimulus generator drives with a high strength, and thus overpowers the other output. By outputting a Z for the duration of the simulation, the stimulus generator cannot affect the node.

Example Two

The second example is a simple example of a clock stimulus which pulses every 5 nanoseconds. It has one output node, OUT1, and the format is represented in binary notation. This example specifies the time as relative to the previous step. IO_STM is an I/O model for stimulus devices and is available in the “dig_io.lib” library file which comes with the digital simulation feature.

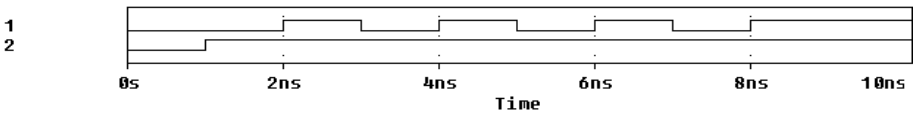
```
UEx2 STIM( 1, 1 ) $G_DPWR $G_DGND  Out1  IO_STM
+ 0s 0; At time=0 initialize Out1 ; to zero.
+ REPEAT FOREVER;repeats loop indefinitely
+ +5ns 1         ;5ns later Out1 is set to 1
+ +5ns 0         ;5ns later Out1 is set to 0
+ ENDREPEAT
```



Example Three

The third example illustrates the use of the timestep; a cycle is equal to one nanosecond:

```
UEx3 STIM( 2, 11 ) $G_DPWR $G_DGND  1 2
+ IO_STM TIMESTEP=1ns
+ 0c 00          ;At time=0ns, both nodes are set to 0.
+ REPEAT 4 TIMES ;What's in the loop is repeated
                 ;4 times
+ +1c 01         ;1ns later node 1 is set to 0
                 ;and node 2 is set to 1.
+ +2c 11         ;2ns later both nodes set to 1.
+ ENDREPEAT
```



Example Four

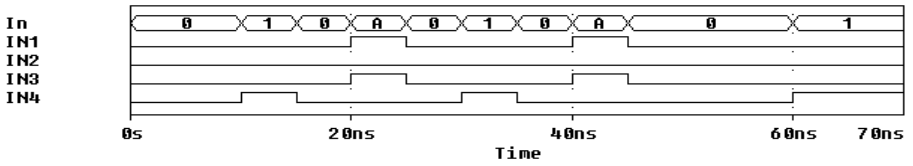
The fourth example has four output nodes. The values of the nodes at each transition are in hexadecimal notation. This is because the <*format array*> is set to 4, meaning <*value*> is one digit representing the value of four nodes. Both the absolute and relative timing methods are used, but, at the start of execution, the simulation converts all absolute values to relative values based on the time of the command and the current step size. The timestep is equal to one nanosecond, setting the cycle to one nanosecond:

```
UEx4 STIM( 4, 4 ) $G_DPWR $G_DGND  IN1 IN2 IN3 IN4
+ IO_STM TIMESTEP=1ns
+ 0s 0    ; At time=0 seconds, all nodes are set to 0.
+ LABEL=STARTLOOP
+ 10C 1   ; At time=10NS, IN1, IN2, & IN3 are set to 0 and IN4
          ;is set to 1.
+  +5NS 0 ; 5NS later, all nodes are set to 0.
+  20NS A ; At time=20NS, nodes IN1 & IN3 are set to 1 and
          ;nodes IN2 &
          ; IN4 are to 0.
+ +5NS 0  ; 5NS later, all nodes are set to 0.
+ 30C GOTO STARTLOOP 1 TIMES ; At time=30NS, execute the
          ;first statement of the loop without
          ;a further delay."1 TIMES" causes the logic to loop
          ; 1 time, actually executing the loop twice.
+ +10C 1  ; After the logic falls through the loop
          ;the second
          ; time and then waiting 10 additional cycles
          ; (or 10 nanoseconds),
   ;IN1, IN2, & IN3 are set to 0 and IN4 is set to 1.
```

Example four produces the following transitions. Note how all of the time values are calculated relative to the previous step:

```
 TIME     VALUE
0.00E+00 = 0000
1.00E-08 = 0001  ; STARTLOOP
1.50E-08 = 0000
2.00E-08 = 1010  ; 1010 in hex=A
2.50E-08 = 0000
3.00E-08 = 0001  ; The GOTO STARTLOOP 1 TIMES causes the
                 ;first statement
                 ; after the STARTLOOP label to be executed
                 ;immediately.
3.50E-08 = 0000
4.00E-08 = 1010
4.50E-08 = 0000  ; At time 5.00E-08 we checked the
                 ;GOTO STARTLOOP
```

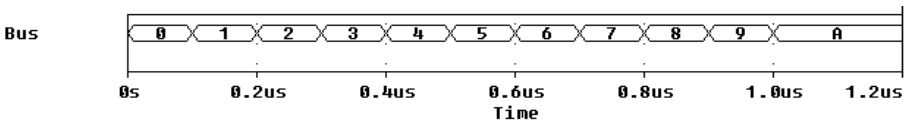
```
                ; 1 TIMES statement, but did not execute it
                ; since it was already completed one time.
6.00E-08 = 0001  ;At 10C=1ns * 10=10ns later we
                 ;execute the
                 ;last statement.
```



Example Five

The fifth example illustrates the use of the “INCR BY” command used to increment the value of the 16 bit bus:

```
UEx5 STIM ( 16, 4444 ) $G_DPWR $G_DGND
+ 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
+ IO_STM  TIMESTEP = 10ns
+ 0s 0000 ; At time=0 seconds, all nodes are set to 0.
+ LABEL=STARTLOOP
+ 10c INCR BY 0001     ; At 100ns, increment bus by 1.
+ 20c GOTO STARTLOOP UNTIL GE 000A   ; If the bus value
                 ;is less
                 ; than 10, branch back to STARTLOOP and
                 ; execute the line following the
                 ; label without a further delay.
```

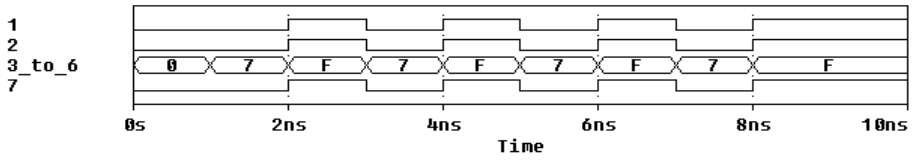


Example Six

The sixth example has seven output nodes: 1, 2, 3, 4, 5, 6, and 7. The <*format array*> specifies the notation (1=binary, 3=octal, or 4=hex) used to define the output of those seven nodes. The first two output signals are defined in binary, the next four are in hexadecimal, and the last one is in binary. In this example, at
time equal to one nanosecond, the value of 0070 creates the bit pattern 0001110 on the output nodes. The first two zeros correspond to outputs one and two, the 0111 (7 in hex) corresponds to output signals 3 through

6, and the last zero
is the value of output signal 7.

```
UEx6 STIM( 7, 1141 ) $G_DPWR $G_DGND  1 2 3 4 5 6 7
 IO_STM
+ 0ns 0000       ; At time=0ns, all nodes are set to 0.
+ REPEAT 4 TIMES ; Repeats what's in loop 4 times.
+ +1ns 0070      ; At time=1ns, nodes 1, 2, & 3 are set to 0,
                 ; nodes 4, 5, & 6 are set to 1,
                 ; and node 7 is set to 0.
+ +2ns 11F1; At time=2ns, all nodes are set to 1.
+ ENDREPEAT
```



File Stimulus

The file stimulus device, FSTIM, allows the digital stimuli to be obtained from a file. This is often useful if the number of stimuli is very large, or if the inputs to one simulation come from the output of another simulation (or even from another simulator). To make the discussion of the FSTIM device more meaningful, the stimulus file format is discussed first.

Stimulus File Format

The stimulus file has a simple format which allows outputs from other simulators, or the simulation output file, to be used with little modification. The file consists of two sections: the *header*, which contains a list of signal names, and the *transitions*, which is one or more lines containing the transition time and columns of values. The header and transitions must be separated by at least one blank line. Below is a simple example of the stimulus file format.

Example

```
* Header, containing signal names (standard comments are
* allowed)
Clock, Reset, In1, In2; four signal names

* Beginning of the transitions - note the blank line
0    0000    ; values are in binary
10ns 1100
20ns 0101
30ns 1110
40ns 0111
```

Header Format

```
[TIMESCALE=<value>]
<signame 1>...<signame n>...
OCT(<signame bit 3> ... <signame lsb>) ...
HEX(<signame bit 4> ... <signame lsb>) ...
```

The header consists of the list of signal names and an optional TIMESCALE value. The signal names can be separated by commas, spaces, or tabs. The list can span several lines, but must **NOT** include the "+" continuation character. The signal names listed correspond to the columns of values in the order that they are listed. Up to 255 signals can be listed in the header, however a maximum of 300 characters are allowed per line.

The OCT and HEX radix functions allows three or four signals to be grouped, respectively, into a single octal or hexadecimal digit in the columns of values. Note that exactly three signals must be included inside the parentheses in the OCT function, and that exactly four signals must be included in the HEX function. Signal names listed without the radix functions default to binary values.

The following example shows the use of the HEX radix function.

Example

```
Clock Reset In1 In2
HEX(Addr7 Addr6 Addr5 Addr4) HEX(Addr3 Addr2 Addr1 Addr0)
ReadWrite

0   0000 00 0; spaces can be used to group values
10n 1100 4E 0
20n 0101 4E 1
30n 1110 4E 1
40n 0111 FF 0
```

In this example, there are four binary signals, followed by two occurrences of the HEX radix function, followed by a single binary

signal. In the list of transitions following the header, there are seven values which correspond, in order, to the list of signals.

The optional TIMESCALE assignment is used to scale the time values in the transitions. The TIMESCALE assignment must be on a separate line. If unspecified, TIMESCALE defaults to 1.0. See <*time*> below for more information on the use of TIMESCALE.

Transition Format

<*time*> <*value*>*

Following the first blank line after the header, the simulator looks for one or more lines containing transitions. Transitions consist of a time value, followed by one or more values corresponding to the signal names in the header. The <*time*> and list of <*values*> must be separated by at least one space or tab.

<*time*>

Transition times are always stated in seconds. Times can be *absolute*, such as 45ns, 1.2e-8, or 10; or *relative* to the previous time. To specify relative time, prefix the time using a "+," such as +5ns or +1e-9.

Time values are always scaled by the value of TIMESCALE. This is useful if the time values in the file are expressed as whole numbers, but the actual units are, for example, 10ns. An example showing the use of TIMESCALE is given below.

<*value*>*

Each value corresponds to a single binary signal (the default) or the entire group of signals inside the OCT or HEX radix functions. The number of values listed must equal the total number of binary signals and radix functions which are specified in the header. Valid <*values*> are:

	Binary	OCT	HEX
Logic/Numeric	0,1	0-7	0-F
Unknown	X	X	X
Hi-impedance	Z	Z	Z
Rising	R	R	
Falling	F	F	

When the <*value*> in a HEX or OCT column is a number, the simulator converts the number to binary and assigns the appropriate logic value of each bit (either zero or one) to the signals inside the radix function. The bits are assigned msb to lsb. When the <*value*> is X, Z, R, or F, all

signals in the radix function take on that value. Note that there can be no "falling" value in a HEX column because "F" is used as a numeric value.

The following example shows the use of TIMESCALE and relative <time> values.

Example

```
TIMESCALE=10ns   ; must appear on separate line
Clock, Reset, In1, In2
HEX(Addr7 Addr6 Addr5 Addr4) HEX(Addr3 Addr2 Addr1 Addr0)
ReadWrite

0  0000 00 0
1  110R 4E 0     ; transition occurs at 10ns
2  0101 4E 1
+  3 1111 4E 1   ; transition occurs at 50ns
7  011F C3 0     ; transition occurs at 70ns
8  11X0 C3 1
```

File Stimulus Device

The file stimulus device, FSTIM, is used to access one or more signals inside a stimulus file. More than one FSTIM device can access the same file. An FSTIM device can even refer to the same signal as another FSTIM device. Any number of stimulus files can be used during a simulation.

Device Format

U<*name*> FSTIM(<*# outputs*>)
+ <*digital power node*> <*digital ground node*>
+ <*node*>*
+ <*I/O model name*>
+ FILE=<*stimulus file name*>
+ [IO_LEVEL=<*interface subckt select value*>]
+ [SIGNAMES=<*stimulus file signal name*>*]

<*# outputs*> Specifies the number of nodes driven by this device.

<*digital power node*> <*digital ground node*>
These nodes are used by the interface devices which connect analog nodes to digital nodes or vice versa. Refer to the *Circuit Analysis User's Guide* for more information.

<*node*>* One or more node names which are output by the file stimulus. The number of nodes specified must be the same as <*# outputs*>.

<*I/O model name*> The name of an I/O model, which describes the driving characteristics of

the stimulus device. I/O models also contain the names of up to four DtoA interface subcircuits, which are automatically called by the simulator to handle interface nodes. In most cases, the I/O model named IO_STM can be used from the "dig_io.lib" library file. Refer to the *Circuit Analysis User's Guide* for a more detailed description of I/O models.

FILE

The name of the stimulus file to be accessed by this device. The <*stimulus file name*> can be specified as a quoted string or as a text expression (see .TEXT (Text Parameter) on page **1-67**). Note that the FILE device parameter is not optional.

IO_LEVEL

An optional device parameter which selects one of the four AtoD or DtoA interface subcircuits from the device's I/O model. The simulator calls the selected subcircuit automatically in the event a node connecting to the primitive also connects to an analog device. If not specified, IO_LEVEL defaults to 0. Valid values are:

0 = the current value of .OPTIONS DIGIOLVL (default=1)
1 = AtoD1/DtoA1
2 = AtoD2/DtoA2
3 = AtoD3/DtoA3
4 = AtoD4/DtoA4

Refer to your PSpice user's guide for more information.

SIGNAMES

Used to specify the names of the signals inside the stimulus file which are to be referenced by the FSTIM device. The signal names correspond, in order, to the <*nodes*> connected to the device. If any or all SIGNAMES are unspecified, The simulator looks in the stimulus file for the names of the <*nodes*> given. Because the number of signal names can vary, the SIGNAMES parameter must be specified last. SIGNAMES can be a list of names or text expressions (see .TEXT), or a mixture of the two.

The first example references a file named "dig1.stm." This file must have a signal named IN1.

Example

```
U1 FSTIM(1) $G_DPWR $G_DGND
+ IN1 IO_STM FILE=DIG1.STM
```

The second example references “dig2.stm.” This file would have to have signals named AD3 through AD0. These are mapped, in order, to the nodes ADDR3 through ADDR0, which are driven by this device.

```
U2 FSTIM(4) $G_DPWR $G_DGND
+ ADDR3 ADDR2 ADDR1 ADDR0
+ IO_STM
+ FILE = DIG_2.STM
+ SIGNAMES = AD3 AD2 AD1 AD0
```

In the third example, the FSTIM device references the file “flipflop.stm.”

```
U3 FSTIM(4) $G_DPWR $G_DGND
+ CLK PRE J K
+ IO_STM
+ FILE = FLIPFLOP.STM
+ SIGNAMES = CLOCK PRESET
```

The contents of “flipflop.stm” are shown below:

```
J K PRESET CLEAR CLOCK

0    0 0 010
10ns 0 0 111
.
.
.
```

In this example, the first two nodes, CLK and PRE, reference the signals named CLOCK and PRESET in the stimulus file. The last two nodes, J and K, directly reference the signals named J and K in the file, and therefore do not need to be listed in SIGNAMES. Note that the order of the SIGNAMES on the FSTIM device does not need to match the order of the names listed in the header of the stimulus file. It is not required that every signal in the file be referenced by an FSTIM device. In the example above, the signal named CLEAR is not referenced. One, several, or all signals in a stimulus file can be referenced by one or more FSTIM devices.

Input/Output Model

Each digital device in the circuit must reference an I/O model. The I/O model describes the device's loading and driving characteristics. It also contains the names of up to four AtoD and DtoA subcircuits that the simulator calls to handle interface nodes.

I/O models are common to device families. For example, of the digital devices in the model library, there are only four I/O Models for the entire 74LS family: IO_LS, for standard inputs and outputs; IO_LS_OC, for standard inputs and open-collector outputs; IO_LS_ST, for schmitt trigger inputs and standard outputs; and IO_LS_OC_ST, for schmitt trigger inputs and open-collector outputs. This is in contrast to timing models, which are unique to each device in the library.

Model Form

`.MODEL <I/O model name> UIO [model parameters]`

Table 3-22 *Input/Output Model Parameters*

UIO Model Parameters	Description	Units	Default
AtoD1	Name of level 1 AtoD interface subcircuit		AtoDDefault
AtoD2	Name of level 2 AtoD interface subcircuit		AtoDDefault
AtoD3	Name of level 3 AtoD interface subcircuit		AtoDDefault
AtoD4	Name of level 4 AtoD interface subcircuit		AtoDDefault
DIGPOWER	Name of power supply subcircuit		DIGIFPWR
DRVH	Output high level resistance	ohm	50
DRVL	Output low level resistance	ohm	50
DRVZ	Output Z-state leakage resistance	ohm	250 Kohm
DtoA1	Name of level 1 DtoA interface subcircuit		DtoADefault
DtoA2	Name of level 2 DtoA interface subcircuit		DtoADefault
DtoA3	Name of level 3 DtoA interface subcircuit		DtoADefault

Table 3-22 *Input/Output Model Parameters (continued)*

UIO Model Parameters	Description	Units	Default
DtoA4	Name of level 4 DtoA interface subcircuit		DtoADefault
INLD	Input load capacitance	farad	0
INR	Input leakage resistance	ohm	30 Kohm
OUTLD	Output load capacitance	farad	0
TPWRT	Pulse width rejection threshold	sec	same as propagation delay
TSTOREMN	Minimum storage time for net to be simulated as a charge	sec	1.0 msec
TSWHL1	Switching time high to low for DtoA1	sec	0
TSWHL2	Switching time high to low for DtoA2	sec	0
TSWHL3	Switching time high to low for DtoA3	sec	0
TSWHL4	Switching time high to low for DtoA4	sec	0
TSWLH1	Switching time low to high for DtoA1	sec	0
TSWLH2	Switching time low to high for DtoA2	sec	0
TSWLH3	Switching time low to high for DtoA3	sec	0
TSWLH4	Switching time low to high for DtoA4	sec	0

INLD and OUTLD are used in the calculation of loading capacitance, which factors into the propagation delay. Refer to your PSpice user's guide for more information.

DRVH and DRVL are used to determine the strength of the output. Refer to your PSpice user's guide for more information.

DRVZ, INR, and TSTOREMN are used to determine which nets should be simulated as charge storage nets.

AtoD1 through AtoD4 and DtoA1 through DtoA4 are used to hold the names of interface subcircuits. Note that INLD and AtoD1 through AtoD4 do not apply to stimulus generators because they have no input nodes. Refer to your PSpice user's guide for more information.

The switching times (TSWLHn and TSWHLn) are subtracted from a device's propagation delay on the outputs which connect to interface nodes. This compensates for the time it takes the DtoA device to change its output voltage from its current level to that of the switching threshold. By subtracting the switching time from the propagation delay, the analog signal reaches the switching threshold at the correct time (that is, at the exact time of the digital transition). The values for these model parameters should be obtained by measuring the time it takes the analog output of the DtoA (using a nominal analog load attached) to change to the switching threshold after its digital input changes. If the switching time is larger than the propagation delay for an output, no warning is issued, and a delay of zero is used. Note that the switching time parameters are not used when the output drives a digital node.

DIGPOWER specifies the name of the power supply subcircuit the simulator calls for when an AtoD or DtoA interface is created. The default value is DIGIFPWR, which is the power supply subcircuit used by the TTL and CMOS device libraries.

For more information on how to change the default power supplies, refer to your PSpice user's guide.

Digital/Analog Interface Devices

The simulator provides two devices for converting digital logic levels to analog voltages or vice versa. These devices are at the heart of the interface subcircuits found in “dig_io.lib”. These devices also provide the Digital Files interface for interfacing using external logic simulators.

Digital Input (N Device)

The digital input device is used to translate logic levels (typically 1s, 0s, Xs, Zs, Rs, and Fs) into representative voltage levels using series resistances. These voltages and resistances model the output stage of a logic device (like a 74LS04) and hence form a “digital input” to the analog circuit. The logic level information can come from two places: The digital simulator, or a file. (The file can be created by hand, or can be an output file from an external logic simulator.)

Note *The following is General Form for digital simulation.*

General Form

N<*name*> <*interface node*> <*low level node*> <*high level node*>
+ <*model name*>
+ DGTLNET = <*digital net name*>
+ <*digital I/O model name*>
+ [IS = *initial state*]

Note *The following is General Form for digital files.*

General Form

N<*name*> <*interface node*> <*low level node*> <*high level node*>
+ <*model name*>
+ [SIGNAME = <*digital signal name*>]
+ [IS = *initial state*]

Example

```
N1 ANALOG DIGITAL_GND DIGITAL_PWR DIN74
+ DGTLNET=DIGITAL_NODE IO_STD
NRESET 7 15  16 FROM_TTL
N12 18 0 100 FROM_CMOS SIGNAME=VCO_GATE IS=0
```

Model Form

```
.MODEL <model name> DINPUT [model parameters]
```

Table 3-23 *Digital Input Model Parameters*

Model Parameters*	Description	Units	Default
CHI	Capacitance to high level node	farad	0
CLO	Capacitance to low level node	farad	0
FILE	Digital input file name (Digital Files only)		
FORMAT	Digital input file format (Digital Files only)		1
S0NAME	State “0” character abbreviation		
S0TSW	State “0” switching time	sec	
S0RLO	State “0” resistance to low level node	ohm	
S0RHI	State “0” resistance to high level node	ohm	
S1NAME	State “1” character abbreviation		
S1TSW	State “1” switching time	sec	
S1RLO	State “1” resistance to low level node	ohm	
S1RHI	State “1” resistance to high level node	ohm	
S2NAME	State “2” character abbreviation		
S2TSW	State “2” switching time	sec	
S2RLO	State “2” resistance to low level node	ohm	
S2RHI	State “2” resistance to high level node	ohm	
.	.		
.	.		
.	.		
S19NAME	State “19” character abbreviation		
S19TSW	State “19” switching time	sec	
S19RLO	State “19” resistance to low level node	ohm	
S19RHI	State “19” resistance to high level node	ohm	
TIMESTEP	Digital input file step-size (Digital Files only)	sec	1E-91

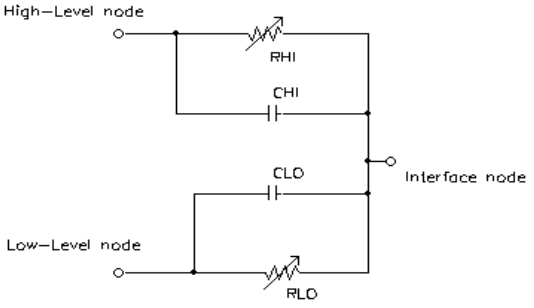
* See .MODEL statement.

The general form for a digital input device, and some of the model parameters, are different for devices which are “driven” from a file, and those which are “driven” by the digital simulation feature. The digital simulation inserts digital input devices *automatically* when a digital device’s output is connected to an analog component. The automatic insertion of digital input devices is discussed in the *Circuit Analysis User’s Guide*. Examples of the devices which are inserted can be found in the “dig_io.lib” library file.

Note *For more information on using the digital input device to simulate mixed analog/digital systems refer to your PSpice user’s guide.*

As shown in Figure 3-2, the digital input device is modeled as a time varying resistor from <*low level node*> to <*interface node*>, and another time varying resistor from <*high level node*> to <*interface node*>. Each of these resistors has an optional fixed value capacitor in parallel: CLO and CHI. When the state of the digital signal changes, the values of the resistors change (exponentially) from their present values to the values specified for the new state over the switching time specified by the new state. Normally the low and high level nodes would be attached to voltage sources which would correspond to the highest and lowest logic levels. (Using two resistors and two voltage levels, any voltage between the two levels can be created at any impedance.

Figure 3-2 *Digital Input Model*



For a digital simulation driven digital input, the parameters

DGTLNET = <*digital net name*> <*digital I/O model name*>

must be specified. Refer to your PSpice user's guide for more information on digital I/O models. The digital net *must not* be connected to any analog devices, otherwise the automatic analog/digital interface process disconnects the digital input device from the digital net.

Digital simulation can send states named "0," "1," "X," "R," "F," and "Z" to a digital input device. The simulation stops if the digital simulation sends a state which is not modeled (does not have SnNAME, SnTSW, SnRLO, and SnRHI specified) to a digital input device.

The initial state of a digital simulation driven digital input is controlled by the bias point solution of the analog/digital system. It is sometimes necessary to override this solution (for example, an oscillator which contains both analog and digital parts). The optional parameter

IS = <*initial state name*>

can be used to do this. The digital input remains in the initial state until the digital simulation value changes from its TIME=0 value.

The model parameters FILE, FORMAT, and TIMESTEP are not used by digital simulation driven digital input devices, and only the FILE parameter is used for VIEW*sim* A/D driven digital inputs. For file driven digital inputs the FILE parameter defines the name of the file to be read, and the FORMAT parameter defines the format of the data in that file. The TIMESTEP parameter defines the conversion between the digital simulation's integer timing tick numbers and the simulation's floating-point time values:

tick number · TIMESTEP = seconds

Note *Tick number must be an integer.*

For a file driven or VIEW*sim* A/D driven digital input, the DGTLNET parameter must not be specified, but the optional parameter

SIGNAME = <*digital signal name*>

is used to specify the name of the digital signal in the file (or the digital net name in VIEW*sim* A/D). If no SIGNAME is given, then the portion of the device name after the leading N identifies the name of the digital signal.

The parameter

IS=<*initial state name*>

can be used as described above to override the initial (TIME=0) values from the file.

The file name "DGTLPSPC" is used with VIEW*sim* A/D to tell the simulator to get digital state values from the VIEW*sim* A/D interface, rather than a file.

Any number of digital input models can be specified, and both file driven and digital simulation driven digital inputs can be used in the same circuit. Different digital input models can reference the same file, or different files. If the models reference the same file, the file must be specified in the same way, or unpredictable results occur. For example, if the default drive is C:, then one model should not have FILE=C:TEST.DAT if another has FILE=TEST.DAT.

For diagnostic purposes, the state of the digital input can be viewed in Probe by specifying B(N*xxx*). The value of B(N*xxx*) is 0.0 if the current state is S0NAME, 1.0 if the current state is S1NAME, and so on through 19.0. B(N*xxx*) cannot be specified on a .PRINT, .PLOT, or .PROBE line. (For digital simulation, the digital window of Probe provides a better way to look at the state of the digital net connected to the digital input.)

Digital Output (O Device)

The digital output device is used to translate analog voltages into digital logic levels (typically “1,” “0,” “X,” “R,” or “F”). The conversion of a voltage into a logic level, models the input stage of a logic device (like a 74LS04) and hence forms a “digital output” from the analog circuit. The logic level information can go to two places: the digital simulation, or a file. (The file can simply be inspected manually, or can be used as a stimulus file for an external logic simulator.)

Note *The following is the General Form for digital simulation*

General Form

O<*name*> <*interface node*> <*reference node*> <*model name*>
+ DGTLNET = <*digital net name*> <*digital I/O model name*>

Note *The following is the General Form for Digital Files*

General Form

O<*name*> <*interface node*> <*reference node*> <*model name*>
+ [SIGNAME = <*digital signal name*>]

Example

```
O12 ANALOG_NODE DIGITAL_GND DO74 DGTLNET=DIGITAL_NODE
IO_STD
OVCO 17   0 TO_TTL
O5   22 100 TO_CMOS SIGNAME=VCO_OUT
```

Model Form

`.MODEL <model name> DOUTPUT [model parameters]`

Table 3-24 *Digital Output Model Parameters*

Model Parameters*	Description	Units	Default
CHGONLY	0: write each timestep, 1: write upon change		0
CLOAD	Output capacitor	farad	0
FILE	Digital input file name (Digital Files only)		
FORMAT	Digital input file format (Digital Files only)		1
RLOAD	Output resistor	ohm	1/GMIN
S0NAME	State “0” character abbreviation		
S0VLO	State “0” low level voltage	volt	
S0VHI	State “0” high level voltage	volt	
S1NAME	State “1” character abbreviation		
S1VLO	State “1” low level voltage	volt	
S1VHI	State “1” high level voltage	volt	
S2NAME	State “2” character abbreviation		
S2VLO	State “2” low level voltage	volt	
S2VHI	State “2” high level voltage	volt	
S19NAME	State “19” character abbreviation		
S19VLO	State “19” low level voltage	volt	
S19VHI	State “19” high level voltage	volt	
SXNAME	State applied when the interface node voltage falls outside all ranges		“?”
TIMESTEP	Digital input file step-size	sec	1E-9
TIMESCALE	Scale factor for TIMESTEP (Digital Files only)		1

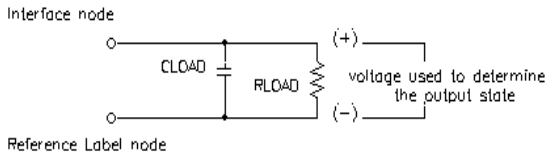
* See .MODEL statement.

The general form for a digital output device, and some of the model parameters, are different for devices which “drive” a file (or VIEW*sim* A/D), and those which “drive” the digital simulation feature. The digital simulation inserts digital output devices *automatically* when a digital device’s input is connected to an analog component. The automatic insertion of digital output devices is discussed in your PSpice user’s guide, and examples of the devices which are inserted can be found in the “dig_io.lib” library file.

Note *For more information on using the digital output device to simulate mixed analog/digital systems, refer to your PSpice user’s guide.*

As shown in Figure 3-3, the digital output device is modeled as a resistor and capacitor, of the values specified in the model statement, connected between <*interface node*> and <*reference node*>. At times which are integer multiples of TIMESTEP, the “state” of the device node is determined and written to the specified file.

Figure 3-3 *Digital Output Model*



The process of converting the input node voltage to a logic state begins by first obtaining the difference in voltage between the <*interface node*> and the <*reference node*>. The DOUTPUT model defines a voltage range, form SxVLO to SxVHI, for each state. If the input voltage is within the range defined for the current state, no state change occurs. Otherwise, the simulator searches forward through the model, starting at the current state, to find the next state whose voltage range contains the input voltage. This state then becomes the new state. When the end of the list (S19) is reached, the simulator wraps around to S0 and continues.

If the entire model has been searched and no valid voltage range has been found, the simulator generates a simulation warning message. Further if the O device is interfacing at the digital simulator, and the SXNAME parameter has not been specified in the model, the simulator uses the state whose voltage range is closed to the input voltage. Otherwise it uses SXNAME as the new state.

This “circular” state searching mechanism allows hysteresis to be modeled directly. The following model statement models the input thresholds of a 7400 series TTL Schmitt-trigger input. Notice that the 0.8 volt overlap between the “0” state voltage range and the “1” state voltage range.

```
.model D074_STd output (
+      s0name=″0″          s0vlo=1.5     s0vhi=1.7
+      s1name=″1″          s1vlo=0.9     s1vhi-7.0
+      )
```

Starting from the “0” state, a positive-going voltage must cross 1.7 volts to get out of the “0” state’s voltage range. The next state which contains that voltage is “1.” Once there, a negative-going voltage must go below 0.9 volts to leave the “1” state’s range. Since no further states are defined, the simulator wraps around back to state “0,” which contains the new voltage

For a digital output driving digital simulation, the parameters

DGTLNET = <*digital net name*> <*digital I/O model name*>

must be specified. Refer to your PSpice user’s guide for more information on digital I/O models. The digital net *must* not be connected to any analog devices, otherwise the automatic analog/digital interface process disconnects the digital output device from the analog net.

For interfacing using digital simulation, the state names must be “0,” “1,” “X,” “R,” “F,” or “Z” (“Z” is usually not used however, since “high impedance” is not a voltage level). Other state names cause the simulator

to stop if they occur; this includes the state "?" which occurs if the voltage is outside all the ranges specified.

The model parameters TIMESCALE, FILE, CHGONLY, and FORMAT are **not** used for digital outputs which drive digital simulation, but the TIMESTEP is used. The TIMESTEP value controls how accurately the analog simulator tries to determine the exact time at which the node voltage crosses a threshold.

To be sure that the transition time is accurately determined, the analog simulator has to evaluate the analog circuit at intervals no larger than TIMESTEP when a transition is about to occur. The default value for TIMESTEP is 1ns, or 1/DIGFREQ (a .OPTIONS option) if it is larger. In many circuits, this is a much greater timing resolution than is required, and some analog simulation time can be saved by increasing the TIMESTEP value.

For digital outputs which write files, or drive VIEW*sim* A/D, the parameter

SIGNAME = <*digital signal name*>

can be used to specify the name written to the file of the digital signal (or for VIEW*sim* A/D, the name of the VIEW*sim* net). If SIGNAME is not specified, then the portion of the device name after the leading O identifies the name of the digital signal.

For digital outputs which write files, the FILE parameter defines the name of the file to be written, and the FORMAT parameter defines the format of the data written to that file.

The file name "PSPCDGTL" is used with VIEW*sim* A/D to tell the simulator to send the digital state values to the VIEW*sim* A/D interface, rather than a file. For VIEW*sim* A/D, the parameters FORMAT and CHGONLY are ignored.

The state of each device is written to the output file at times which are integer multiples of TIMESTEP. The "time" that is written is the integer

time = TIMESCALE·TIME/TIMESTEP

TIMESCALE defaults to 1, but if digital simulation is using a very small timestep compared to the analog simulation timestep, it can speed up the simulation to increase the value of both TIMESTEP and TIMESCALE. This is because the simulator must take timesteps no greater than the digital TIMESTEP size when a digital output is about to change, in order to accurately determine the exact time that the state changes. The value of TIMESTEP should therefore be the time resolution required at the analog-digital interface. The value of TIMESCALE is then used to adjust the output time to be in the same units as digital simulation uses.

For example, if a digital simulation using a timestep of 100 ps is being run, but the circuit has a clock rate of 1us, setting TIMESTEP to 0.1us should provide enough resolution. Setting TIMESCALE to 1000 scales the output time to be in 100 ps units.

If CHGONLY = 1, only those timesteps in which a digital output state changes are written to the file.

Any number of digital output models can be specified, and both file writing and digital simulation driving digital outputs can be used in the same circuit. Different digital output models can reference the same file, or different files. If the models reference the same file, the file must be specified in the same way, or unpredictable results occur. For example, if the default drive is C:, then one model should not have FILE=C:TEST.DAT if another has FILE=TEST.DAT.

For diagnostic purposes, the state of the digital output can be viewed in Probe by specifying B(O*xxx*). The value of B(O*xxx*) is 0.0 if the current state is S0NAME, 1.0 if the current state is S1NAME, and so on through 19.0. B(O*xxx*) cannot be specified on a .PRINT, .PLOT, or .PROBE line. (For digital simulation, the digital window of Probe provides a better way to look at the state of the digital net connected to the digital output.)

Digital Libraries

Table 3-25 lists the library files containing digital devices in the model library:

Table 3-25 *Digital Libraries*

File	Contents
7400.LIB	7400-series TTL
74AC.LIB	Advanced CMOS
74ACT.LIB	TTL-compatible, Advanced CMOS
75ALS.LIB	Advanced Low-Power Schottky TTL
74AS.LIB	Advanced Schottky TTL
74F.LIB	FAST
74H.LIB	High-Speed TTL
74HCT.LIB	TTL-compatible, High-Speed CMOS
74HC.LIB	High-Speed CMOS
74L.LIB	Low-Power TTL
74LS.LIB	Low-Power Schottky TTL
74S.LIB	Schottky TTL
CD4000.LIB	CD4000 devices
DIG_ECL.LIB	10 K and 100K ECL devices
DIG_GAL.LIB	GAL devices
DIG_IO.LIB	I/O models, AtoD and DtoA interface subcircuits, digital power supply subcircuits
DIG_MISC.LIB	pull-up/down resistors, delay line
DIG_PAL.LIB	PAL devices
DIG_PRIM.LIB	Digital primitives
NOM.LIB	master library: which references NOM_DIG.LIB,* which references each of the above libraries.

*Depending upon the platform being worked on, NOM.LIB references the appropriate list of libraries. For “digital only” platforms, NOM.LIB references NOM_DIG.LIB.

7400-Series TTL and CMOS Library Files

The *Library Reference Manual* shows, by part type and technology, each item in the library and gives the order of the pins for that function. This information is needed if a netlist is created manually. Netlists normally are generated automatically by the schematic capture package.

4000-Series CMOS Library

The *Library Reference Manual* shows, by part type and technology, each item in the library and gives the order of the pins for that function. This information is needed if a netlist is created manually. Netlists normally are generated automatically by the schematic capture package.

If power supply nodes on CD4000 devices are not specified in the circuit, they can use the default power supply nodes $G_CD4000_VDD and $G_CD4000_VSS, which default to 5 volts. A new power supply can be created, and new power supply nodes can be specified to the devices in the circuit. Refer to your PSpice user's guide for more information on specifying your own power supplies. Output drives and input thresholds are correctly modeled for power supplies between 3 and 18 volts. Currently, propagation delays do not vary using supply voltages. For correct propagation delays at supply voltages other than 5 volts, the timing models in "cd4000.lib" have to be modified.

Programmable Array Logic Devices

Using a PLD from the library is just like using any other logic device from the library, except that the simulator has to be told the name of the JEDEC file which contains the program for the part. A TEXT parameter name JEDEC_FILE is used to specify the file name, as shown in the following example:

Example

```
X1 IN1 IN2 IN3 IN4 IN5 IN6 IN7 IN8 IN9 IN10 IN11 IN12
+ IN13 IN14
+ OUT1 OUT2 OUT3 OUT4
+ PAL14H4
+ TEXT: JEDEC_FILE = “myprog.jed”
```

This example creates a 14H4 PAL which is programmed by the JEDEC file “myprog.jed.”

Parts

4

Overview

This chapter covers the modeling of off-the-shelf parts. MicroSim Parts allows the conversion of information from the component manufacturer's data sheet (without taking measurements of a real device) into parameter values used by PSpice. Once the parameter values have been obtained, the device can then become part of your own library of devices.

Introduction

One of the difficult areas in using analog circuit simulators is finding accurate models for off-the-shelf parts. The Parts program is a semiautomatic aid for determining the model parameters for standard devices, such as bipolar transistors. It is also used for the subcircuit definitions of more complex models, such as operational amplifiers. Parts allows the conversion of information from the component manufacturer's data sheet (without taking measurements of a real device) into parameter values used by PSpice. Once the parameter values have been obtained, the device can be included as part of your own library of devices.

The question can be asked, "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 the part parameter range that is guaranteed by the manufacturer. The device's operating characteristics 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 show how most of the devices can operate. Therefore, while Parts can work using measurements taken from a specific device, *this is not necessary*. Most of the simulation work would probably use typical values from the data sheets. The best/worst-case models can also be used for checking the design.

User Interface

Toolbar Buttons

Toolbar buttons provide shortcuts for initiating common functions. Choosing a toolbar button (by a click), starts the same response as would selecting the corresponding menu item. Below is a list of the toolbar buttons and the equivalent menu item or action taken in Parts.

Table 4-1 *Toolbar buttons*

Button	Equivalent Menu Item or Action
	File/Open
	File/Save Library
	Part/New
	Part/Get
	Plot/Display
	Immediate hard copy print
	View/In
	View/Out
	View/Area
	View/Fit
	Plot/X Axis Settings/Scale: Linear or Log
	Plot/Y Axis Settings/Scale: Linear or Log
	Extract/Parameter

Keyboard Accelerators

A keyboard accelerator is a key, or combination of keys, that when pressed is equivalent to selecting a menu or dialog box item. Parts provides a default set of keyboard accelerators. These various keystrokes can be used as “shortcuts” to menu/command items. The default accelerator keys and the menu/command item that is activated when accelerators are used are shown next to the menu item in the view window.

Windows Menu Functions

File Menu

Menu Option	Description
Open/Create Library	Allows the opening of a model library file. A new library file can be created using this command or the opening of an existing one. If a model library file name is written in the Open Library dialog box and the file does not exist, a new library file is created using the name that was entered.
Save Library	Saves the existing data and model parameters to the opened library.
Save As Library	Allows the current data and model parameters to be saved to a new file, or overwrites an existing library file.
Print	Allows a copy of any or all of the plot windows wanted to be printed. When Print is selected, the Print dialog box appears having a list box displaying a list of all plot windows that are currently open. At least one plot window must be selected, but all can be selected if wanted. The Select All button can be used to quickly select all of the plot windows for print.
Page Setup	Sets the print related options. The controls in this dialog box allow a variety of settings to be changed that alters the way the printed plots appear on the hard copy page. The options available are dependent upon the printer selected (see **Printer Select** below).
Printer Select	Allows changing the current default printer. To select a printer other than the current default, use the Specific Printer pull-down.
Log Commands	Allows commanding logging on and off. Turning on command logging allows a file to be created for the commands that are wanted. This file can be used to play back those commands by using Run Commands. When Log Commands is first selected, a file name is entered which stores the commands. From this point on, all actions taken in Parts are stored in this log file. In order to show that command logging is in progress, the Log Commands selection is marked using a check box. Command logging can be turned off by selecting Log Commands again, which also removes the marker in the check box.
Run Commands	Allows a command file created using Log Commands to be played back. A file from a dialog box can be chosen showing the available command files. The Run Command File dialog box appears allowing the selected command file to run.
Exit	Exits Parts and returns control to the Windows environment.
1-4	A shortcut for opening one of the four most recently used library files. The file name listed by 1 is for the most recently used file, and the file name by 4 is for the fourth most recently used file.

Edit Menu

Menu Option	Description
Cut	Allows a selected trace to be deleted. Click on a trace name in the device curve plot window to select it, then select Cut to remove the trace from the plot.
Parameter	Allows editing the internal parameters of the model definition. First select a model parameter from the Parameters list box found on the right-hand side of the main window. Once the item is selected choose Edit/Parameter or double-click on the highlighted item to bring up the Edit Parameter dialog box. Edit the values in the fields provided, then click OK to save the modifications. Click Cancel to disregard all actions. To edit parameters specify an upper or lower boundary from which parameters can be extracted. Click in the Freeze current value in the extraction check box to show that the current value remains as specified during parameter extraction. For example, if the current value for EG (activation energy) is 1.110 and the Freeze current value check box is clicked, then the EG value of 1.110 is maintained during parameter extraction.
Spec	Allows modification of the model specifications for the model template chosen. For example, one of the model parameters for the diode template is Forward Current. To edit the specification for forward current, select Forward Current in the left list box, then click select Edit/Spec. Or, double-click on Forward Current to bring up the Edit Model Spec dialog box. Once in the dialog box, Add, Modify, and Delete specification data as wanted. Click OK to save the changes, and click Cancel to disregard all actions.

Part Menu

Menu Option	Description
New	Allows a creation of a new device model. A library must be open in order to create a new part. The Create New Part dialog box is prompting for a part name and a part type. Enter the name in the Part Name field and choose a part type from the Part Type field. Scroll through the list box if needed to select an available template. Click OK to end the dialog box and begin entering the model specifications and parameters. The default parameters appear for the template chosen. These default parameters are created by Parts. Parts parameter defaults are different than PSpice default parameters. See **Chapter 2,Analog Devices** for the default PSpice parameters.
Copy	Copies an existing part definition to a new part name in a user-selected library. Enter a unique name for the new part in the New Part Name field. A part can be selected from the opened library by scrolling through the Part list box and clicking on a part name. Clicking on a part name sends the part name to the Existing Part Name field. Click OK to copy the part definition to the new part name. The opened library name is displayed above the Select Lib button. If no library is currently open, click Select Lib to browse and choose a library file to be opened. (See Select Lib below.)
Select Lib	Allows browsing through directories to select a library file. Select a library file which contains the existing part for copying. Click OK after choosing the library file name. Click Cancel to return to the Copy Part dialog box.
Get	Retrieves a part from a model library file. A library file must be opened (File/Open) in order to get a part. Scroll through the list box to select a part, then click OK. Once obtained, this part becomes the current part named in the title bar.
Save	Saves the current part into the opened library. The current part name is displayed in the title bar at the top of the window. All changes made up to this point were not saved in the library. Use File/Save to save the data on disk.
IBIS translator	Translates a model definition in IBIS model format to a PSpice model definition. The IBIS model must be saved in a ".ibs" file. Once a file is selected to be translated in the IBIS Translator dialog box, click OK. A PSpice model file in a ".mod" format is created in the same working directory.
Export	Writes the part definitions from the current part to an ASCII text file and gives the exported file a ".mod" extension if no extension is specified. This command allows transferring model definitions from one library to another.
Import	Imports part definitions from ".mod" files into the current model library.

Trace Menu

Menu Option	Description
Add *	Adds traces to an existing plot. Use Plot/Display to plot a curve. The default trace variable in the device curve window is temperature. A temperature can be specified in the Add Traces dialog box where the trace displayed is wanted. For example, the reverse drain current at 10 degrees is wanted for display. In this case enter 10 in the Temperature field of the Add Traces dialog box, then click OK. The trace displayed would be named Idr(10°ëc). **CAUTION:** Each curve in Parts is defined only by the parameters being adjusted. The forward current curve displayed in Parts only shows the part of the current equation which is associated with the forward characteristic parameters (e.g., IS, N, Rs). However, PSpice uses the full equation, which includes a term involving the reverse characteristic parameters (e.g., ISR, NR), which could have a significant effect at low current. This means that the curve displayed in Parts is not exactly like what is displayed in Probe after a simulation. Models should always be tested and verified using PSpice and fine-tuned if necessary.

* Some traces cannot have temperature effects. Therefore changing the temperature value cannot alter the trace. To change the trace variable, select Plot/X Axis Settings/Trace Variable.

Plot Menu

Menu Option	Description
Display	Displays one or more traces for the selected model parameters. To display a trace, select a parameter from the Model Spec list box in the current device window, then chose Plot/Display. To select more than one traces for display, click on the first model parameter, then <Shift>-click on one or more model parameters. Select Display from the Plot menu to view the traces. Each trace can generate its own plot window.
X Axis Settings	Modifies the X axis settings for the currently selected plot window.
Data Range	This sets the range of the X axis to the specification wanted. Once set, the range is not affected by changes in the axis' variable/traces. It can only be changed by one of the view commands or another user defined range. To enter a data range, enter the beginning value of the range in the first field, then enter the end value in the second field. By specifying the correct units, assures that the units can appear on the numbers labeling the axis' tick marks. This command can be used to invert an axis. For instance, instead of having a range of (0V to 5V), the range can be set to (5V to 0V). All the traces on that axis are then inverted.

Menu Option	Description
Scale	
Linear	Sets the current X axis to linear.
Log	Sets the current X axis to logarithmic. The axis cannot be put into a log scale if either end of the axis range is zero or negative.
Trace Variable	Allows the trace variable for the X axis to be modified when using Trace/Add. Click on the Trace Variable button, then select a variable from the list box. Click OK to accept the changes, or click Cancel to disregard. The X axis can show which trace variable is being used.
Y Axis Settings	Allows the modification of the Y axis settings.

Menu Option	Description
Data Range	
Auto Range	Sets the range of the Y axis to be the range of its variable/traces, rounded to a convenient value. The range of the axis is automatically adjusted as its variable/traces are changed.
User Defined	Sets the range of the Y axis to the specification wanted. Once set, the range is not affected by changes in the axis' variable/traces. It can only be changed by one of the view commands or another **User Defined** command. To enter a user-defined data range, enter the beginning value of the range in the first field, then enter the end value in the second field. Specifying the correct units, assures that the units appear on the numbers labeling the axis' tick marks.
Scale	
Linear	Sets the current Y axis to linear.
Log	Sets the current Y axis to logarithmic. The axis cannot be put into a log scale if either end of the axis range is zero or negative.

View Menu

Menu Option	Description
Fit	Sets the display back to the default view factor, and refreshes the display.
In	Zooms in by a factor of two around the point specified using the mouse.
Out	Zooms out by a factor of two around the point specified using the mouse. The View/Fit limits cannot be exceeded. If the View/Fit limits are exceeded in either the X or Y direction, then only the direction in which they are not exceeded can zoom out by the full factor of two. The exceeded direction can only zoom out to the View/Fit limits.
Area	Causes the display to view into the area specified using the mouse. This area is specified by holding down the left mouse button and dragging the mouse. This forms a window into which to view. The area can be specified before or after the View/Area operation is selected.
Previous	There is one view stack per plot. View changes for all Y axes for a plot go into the view stack for that plot.
Redraw	Causes the active plot window to be redrawn.
Pan - New Center	Changes the center of the plot view without changing the scale of the plot. To do this, select Pan - New Center, and click the mouse at the point on the plot that is wanted for the center of the new view.

Extract Menu

Menu Option	Description
Parameters	Starts the parameter extraction process. Model parameters are modified according to the specification data. After parameter extraction is done, all plots can be updated to use the new model parameters.

Options Menu

Menu Option	Description
Toolbar	Allows the toolbar to be turned on and off. A check box in the menu shows that the toolbar is turned on.

Window Menu

Menu Option	Description
Close	Closes the current window.
Arrange	Arranges the plot windows by either cascading or tiling.
1-9	Brings the selected plot window to the foreground.

Help Menu

Menu Option	Description
Index	Lists all of the topics available in the Help menu.
Keyboard	Lists the special keys on the keyboard and their function.
Menu Commands	Lists the various menu commands and their function.
Procedures	Describes how to use the menu commands to perform various tasks.
Using Help	Describes how to use the Help menu.
About	Displays the MicroSim copyright notice, the software version, the serial number, and user identification information. Verifying the software version and customer ID number and have it ready before contacting MicroSim for technical support.

Sun and HP Menu Functions

Main Menu

Menu Option *	Description
Exit Program	EXITS Parts and returns control to DOS.
Diode (signal/rectifier/Zener)	Enters the Diode menu set.
Device part number (or name)	After the part name is entered, control passes to the Parameter Modification menu. The diode model parameters are defined as discussed in **Chapter 2,Analog Devices**.
Bipolar Transistor (general purpose)	Enters the Bipolar Transistor menu set.
Device part number (or name)	Enters the part name of the device being generated. After the part name is entered, control passes to the Parameter Modification menu. The bipolar transistor model parameters are defined as discussed in **Chapter 2,Analog Devices**.
JFET (small-signal, general purpose)	Enters the JFET menu set. The first screen prompts for the following information: N-channel P-channel Select: After the selection has been made, control passes to the Parameter Modification menu. The junction field-effect transistor model parameters are defined as discussed in **Chapter 2,Analog Devices**.
Power MOSFET Transistor (all types)	Enters the Power MOSFET Transistor menu set. The first screen prompts for the following information: N-channel P-channel Select: After the selection has been made, control passes to the Parameter Modification menu. The MOSFET transistor model parameters are defined as discussed in **Chapter 2,Analog Devices**.

Menu Option *	Description
Operational Amplifier (bipolar/FET)	Enters the Operational Amplifier menu set. The first screen prompts for the following information: Bipolar, NPN input (example: uA741) Bipolar, PNP input (example: LM324) JFET, N-channel input (example: LH032) JFET, P-channel input (example: LF355) Select: After the selection has been made, the following screen is displayed: Internally compensated (example: uA741) Externally compensated (example: uA748) Select: After the selection has been made, control passes to the Parameter Modification menu.
Voltage Comparator (bipolar OC)	Enters the Voltage Comparator menu set. The first screen prompts for the following information: Bipolar, NPN input (example: LM319) Bipolar, PNP input (example: LM339) Select: After the selection has been made, the following screen is displayed: Output grounded to -V supply (example: LM339) Output has separate ground (example: LM319) Select: After the selection has been made, control passes to the Parameter Modification menu. The voltage comparator model parameters are defined as discussed in **Chapter 2,Analog Devices**.
Nonlin. Magnetic Core (ferrite/MPP)	Enters the Nonlinear Magnetic Core menu set. The first screen prompts for the following information:
Device part number (or name)	After the part name is entered, control passes to the Parameter Modification menu. The nonlinear magnetic core model parameters are defined as discussed in **Chapter 2,Analog Devices**.
Voltage Regulator (positive/adjustable)	Enters the Voltage Regulator menu set. The first screen prompts for the following information:

Menu Option *	Description
Device part number (or name)	After the part name is entered, control passes to the Parameter Modification menu. The nonlinear magnetic core model parameters are defined as discussed in **Chapter 2,Analog Devices**.
Voltage Reference (2-terminal)	Enters the Reference Voltage menu set. The first screen prompts for the following information:
Device part number (or name)	Allows entering the part name of the device being generated.

* The Main menu appears when Parts is executed and allows selection of the device type of the model wanted for generation. After the device is selected, a series of screens are displayed allowing the entry and modification of device curve data and model parameters

Parameter Modification Menu

Menu Option*	Description
Exit	Returns to the Main menu.
Next_set	Advances to the next set of model parameters for verification or modification.
Screen_info	Enters the Screen Info menu and displays information about the current parameter set.
Exit	Exits the Screen Info menu and returns to the Parameter Modification menu.
Down_page	Advances to the next page of parameter information.
Up_page	Returns to the previous page of parameter information.
Device_curve	Enters the Device Curve menu which allows the modification of the device curve parameters.
Exit	Exits the Device Curve menu and returns to the Parameter Modification menu.
Add	Allows the entry of additional device curve data.
Change	Allows the modification of device curve data.
Delete	Allows the elimination of device curve data.
Model_parameters	Enters the Model Parameters menu which allows the modification of the model parameters.
Fit	Performs an iterative set of calculations using an optimizer to get the best fit curve reducing the amount of error.

Menu Option*	Description
Trace	Enters the Trace menu allowing a display of an additional trace for the current set of parameters, given a change in the selected variable.
Exit	Exits the Trace menu and returns to the Parameter Modification menu.
Add_trace	Allows a display of an additional trace for the given set of parameters, using the current Trace_variable.
Delete_trace	Allows removing a specified trace from the display.
Trace_variable	Allows selection of a different Trace_variable.
X_axis	Enters the X Axis menu allowing modifications of the scale and range.
Exit	Exits the X Axis menu and returns to the Parameter Modification menu.
Log/Linear	Changes the scale of the X axis to either log or linear.
Set_range	Allows setting the range for the X axis.
Y_axis	Enters the Y Axis menu allows modifying the scale and range.
Exit	Exits the Y Axis menu and returns to the Parameter Modification menu.
Log/Linear	Changes the scale of the Y axis to either log or linear.
Set_range	Allows setting the range of the Y axis.
Hard_copy	Enters the Hard Copy menu allows printing a hard copy of the current parameter information and plot.
Exit	Exits the Hard Copy menu and returns to the Parameter Modification menu.
1_page_long	Prints the current parameter information and plot on a single sheet of paper.
2_pages_long	Prints the current parameter information and plot on two sheets of paper.
Other_length	Allows specifying in inches the wanted length of the printed plot.

* The Parameter Modification menu allows the verification and modification of device curve data and model parameters.

Parameter Definitions

The following sections describe in detail, each set of device curve and model parameters for the different devices.

Note *Each curve in Windows Parts is defined only by the parameters it adjusts. So, the forward current curve displayed in Parts only shows the part of the current equation which is associated using the forward characteristic parameters (e.g., IS, N, Rs). However, PSpice uses the full equation, which includes a term involving the reverse characteristic parameters (e.g., ISR, NR), which could have a significant effect at low current. This means that the curve displayed in Parts 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.*

Supported Models

The supported model types for the Parts program are as follows:

- Bipolar Transistor Model
- Diode Model
- Insulated Gate Bipolar Transistor (IGBT)
- Junction Field-Effect Transistor (JFET) Model
- Nonlinear Magnetic Core Model
- Operational Amplifier Model
- Power MOSFET Model
- Voltage Comparator Model
- Voltage Regulator Model
- Voltage Reference Model (2-terminal)

Diode Model

The diode element is a superset of the model in U.C. Berkeley SPICE, which features:

- a current generator to model the ideal diode law (Shockley equation) and reverse breakdown,
- a variable capacitance to account for the semiconductor junction charge storage, and
- a series resistance to account for the ohmic resistance.
- The PSpice enhancements include:
- enhanced temperature variation modeling,
- the high-level injection effects of a real diode, and
- a more realistic modeling of reverse leakage characteristics.

See the diode device in the **Chapter 2,Analog Devices** for the internal model used by the simulator.

Table 4-2 *Diode Model Default Parameters*

Parameter	Default
BV	100
CJO	1e-12
EG	1.11
FC	.5
IBV	1e-4
IKF	0
IS	1e-14
ISR	1e-10
M	.333
N	1
NR	2
RS	1e-3
TT	5e-9
VJ	.75
XTI	3

Diode - Forward Current

Device curve:

Vfwd	forward voltage across junction for Ifwd
Ifwd	forward current @ Vfwd

Model parameters:

IS	saturation current
N	emission coefficient
RS	series resistance
IKF	high-injection “knee” current
XTI	IS temperature coefficient
EG	activation energy

This screen estimates the parameters IS and RS from three voltage and current values. Try to include data from low current values (where the increase in current is exponential), moderate current values, and high current value (where the increase in current is clearly resistive).

The last two model parameters, XTI and EG, can be changed. We have set them to be normal values for silicon diodes. For Schottky-barrier diodes these can be changed to XTI = 2 and EG = 0.69, which gives better modeling over temperature.

Also, it is sometimes helpful to set up traces for a few values of temperature (use the **Trace/Add** command) for adjusting XTI.

Diode - Junction Capacitance

Device curve:

Vrev	reverse voltage across diode (junction) for Cj
Cj	junction capacitance @ Vrev

Model parameters:

CJO	zero-bias junction capacitance
VJ	junction potential
M	junction grading coefficient
Fc	coefficient for onset of forward-bias depletion capacitance

This screen estimates the parameters CJO and M from a capacitance values given at nonzero reverse biases (a zero value for a Vj data point is OK).

The value for FC has been set to be normal for silicon diodes, but is relatively unimportant, as forward capacitance is dominated by diffusion capacitance (and modeled by transit time).

The data sheets for most switching and power diodes have little detail about reverse bias capacitance, because it is not too important. Varicap diodes usually have better, more complete information. Be aware that the diode package adds some fixed amount of capacitance that is not included in the device model, but can be included by the using a small capacitor across the diode. Having determined the package capacitance, subtract that from the total capacitance to model the diode junction.

Diode - Reverse Leakage

Device curve:

Vrev	reverse voltage for Irev
Irev	reverse (leakage) current @ Vrev

Model parameters:

ISR	recombination current saturation value
NR	recombination current emission coefficient

This screen derives the generation-recombination current values for the device which, using the capacitance modeling (previous screen), provides the primary leakage mechanism of the diode junction.

Reverse current leakage is increased by imperfections in manufacturing which are not modeled. Breakdown also increases reverse current, but this is modeled in the next screen.

Diode - Reverse Breakdown

Device data:

Vz	nominal Zener voltage @ Iz
Iz	nominal Zener current for Vz
Zz	Zener impedance (resistance) @ Vz,Iz

Model parameters:

BV	reverse breakdown voltage (a positive value)
IBV	reverse breakdown current (a positive value)

This screen estimates the parameters BV and IBV for reverse breakdown operation, which is how voltage regulator (Zener or avalanche) diodes work. Enter the values for Vz, Iz, and Zz.

BV and IBV nearly equals Vz and Iz. As the breakdown effect is modeled by an exponential function, the value of BV and IBV can adjust so that device impedance, Zz (ratio of the change in voltage to the change in current) is correct at Vz,Iz.

Diode - Reverse Recovery

Device data:

Trr	reverse recovery time
Ifwd	forward current (before switching)
Irev	initial reverse current
Rl	load resistance (total load of test fixture)

Model parameters:

TT	transit time

This screen estimates the parameter TT from switching time. Enter values for the upper list box. Be sure to include the test fixture resistance and pulse generator resistance in Rl.

The screen does a transient simulation of the diode switching. Some of the parameters from earlier screens that have dynamic effects, for example, CJO, are included in the simulation. Adjust the X axis as required to see the entire waveform.

Bipolar Transistor Model

The bipolar transistor element is an extended Gummel-Poon model. It is a superset of the Gummel-Poon model, which in turn is a superset of the Ebers-Moll model: by allowing certain parameters to default to ideal values, this model covers the full range from very simple to very sophisticated. The Gummel-Poon model includes: (i) complete DC characterization, (ii) charge-storage effects, (iii) basewidth modulation, and (iv) temperature variations. The extensions include: (v) variable base resistance, (vi) collector-base capacitance splitting, and (vii) variable forward transit time.

For a more complete description of bipolar transistor models and their derivation, refer to:

> Ian Getreu, *Modeling the Bipolar Transistor*, Tektronix, Inc., part # 062-2841-00

For more information about the bipolar transistor device for the internal model used by PSpice, see **Chapter 2,Analog Devices**.

Table 4-3 *Bipolar Transistor Model Default Parameters*

Parameter	Default
BF	100
BR	1
CJC	0
CJE	0
EG	1.11
FC	0.5
IKF	0
IKR	0
IS	1e-14
ISC	0
ISE	0
ITF	1
MJC	0.33
MJE	0.33
NC	2

Parameter	Default
NE	1.5
NF	1
NK	0.5
NR	1
PTF	0
RB	0
RC	0
RE	0
TF	10e-9
TR	10e-9
VAF	100
VAR	100
VJE	0.75
VJC	0.75
VTF	10
XTB	0
XTI	3
XTF	10

Bipolar Transistor - Junction Voltage

Device data:

Vbe	base-emitter voltage @ Ib (device in saturation)
Vce	collector-emitter voltage (device in saturation)
Ib	base current for Vbe and Vce
%Ib	fraction of Ib (not a data sheet value)

Model parameters:

IS	saturation current
XTI	temperature coefficient for IS
EG	activation energy

This screen estimates the parameter IS from the saturation characteristics of the transistor. IS is a semiconductor junction parameter and should not be confused with the collector current in saturation. The data sheet has values or curves for Vbe and Vce in a “forced beta” (where the ratio Ic/Ib is much lower than the normal current gain) or “saturated” condition. Enter values of Vbe and Vce for the same Ib.

The value of %Ib is a “fudge” value and is not critical. It factors how much of the base current is shunted through the ideal diode of the Gummel-Poon transistor model. We have set it to a “normal” amount.

Obtaining an accurate value for IS is not critical, since other parameters are set relative to IS, and only the ratio between values are important. It is necessary, though, to not have a wildly inaccurate value. The last two model parameters, XTI and EG, can be changed. We have set them to be normal values for silicon transistors.

The display graphs for this screen are not too useful. However, they do tell when something is happening.

Bipolar Transistor - Output Admittance

Device curve:

Ic	collector current for hoe
hoe	small-signal open-circuit output admittance @Ic (and @Vce)

Conditions:

Vce	collector-emitter voltage for the device curve

Model parameters:

VAF	forward Early voltage

This screen estimates the parameter VAF, which sets the output conductance of the transistor in a common emitter configuration.

The parameter VAF controls one aspect of basewidth modulation in the Gummel-Poon transistor model. This manifests itself as output conductance. Typical values are 50-to-100 volts for normal transistors and 1-to-10 volts for super-beta transistors.

Bipolar Transistor - Forward DC Beta

Device curve:

Ic	collector current for hFE (@ Vce)
hFE	forward DC beta @ Ic

Conditions:

Vce	collector-emitter voltage for the device curve

Model parameters:

BF	ideal maximum forward beta
ISE	non-ideal base-emitter diode saturation current
NE	non-ideal base-emitter diode emission coefficient
IKF	forward beta roll-off “knee” current
NK	forward beta roll-off slope exponent
XTB	forward beta temperature coefficient

This screen estimates parameters for the celebrated Gummel-Poon bipolar transistor model. Try to include data from low current values (beta rising), moderate current values, and high current value (beta falling). The value Vce adjusts the beta data for basewidth modulation effects.

Transistor data sheets usually show minimum beta values and have a maximum value for only one collector current value. One way to obtain an average value is to use the current level that specifies both minimum and maximum beta, using a value somewhat below the average of the minimum and maximum. Then ratio the other minimum values by the same amount. Or just use the curves (if available) from the data sheet.

The value for XTB has been set to be normal for bipolar transistors but can be changed. It is sometimes helpful to set up traces for a few values of temperature (use the **Trace/Add** command) for adjusting XTB.

Bipolar Transistor - Vce(sat) Voltage

Device curve:

Ic	collector current for Vce (@ Ib/Ic)
Vce	collector-emitter voltage @ Ic

Conditions:

Ic/Ib	“forced beta” ratio for device curve

Model parameters:

BR	ideal maximum reverse beta
ISC	non-ideal base-collector diode saturation current
NC	non-ideal base-collector diode emission coefficient
IKR	reverse beta roll-off “knee” current
RC	series collector resistance

This screen estimates more parameters for the Gummel-Poon transistor model. Try to include data from low current values (Vce falling), moderate current values, and high current value (Vce rising). Also, be sure to verify and enter the value for the “forced beta” ratio of collector to base current.

The reverse Gummel-Poon parameters correspond to the forward parameters, except they are for reverse operation (that is, emitter swapped with the collector). It would be more accurate to obtain these the same way as the forward parameters, but reverse operation is rarely published data. Fortunately, it does not affect operation when the transistor is saturated, that is, when the base-collector junction is forward biased.

Bipolar Transistor - C-B Capacitance

Device curve:

Vcb	reverse voltage collector-base junction for Cobo
Cobo	open circuit output capacitance @ Vcb

Model parameters:

CJC	0-bias collector-base junction capacitance
VJC	collector-base junction potential
MJC	collector-base junction grading coefficient
FC	coefficient for onset of forward-bias depletion capacitance

This screen estimates the parameters CJC and MJC from capacitance values given at nonzero reverse biases (a zero value for Vcb is OK).

The value for FC has been set to be normal for silicon transistors but can be changed. The value of FC is relatively unimportant, as forward capacitance is dominated by diffusion capacitance (and modeled by transit time).

Be aware that the transistor package adds some fixed amount of capacitance that is not included in the device model, but can be included by using a small capacitor across the junction. Having determined the package capacitance, subtract that from the total capacitance to model the junction.

Bipolar Transistor - E-B Capacitance

Device curve:

Veb	reverse voltage emitter-base junction for Cibo
Cibo	open circuit input capacitance @ Veb

Model parameters:

CJE	zero-bias emitter-base junction capacitance
VJE	emitter-base junction potential
MJE	emitter-base junction grading coefficient

This screen estimates the parameters CJE and MJE from capacitance values given at nonzero reverse biases (a zero value for Veb1 is OK).

The value of FC from the previous screen is used and is still relatively unimportant, as forward capacitance is dominated by diffusion capacitance.

Be aware that the transistor package adds some fixed amount of capacitance that is not included in the device model, but can be included by using a small capacitor across the junction. Having determined the package capacitance, subtract that from the total capacitance to model the junction.

Bipolar Transistor - Storage Time

Device curve:

Ic	collector current for ts (@ Ic/Ib)
ts	storage time @ Ic

Conditions:

Ic/Ib	“forced beta” ratio for device curve

Model parameters:

TR	reverse transit time

This screen estimates the parameter TR, which controls the delay until the transistor leaves saturation when switching off. Be sure to verify and enter a value for the “forced beta” ratio when the transistor was on and saturated.

The storage time curve is controlled by the forward and reverse beta characteristics of the transistor. The parameter TR acts like a multiplying factor without changing the character of the curve. Use the storage time for the collector current range of interest.

Bipolar Transistor - Gain Bandwidth

Device curve:

Ic	collector current for fT (@ Vce)
fT	frequency at which small-signal forward current transfer ratio extrapolates to unity @ Ic

Conditions:

Vce	collector-emitter voltage for device curve

Model parameters:

TF	forward transit time
ITF	current for TF dependency on Ic
XTF	coefficient for TF dependency on Vce
VTF	voltage for TF dependency on Vce

This screen estimates the parameter TF, which, along with collector-base capacitance, limits high-frequency gain. The value of TF also controls rise/fall times in switching circuits, which is another way to measure transistor speed, though we haven't thought of a rule-of-thumb conversion between rise/fall time and high-frequency cutoff.

Also, it is sometimes helpful to set up traces for a few values of Vce (use the **Trace/Add** command) for adjusting VTF.

Insulated Gate Bipolar Transistor (IGBT) Model

Table 4-4 *IGBT Model Parameters and Default Values*

Parameter	Default
AGD	4.800E-6
AREA	12.50E-6
BVF	1
BVN	4
CGS	18.57E-9
COXD	43.30E-9
JSNE	650.0E-15
KF	0.5005
KP	2.170
MUN	1.500E3
MUP	450
NB	200.0E12
TAU	447.4E-9
THETA	20.00E-3
VT	3.497
VTD	-5
WB	117.0E-6

IGBT - Fall Time

Device data:

Icmax	Absolute maximum continuous collector current, in amps, at 25C
Bvces	Absolute maximum collector-emitter breakdown voltage, in volts, with gate-emitter shorted
tf	Collector current fall time, in seconds, with inductive load at the given Ic and Vce
Ic	Collector current, in amps, at which tf is measured
Vce	Collector-emitter voltage, in volts, at which tf is measured

Model parameters:

AREA	Device active area, in square meters
TAU	Base lifetime, in seconds
WB	Metallurgical base width, in meters

This screen shows the fall time of the collector current measured with inductive load at turn off. The initial collector current is modeled by Ic. At turn off, this current falls rapidly, followed by a slow decaying tail. The rate of decay is controlled by the recombination rate of excess carriers in the lightly-doped epitaxial base layer. This recombination rate, in turn, is described by the base lifetime parameter TAU.

The maximum collector current, Icmax, and the maximum collector-emitter breakdown voltage, BVces, are obtained in data sheets in the absolute maximum ratings table.

IGBT - Transfer Characteristics

Device data:

Vge	Gate-emitter voltage, in volts, at 25C at which the transfer characteristics is measured
Ic	Collector current at the given Vge, in amps, at 25C at which the transfer characteristics is measured
Vce	Collector-emitter voltage, in volts, at which Vge and Ic are measured

Model parameters:

KP	MOSFET transconductance, in amps/(square volt)
VT	Internal MOSFET channel threshold voltage, in volts

This screens displays the transfer characteristics at nominal temperature as the gate-emitter voltage increases from zero volt.

Data sheets usually provide the transfer characteristics curve. Points (Vge, Ic) should be sampled along the entire region of the curve. Care should be taken when sampling points near the threshold region as they will affect the accuracy of the parameter VT.

IGBT - Saturation Characteristics

Device data:

Vce — Collector-emitter voltage at the given Ic, in volts, at 25C at which the saturation characteristics is measured

Ic — Collector current at the given Vge, in amps, at which the saturation characteristics is measure

Vge — Gate-emitter voltage, in volts, at which the saturation characteristics is measured

Model parameters:

KF — MOSFET linear region transconductance, in amps/(square volt)

This screen shows the saturation characteristics at nominal temperature as the collector current increases from zero amp.

Data sheets usually provide the saturation characteristics curve. Points should be sampled along the entire region of the curve.

IGBT - Gate Charge

Device data:

Qge	Gate-emitter charge at turn-on at the given Vcc and Ic, in coulombs
Qgc	Gate-collector charge at turn-on at the given Vcc and Ic, in coulombs
Qg	Total gate charge at turn-on at the given Vg, Vcc, and Ic, in coulombs
Vg	Gate voltage at which Qg is measured, in volts
Vcc	Collector voltage at which Qge, Qgc, and Qg are measured, in volts
Ic	Collector current at which Qge, Qgc, and Qg are measured, in amps

Model parameters:

CGS	Internal MOSFET gate-source capacitance per unit area, in farads/(square cm)
COXD	Internal MOSFET gate-drain overlap oxide capacitance per unit area, in farads/(square cm)
AGD	Internal MOSFET gate-drain area, in square centimeters

This screen displays the gate charge characteristics at turn-on at the given Vcc and Ic. It shows the gate-emitter voltage, Vge, as a function of gate charge. Usually, the gate charge curve is divided into three distinct regions.

The first region shows Vge rising at a constant rate until the collector current reaches Ic as a constant gate current is charging the constant gate-emitter capacitance CGS. The total charge supplied to the gate in this region is Qge. This parameter is obtained in data sheets either in the electrical characteristics table or from the gate-charge curve.

In the second region, Vge is nearly constant as the gate current discharges the internal MOSFET gate-drain capacitance. The charge supplied in this region is Qgc. Like Qge, it is obtained in data sheets either in the electrical characteristics table or from the gate-charge curve.

Vge increases at a constant rate again in the third region as the device is now operating in the linear region. The gate current charges both CGS and the internal MOSFET gate-drain overlap oxide capacitance COXD. Qg and Vg represent a point along the curve in this region. They are obtained either in the electrical characteristics table or from the gate-charge curve. Note that Qg must be greater than the sum of Qge and Qgc. Furthermore, Vg must be greater than the gate-emitter plateau voltage Vge in the second region.

Junction Field-Effect Transistor (JFET) Model

The JFET element is a superset of the U.C. Berkeley SPICE model, which is a square-law device featuring:

- complete DC characterization, and
- capacitive effects.

The PSpice enhancements include:

- enhanced temperature variation modeling, and
- a more realistic modeling of gate leakage currents, both passive and active.

Also, see **Chapter 2,Analog Devices** on the JFET device for the internal model used by PSpice.

Table 4-5 *Junction-Field-Effect Transistor Model Default Parameters*

Parameter	Default
AF	1
ALPHA	1e-6
BETA	1e-4
BETATCE	-0.5
CGD	1e-12
CGS	1e-12
FC	0.5
IS	1e-14
ISR	0
KF	1e-18
LAMDA	1e-6
M	0.5
N	1
PB	1
RD	1
RS	1
VK	1

Parameter	Default
VTO	-2
VTOTC	-2.5e-3
XTI	3

JFET - Transconductance

Device curve:

Id	drain current for gFS
gFS	forward transconductance @ Id

Model parameters:

BETA	transconductance coefficient
RD	drain resistance
RS	source resistance
BETATCE	temperature coefficient for BETA

This screen estimates the parameter BETA, which sets the change in drain current vs. gate-source voltage. BETATCE is set manually, using traces at other temperatures to judge the effect (the default setting is a nominal value chosen from inspecting many data sheets).

Also, it is sometimes helpful to set up traces for a few values of temperature (use the **Trace/Add** command) for adjusting BETATCE.

JFET - Output Conductance

Device curve:

Id	drain current for gOS
gOS	output conductance @ Id

Model parameters:

LAMBDA	channel-length modulation

This screen estimates the parameter LAMBDA, which sets the slope of the drain current vs. drain-source voltage in saturation.

JFET - Transfer Curve

Device curve:

Vgs	gate-source voltage for Id (@ Vds)
Id	drain current @ Vgs

Conditions:

Vds	drain-source voltage for device curve

Model parameters:

VTO	threshold voltage
VTOTC	temperature coefficient for VTO

This screen estimates the parameter VTO, which is the threshold ("pinchoff") voltage. VTOTC is set manually, using traces at other temperatures to judge the effect (the default setting is a nominal value chosen from inspecting many data sheets).

Note: The SPICE "standard" is for VTO to be a negative value for a depletion transistor, regardless of device type (NJF or PJF).

JFET - Reverse Transfer Capacitance

Device curve:

Vgs	gate-source voltage for Crss (@ Vds)
Crss	reverse transfer capacitance @ Vgs

Conditions:

Vds	drain-source voltage for device curve

Model parameters:

CGD	zero-bias gate-drain capacitance
M	junction grading factor
PB	built-in potential
FC	forward-bias coefficient

This screen estimates the parameters CGD and M. The reverse transfer, or "Miller," capacitance is modeled.

The parameter FC applies to forward-biased junctions and is included for completeness.

JFET - Input Capacitance

Device curve:

Vgs	gate-source voltage for Ciss (@ Vds)
Ciss	input capacitance @ Vgs

Conditions:

Vds	drain-source voltage for device curve

Model parameters:

CGS	zero-bias gate-source capacitance

This screen estimates the parameter CGS, which is derived from Ciss - Crss. As a check, since most JFETs are designed to be symmetrical, the value found for CGS should be close to that found for CGD (previous screen).

JFET - Passive Gate Leakage

Device curve:

Vdg	drain-gate voltage for Igss
Igss	gate leakage current @ Vdg

Model parameters:

ISR	recombination current saturation value
NR	recombination current emission coefficient
IS	junction saturation current
N	junction emission coefficient
XTI	IS temperature coefficient

This screen derives the generation-recombination current values for the device which, using the capacitance modeling (previous screens), provides the primary leakage mechanism of the device's junction.

Passive reverse current leakage is increased by imperfections in manufacturing and breakdown, which are not modeled.

Also, it is sometimes helpful to set up traces for a few values of temperature (use the **Trace/Add** command) for adjusting XTI.

JFET - Active Gate Leakage

Device curve:

Vdg	drain-gate voltage for Ig (@ Id)
Ig	gate leakage current @ Vdg

Conditions:

Id	drain current for device curve

Model parameters:

ALPHA	impact ionization coefficient
VK	ionization “knee” voltage

This screen estimates active gate current when the JFET is on, which can be much larger than when the JFET is cut off.

Impact ionization by drain current carriers generate carriers in the gate space-charge region, which get swept out through the gate. This causes gate current which is an exponential function of drain voltage and proportional to drain current.

Note that the lowest values of active leakage current are generally less than the passive leakage values (previous screen); this is because the passive values are measured using source and drain shorted together, which usually doubles the junction area and, thus, the current. Active leakage current occurs in the drain-gate junction only, so the lowest levels represent passive leakage for that junction.

JFET - Noise Voltage

Device curve:

Freq	frequency for en (@ Ids)
en	equivalent input noise voltage (in volts/root-hertz) @ Freq

Conditions:

Ids	drain current for device curve

Model parameters:

KF	flicker noise coefficient
AF	flicker noise exponent

This screen estimates the parameter KF, to set the correct amount of flicker noise. AF can be set manually but is normally close to one. The broadband noise of a JFET is “shot” noise and is set by the conductance of the channel.

Power MOSFET Model

The power MOSFET model uses the U.C. Berkeley MOS model, level 3, to incorporate the short-channel effects of vertical devices and nonlinear capacitance effects. In addition to the MOS model, PSpice has been enhanced to include: (i) drain-source ohmic leakage (modeling "off" state conduction), (ii) bulk/substrate series resistance (modeling reverse conduction), and (iii) gate series resistance (modeling switching delay and gate current).

These additions, along with the use of MOS level 3, remove the need to create a subcircuit representation for the power MOSFET device (as described in the paper by Bowers and Neinhaus, below). There is the lead inductance issue, of course, which shows up for any packaged device at high frequencies.

For a more complete description of the power MOSFET models and their derivation, refer to

> J. C. Bowers, and H. A. Neinhaus, *SPICE2 Computer Models for HEXFETs*, Application Note 954A, reprinted in HEXFET Power MOSFET Databook, International Rectifier Corporation #HDB-3.

Also, see **Chapter 2,Analog Devices** on the MOS transistor device for the internal model used by PSpice.

Table 4-6 *MOSFET (NMOS, PMOS) Default Parameters*

Parameter	Default
CBD	1e-9
CGDO	1e-11
CGSO	4e-11
DELTA	0
ETA	0
FC	0.5
GAMMA	0
IS	1e-14
KAPPA	0
KP	2e-5
L	2e-6
LEVEL	3
MJ	0.5
N	1.
PB	0.8
PHI	0.6
RB	1e-3
RD	10e-3
RDS	1e6
RG	5
RS	10e-3
THETA	0
TOX	2e-6
UO	600
VMAX	0
VTO	3
W	0.5
XJ	0

Power MOSFET - Transconductance

Device curve:

Id	drain current for gFS
gFS	forward transconductance @ Id

Model parameters:

KP	transconductance
W	channel width
L	channel length
RS	source ohmic resistance

This screen estimates the basic geometry of the power MOSFET, its conductance parameter, and high-current effects of series resistance in the device.

Many general assumptions are made about the device structure (such as oxide thickness), but the model remains accurate in spite of these assumptions. The transconductance would ideally increase proportional to the square-root of the drain current, but is limited by the effects of RS.

Power MOSFET - Transfer Curve

Device curve:

Vgs	gate-source voltage for Id
Id	@ Vgs

Model parameters:

VTO	zero-bias threshold voltage

This screen estimates the device threshold voltage.

The actual value of VTO is not as important as obtaining a good value of drain current vs. Vgs as the device is used. For library use, use a drain current close to the maximum continuous rating.

Power MOSFET - Rds (on) Resistance

Device curve:

Id	drain current for Rds
Rds	static drain-source on-state resistance @ Id

Conditions:

Vgs	gate-source voltage for device curve

Model parameters:

RD	ohmic drain resistance

This screen estimates the “on-resistance” of the device.

The MOS model has three contributions to the “on-resistance”: the channel resistance of the device, and an ohmic resistance in series with each the source and the drain. This screen adjusts RD so the total resistance is correct. However, RD cannot become negative.

Power MOSFET - Zero-Bias Leakage

Device curve:

Vds	drain-source voltage for Idss
Idss	zero gate voltage drain current @ Vds

Model parameters:

RDS	drain-source shunt resistance (PSpice extension MOS model)

This screen estimates the drain-source leakage of the device. This leakage is due primarily to surface effects and is modeled by a shunt drain-source resistance. Enter the values for the upper list box.

Power MOSFET - Turn-On Charge

Device data:

Qgd	gate drain charge
Qgs	gate-source charge to start switching
Vds	drain source voltage
Id	load (drain) current

Model parameters:

CGSO	gate-source overlap capacitance
CGDO	gate-drain overlap capacitance

This screen estimates the device's stray capacitances associated with the gate. These capacitances, along with the channel capacitance, make up the amounts of charge required to switch the device.

The value Qgs is the amount of charge required to raise the gate-source voltage from zero to that required to support the load current. Qgd is the charge required to discharge the gate-drain ("Miller ") capacitance. It is this charge that brings the device operating from the saturation region to linear region.

Note that the values of CGSO and CGDO are multiplied by the channel width to yield the actual value of the capacitance.

Power MOSFET - Output Capacitance

Device data:

Coss	output capacitance @ Vds
Vds	drain-source voltage for Coss

Model parameters:

CBD	zero-bias bulk-drain junction capacitance
PB	bulk junction potential
MJ	bulk junction grading coefficient
FC	bulk junction forward-bias capacitance coefficient

This screen estimates the output capacitance of the device.

The output capacitance is usually not critical, being small enough when compared using the load currents that are controlled by the device.

Power MOSFET - Switching Time

Device data:

tf	fall time for switching load, Id, using supply, Vdd
Id	load (drain) current for tf
Vdd	supply voltage for tf
Zo	input generator impedance

Model parameters:

RG	gate ohmic resistance

This screen estimates the value of series gate resistance from switching time.

Most power MOSFET devices use a self-aligned process having polysilicon gate material. The polysilicon impedes the gate current, reducing the charging rate of the gate, which increases the turn-on time. While there are many switching times specified (e.g., turn-on delay and rise time), they are all related by the parasitic capacitances, which have already been determined in the “gate charge” screen. Only the series resistance needs to be determined, which can be obtained reliably by using the fall time characteristic.

Note that “fall time” means the period in which the drain current is “falling” in value, not the output voltage.

Power MOSFET - Reverse Drain Current

Device curve:

Vsd	diode (source-drain) forward voltage for Idr
Idr	reverse drain current @ Vsd

Model parameters:

IS	bulk junction saturation current
N	bulk junction emission coefficient
RB	bulk series resistance

This screen estimates the forward voltage drop of the “body” diode.

The actual value of IS is not so important as obtaining a good value of voltage drop vs. current as the device is used.

Operational Amplifier Model

The operational amplifier (opamp) is not an internal PSpice model. Instead, it is an equivalent circuit, or “macro model,” composed of several devices and bound together using the subcircuit feature of PSpice (see .SUBCKT in). The model includes the following effects: (i) input impedance and bias current, (ii) differential and common-mode gain, (iii) open-loop gain and phase vs. frequency, (iv) output slew-rate limiting and resistance, (v) output voltage and current limiting, and (vi) DC power drain.

Figure 4-1 *Parts operational amplifier macromodel (simplified)*

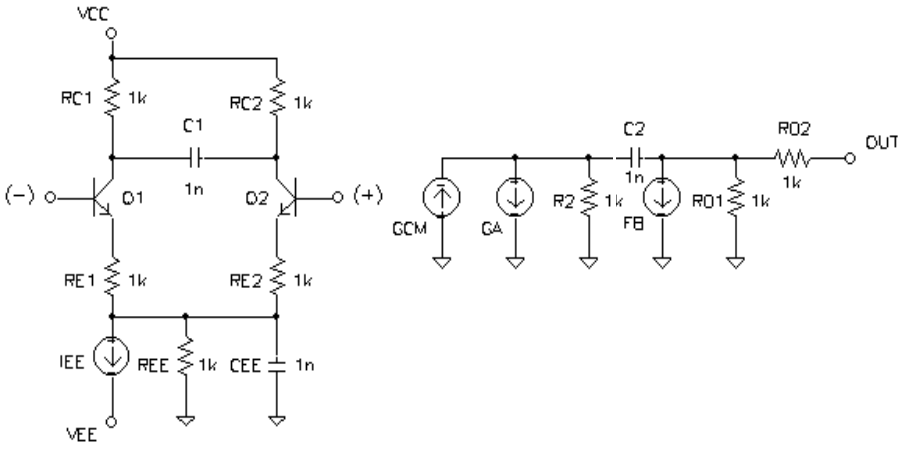


Figure 4-1 shows a simplified version of the opamp macro model. Not shown is local ground generation, output voltage limiting, and output current limiting. Output limiting is controlled by local feedback to the current generator, FB. Other input devices (PNP or JFET) can replace the input transistors.

For more detailed information on opamp macro modeling, refer to

G. R. Boyle, B. M. Cohn, D. O. Pederson, and J. E. Solomon, “Macromodeling of integrated circuit operational amplifiers,” *IEEE Journal of Solid-state Circuits*, SC-9, 353 (1974).

Operational Amplifier - Large Signal Swing

Device data:

+Vpwr	positive power supply
-Vpwr	negative power supply
+Vout	maximum positive output swing
-Vout	maximum negative output swing
+SR	positive-going slew-rate limit
-SR	negative-going slew-rate limit
Pd	quiescent power dissipation

Macro model internal parameters:

VC	output limiter offset (to Vcc)
VE	output limiter offset (to Vee)

This screen sets the value of output voltage limiters but also gathers information that can be useful in later screens. The graph shows the largest amplitude output a sinewave signal can be for a given frequency to have no distortion. This is limited by the amplifier's output swing and slew-rate.

Power supply values: These are the data sheet values used in conjunction with the maximum output values and are not the power supply values for the circuit simulation (which can be different). The opamp model limits the output swing by an amount relative to the power supply, so the output swing limit tracks the power supply in the simulation.

Slew-rates: Since Parts uses primary units (e.g., volts, amps and farads), the variety of ways of specifying slew-rate needs to be converted to volts/second, for example, 5V/uS is 5,000,000 V/S.

Operational Amplifier - Open Loop Gain

Bipolar Input Only

Device data:

Cc	compensation capacitor
Ib	input bias current
Av-dc	open-loop gain (DC)
f-0db	unity gain frequency
CMRR	common-mode rejection ratio

Macro model internal parameters:

BF	input transistor beta
C2	compensation capacitor
CEE	slew-rate limiting capacitor
GA	interstage transconductance
GCM	common-mode transconductance
IEE	input stage current
RC	input stage load resistance
RE	input stage emitter resistance
REE	input stage current source output resistance
RP	power dissipation

This screen completes the input stage and inner stage. The compensation capacitor value (Cc) is sometimes available on the data sheet in the circuit diagram of the opamp. If not, 20-to-30pF is a fair value. For opamps using external compensation, use one of the values on the data sheet for the external capacitor. Then be sure to use that value for the other input data.

About open-loop gain: this is a ratio of input/output signal, i.e., small-signal amplification. Being a pure number, it has no units. If the gain is specified as 20V/mV, the gain is 20,000; if the gain is specified as 90 dB, put in 90 dB (Parts converts *x* dB to $10^{x/20}$).

Unity gain frequency: This frequency is the intersection of a straight-line extension of the of the mid-band, open-loop, gain roll-off to unity gain (zero decibel). The graph can show gain having only the low-frequency pole included. The high-frequency pole is calculated from open-loop phase margin.

CMRR has no frequency dependence.

Operational Amplifier - Open Loop Gain

FET Input Only

Device data:

Cc	compensation capacitor
Ib	input bias current
Av-dc	open-loop gain (DC)
f-0db	unity gain frequency
CMRR	common-mode rejection ratio

Macro model internal parameters:

BETA	input transistor transconductance
C2	compensation capacitor
CSS	slew-rate limiting capacitor
GA	interstage transconductance
GCM	common-mode transconductance
IS	input leakage current
ISS	input stage current
RD	input stage load resistance
RP	power dissipation
RSS	input stage current source output resistance

This screen completes the input stage and inner stage. The compensation capacitor value (Cc) is sometimes available on the data sheet in the circuit diagram of the opamp. If not, 10-to-20pF is a fair value. For opamps using external compensation, use one of the values on the data sheet for the external capacitor. Then be sure to use that value for the other input data.

Open-loop gain: This is a ratio of input/output signal, i.e., small-signal amplification. Being a pure number, it has no units. If the gain is specified as 20V/mV, the gain is 20,000; if the gain is specified as 90 dB, put in 90 dB (Parts converts *x* dB to $10^{x/20}$).

Unity gain frequency: This frequency is the intersection of a straight-line extension of the of the mid-band, open-loop, gain roll-off to unity gain (zero decibel). The graph can show gain having only the low-frequency pole included. The high-frequency pole is calculated from open-loop phase margin.

CMRR has no frequency dependence.

Operational Amplifier - Open Loop Phase

Device data:

Phi	phase margin (in degrees) @ unity gain frequency

Macro model internal parameter:

C1	phase control capacitor

This screen adjusts the open-loop unity-gain phase margin, which models the high-frequency pole. Sometimes this value is not available in a table but can be found from a graph. This value is not critical for lower-frequency circuits or lower-Q filters: just use the value we provided, which is typical for normal opamps.

Operational Amplifier - Maximum Output Swing

Device data:

Ro-dc	DC output resistance
Ro-ac	AC output resistance
Ios	short-circuit output current limit

Macro model internal parameters:

RO1	output resistor #1
RO2	output resistor #2
GB	output stage transconductance

This screen adjusts the output drive. The graph shows the maximum output level for a resistive load. The data sheet usually lists an output resistance Ro = Ro-dc + Ro-ac. Split this value so that Ro-dc is about two times Ro-ac.

Voltage Comparator Model

The voltage comparator is not an internal PSpice model. Instead, it is an equivalent circuit, or “macro model,” composed of several devices and bound together using the subcircuit feature of PSpice (see .SUBCKT in **Chapter 1,Commands**). The model includes the following effects: (i) input impedance and bias current, (ii) differential gain, (iii) output resistive and capacitive loading, (iv) time delay and slew-rate vs. input overdrive, and (v) DC power drain.

Voltage Comparator - Transfer Function

Device data:

+Vpwr	positive power supply
-Vpwr	negative power supply
+Vicr	positive common-mode range
-Vicr	negative common-mode range
Ib	input bias current
Avd	DC gain
Rl	output load resistance
Pd	power dissipation

Macro model internal parameters:

BF1	input stage gain
BF5	output stage gain
RP	power dissipation resistance
VI	input offset

This screen sets gain values and an input offset (to model comparators whose common-mode input includes ground). The screen shows the transfer function, usually uninformative, except it tells when something is happening.

About power supply values: these are the data sheet values used in conjunction with the maximum output values, and are not the power supply values for the circuit simulation (which can be different).

Voltage Comparator - Falling Delay

Device data:

Vst	input voltage step size
Vod	input voltage step overdrive
td	delay time

Macro model internal parameter:

TR3	input stage reverse transit time

This screen sets reaction time to input signals. The data sheet usually gives “falling delay” which includes some of the transition in the output waveform (from 100% to 90%). Usually the transition is much faster than the delay and can be ignored (or subtracted from the value). The precise value is not critical given the unit-to-unit variation.

Voltage Comparator - Transition Time

Device data:

Vst	input voltage step size
Vod	input voltage step overdrive
ttr	transition time

Macro model internal parameter:

TF5	output transistor forward transit time

This screen sets the “slew-rate” of the output. The data sheet usually gives a value going from 90% to 10%, which is within 25% of the full swing time. The precise value is not critical given the unit-to-unit variation.

Voltage Comparator - Rising Delay

Device data:

Vst	input voltage step size
Vod	input voltage step overdrive
td	delay time

Macro model internal parameter:

TR5	output transistor reverse transit time

This screen sets the reaction to input signals, but in the opposite direction. The data sheet usually gives “rising delay,” which includes some of the transition in the output waveform (from 0% to 10%). Usually the transition is much faster than the delay and can be ignored (or subtracted from the value). The precise value is not critical given the unit-to-unit variation.

Nonlinear Magnetic Core Model

The nonlinear magnetic core model uses a derivative of the Jiles-Atherton formulation to provide a closed-form, analytic solution. In its current implementation, the LEVEL=2 model is missing the interdomain coupling and damping effects; this makes the model more suitable for ferrite and MPP (Magnetic Packed Powder) materials, and can provide valid results for all analyses. The LEVEL=2 model can also be more efficient than the original, numerically calculated, LEVEL=1 model. The basic difference in these two models is in the formula for the anhysteric curve, as detailed in the K-device description in the **Chapter 2,Analog Devices**.

There is only one Parts screen for extracting parameters for this device, since Parts is only characterizing the bulk magnetic properties of the core material. When finished, the text of the model could require editing to provide values for a particular core's mean magnetic cross-sectional AREA, mean magnetic PATH length, and the effective air GAP length and PACK (stacking) factor, if appropriate. Or, this model could be referenced as is, by using the AKO syntax, as the basis for other models using different geometries.

Nonlinear Magnetic Cores - Hysteresis Curve

Device curve:

H	magnetic influence (in Oersteds)
B	magnetic flux (in Gauss)

Conditions:

μi	initial permeability

Model parameters:

Ms	magnetization saturation
A	thermal energy parameter
C	domain flexing parameter
K	domain anisotropy parameter

This screen estimates the bulk material parameters from the envelope of the B-H curve and the value for the initial permeability of the material. The “initial” B-H curve, starting from the origin, is also shown but is characterized only by the value for initial permeability.

Unlike most screens in Parts, in this screen the extent of the X axis plays a role in extracting the model parameters. Owing to the hysteresis (memory) effects in magnetic materials, how the material behaves depends on, virtually, its entire history. This means the B-H curve is dependent on how strong a field it has been subjected to. To accommodate this behavior, Parts simulates a range of fields using magnitudes up to the maximum extent displayed by the X axis. For example, if the X axis is set for a range of -5 to + 5, Parts can use fields in that range when extracting the model parameters; the same range of fields can also be used if the X axis is set for a range of zero to +5. After the parameters have been fitted, the X axis can be changed to see the material’s behavior under a different range of external fields.

Warning: Numerous evaluations are made when extracting these model parameters. Even the fastest computers can appear to stall, momentarily, while performing the extraction.

Voltage Regulator Model

The positive adjustable voltage regulator model is not an internal PSpice model. Instead, it is an equivalent circuit, or “macro model,” composed of several devices and bound together using the subcircuit feature of PSpice (see .SUBCKT in **Chapter 1,Commands**). The model is a 3-terminal floating series regulator and includes the following effects: (i) reference voltage, (ii) adjustment pin current, (iii) output impedance, (iv) dropout voltage, (v) current limit, and (vi) foldback current. The model does not include the following effects: (i) line regulation, (ii) load regulation, (iii) ripple rejection having frequency dependence, (iv) temperature, and (v) noise.

For more information on the positive adjustable voltage regulator model, refer to:

[1] G.M. Wierzba, K.V. Noren, “A SPICE Macromodel for an Adjustable Positive Voltage Regulator.”

Voltage Regulator - Reference Voltage

Device Data:

Vref reference voltage

Conditions:

Dropoutdropout voltage

$(Vi\text{-}Vo)_{max}$maximum input-output voltage differential

IOmin minimum output current to maintain regulation

Model Parameters:

VREF reference voltage
N emission coefficient

This screen shows the reference voltage across the output (OUT) pin and adjustment (ADJ) pin as the input-output voltage differential increases from 0V to (Vi-Vo)max. The parameter (Vi-Vo)max is used for graphing purposes only. It does not affect the model characteristics.

The dropout voltage specifies the minimum input-output voltage differential below which the circuit ceases to regulate. This parameter is either obtained from the condition of the reference voltage parameter or is given as a parameter by itself in data sheets.

The minimum output current parameter is obtained from the condition of the reference voltage parameter in data sheets. Be careful that this parameter is also given at the condition Vi-Vo=(Vi-Vo)max. Do not use this value.

Voltage Regulator - Adjustment Pin Current

Device Data:

Iadj adjustment pin current

Model Parameter:

BETA transconductance of JFET transistor

This screen displays the adjustment pin current as the input voltage increases from zero volts to (Vi-Vo)max. The parameter (Vi-Vo)max is used for graphing purposes only. It does not affect the model characteristics.

The adjustment pin current represents an error term in the design equation:

Vo = Vref (1 + R2/R1) + Iadj * R2

where R1 and R2 are two external resistors. Usually, Iadj is small that the voltage Iadj*R2 is negligible in the above equation.

Voltage Regulator - Output Impedance

Device Data:

Zout	low frequency output impedance
Zero	dominant zero frequency of output impedance
RR	low frequency ripple rejection in decibels

Conditions:

Frequencyfrequency at which Zout and RR are obtained:

IO	output current at which Zout and RR are obtained

Model Parameters:

VAF	Early voltage of output pass transistor
CPZ	output impedance zero capacitor

This screen calculates the output impedance over frequency. The output impedance is the impedance seen looking back into the OUT pin, excluding the effects of any external components connected to it. Data sheets usually present the output impedance graphically. Obtain Zout at the same frequency where the ripple rejection value is given. Assume there is no capacitance connected to the adjustment pin (Cadj=0).

The condition Frequency is used for parameter referencing purposes only. It does not affect the model characteristics.

Voltage Regulator - Current Limit

Device Data:

IOmax maximum output current

Device Curve:

Iofb foldback current

Vi-Vo input-output voltage differential

Model Parameters:

RB2 base resistance of output pass transistor

ESC1 coefficient of current limit voltage source

ESC2 coefficient of current limit voltage source

EFB1 coefficient of foldback current voltage source

EFB2 coefficient of foldback current voltage source

EB first stage voltage gain

This screen shows the output current as the voltage Vi-Vo increases from 0V to (Vi-Vo)max as given in voltage reference screen when the output is shorted to ground.

The foldback current is the current when the maximum output current is reduced using increasing Vi-Vo voltage. This current is shown as part of the current limit graph given in data sheets. Parts performs a quadratic curve fit on the given data points of the foldback current.

Voltage Reference Model

The voltage reference element is not an internal PSpice model. Instead, it is an equivalent circuit, or “macro model,” composed of several devices and bound together using the subcircuit feature of PSpice (see .SUBCKT in **Chapter 1,Commands**). The model is a 2-terminal reference circuit diode and includes the following effects:

- reverse dynamic impedance having reverse current dependence,
- reference (reverse breakdown) voltage,
- reference voltage temperature drift,
- reverse leakage characteristics, and
- forward current characteristics.

The model does not include the following effects:

- reverse dynamic impedance having frequency dependence,
- noise voltage, and
- response time.

The voltage reference element is based partly on the following paper:

[1] S. Wong and C. Hu, “SPICE Macro Model for the Simulation of Zener Diode I-V Characteristics,” IEEE Circuits & Devices, Vol. 7, No. 4, July 1991, pp. 9-12.

Voltage Reference - Reverse Dynamic Impedance

Device Curve:

Ir	reverse current
Rz	dynamic impedance

Model Parameters:

NZ	reverse breakdown coefficient
RZ	dynamic impedance

This screen shows the reverse dynamic impedance using reverse current at a low frequency. The dynamic impedance is the impedance seen looking into the cathode terminal at a given reverse current.

Data sheets usually present the reverse dynamic impedance characteristics graphically. Obtain Rz at nominal operating temperature. Ir requires positive values.

Voltage Reference - Reference Voltage

Device Data:

Vref	reverse breakdown voltage
Ir	reverse current at which Vref is obtained.
Irmax	absolute maximum reverse breakdown current

Model Parameters:

RBV	reverse breakdown reference resistance
IRMAX	absolute maximum reverse breakdown current

This screen displays the reverse characteristics as the reverse current increases to the absolute maximum breakdown current. The portion of the curve below Vref is shown as an approximation here. It is more accurately modeled in the Reverse Characteristics screen.

The reverse breakdown voltage is the reference voltage which exhibits tight tolerance and low temperature drift. Obtain Vref at nominal operating temperature (Ir). Data sheets usually reference Irmax under the absolute maximum ratings section. Vref requires positive values.

Temperature Drift

Device Curve:

Temp	operating temperature in degrees Celsius
Vref	reverse breakdown voltage at Temp

Model Parameters:

TC1	first-order temperature coefficient
TC2	second-order temperature coefficient

This screen shows the reverse breakdown voltage variation with temperature at the reverse current. Ir is provided in the previous Reference Voltage screen. The curve can pass through Vref at nominal temperature (as specified in the previous Reference Voltage screen).

Parts does a quadratic curve fit to the given data points. If the temperature drift is not shown graphically in data sheets, and only the average temperature coefficient is provided, enter data points so that the maximum deviation of Vref, divided by the maximum temperature range, can result in the correct average temperature coefficient.

Voltage Reference - Reverse Characteristics

Device Curve:

Vr	reverse voltage
Ir	reverse current

Model Parameters:

IREV	reverse saturation current
NREV	reverse current coefficient

This screen shows the reverse characteristics for reverse voltages up to Vref. See the Reference Voltage screen for voltages greater than Vref.

Data sheets usually show the reverse characteristics graphically. Obtain data points at nominal temperature. Both Vr and Ir require positive values.

Voltage Reference - Forward Characteristics

Device Curve:

Ifwd	forward current @ Vfwd
Vfwd	forward voltage across junction for Ifwd

Model Parameters:

IS	saturation current
N	emission coefficient
RS	series resistance
IKF	high-injection “knee” current
XTI	IS temperature coefficient

This screen estimates the parameters IS and RS from three voltage and current values. Try to include data from low current values (where the increase in current is exponential), moderate current values, and high current value (where the increase in current is clearly resistive).

Also, it is sometimes helpful to set up traces for a few values of temperature (use the **Trace/Add** command), for adjusting XTI.

Customizing Device Equations

5

Overview

This chapter provides instruction on how to use the Device Equations option.

The purpose of the Device Equations option is to allow the built-in model equations to be changed for one or more of the semiconductor devices. This option is not an addition to PSpice: it is a different packaging of the program which includes the source code for the device model subroutines.

Introduction

There are several kinds of changes that can be made using the Device Equations option. These changes include in ascending order of complexity

changing a parameter's name
giving a parameter an alias
adding a parameter
changing the device equations
adding a new device
specifying new internal device structure

Note *In order to take advantage of this option, Microsoft Visual C++ 32-bit Compiler, version 4.1, must be installed. These compilers will run under Windows 95 or Windows NT. (For the SunPro SPARCompiler 3 is required.)*

Making Device Model Changes

To get started, look at the files M.H and MOS.C, which implement the MOSFET equations. The other devices have a similar structure.

M.H contains two important data structure definitions, the structure for the MOS transistor (struct m_), and the structure for the MOS model (struct M_).

During read-in, the simulator creates a copy of the transistor structure for every MOSFET in the circuit and a copy of the model structure for every .MODEL statement of type NMOS or PMOS. The transistor structure is set up using information particular to that transistor, such as the nodes to which it is connected, its length and width, and the locations of its entries in the circuit's conductance matrix. All parameters of the model structure are set up using the values from the .MODEL statement, if one exists; otherwise, the default values are used.

The transistor structure corresponds to the LOC, LOCV, and LX tables in U.C. Berkeley SPICE2. The model structure corresponds to the LOC and LOCM tables in SPICE.

Note *Do not change the transistor structure. It is included only to allow compiling of MOS.C.*

The simulator needs some way to associate each entry in the model structure using a model parameter name (and default value) in the .MODEL statement. This is accomplished using the ASSOCIATE macro. Just below the model structure in M.H a list of all the parameters can be seen, each in an ASSOCIATE macro. The occurrence of ASSOCIATE "binds" together the structure entry, the parameter name, and a default value. The read-in section of the simulator uses this information to parse the .MODEL statement.

The general procedure to follow for changing the MOSFET model is to change M.H and MOS.C, compile all .C files that use the MOSFET model, and then link the object files to produce a new program (.EXE) file.

Changing a Parameter's Name

This is the easiest change. Find the parameter in the list of ASSOCIATE macros. Change the parameter's name (last item on the line) and/or the default value (middle item). The names and defaults of the model parameters that are supplied can be changed, as well as those parameters that are added.

When the simulator runs, it prints the parameter values for each .MODEL statement unless the NOMOD option is used in the .OPTIONS statement. Normally only parameters which have not been defaulted are listed. A parameter can be forced to be listed, whether or not it has been defaulted, by preceding its name using an "*". For example, VTO is listed that way in M.H.

Giving a Parameter an Alias

Sometimes a parameter requires an alternate name (an alias). Several bipolar model parameters, such as ISE, already have alternate names. The alias for ISE is C2. Look in Q.H at the occurrences of the parameters ISE and C2 in the ASSOCIATE macros for an example of how this is accomplished. There is only one entry in the model structure (Q_ise) for the parameter, but there are two ASSOCIATE entries. This means that either name (ISE or C2) on the .MODEL statement can put a number into the structure entry Q_ise.

Note *When model parameters are listed, the first name found in the ASSOCIATE list (searching downward) is the name which is echoed on the output.*

Insert the new name first if it is the name to be printed.

Adding a Parameter

Adding a parameter is probably the most common case. The parameter must be added to both the model structure (e.g., struct M_) and the corresponding ASSOCIATE list. It is recommended to follow MicroSim's naming convention (e.g., M_wd and M_vto), but it is not required.

Model parameters are set forth as pairs of elements instead of simple floating point values. This is to provide the use of expressions for model parameters. Because of this, when adding a parameter (for example, M_new), the following line is required:

MXPR(M_new, Mx_new);

instead of

float M_new;

Note *Do not modify the value of the Mx_new structure element.*

The read-in mechanism can handle expressions for user-added parameters. By the time the model code is called, the expressions have been evaluated and their value placed in the appropriate fields. See the include file "m.h" for further examples and comments.

When the simulator is doing a read-in, model parameters are listed for each .MODEL statement (unless NOMOD has been specified on the .OPTIONS statement). Normally, only those parameters that have not been defaulted are listed. A parameter can be forced to be listed, even if it has been defaulted, by preceding its name using an "*" in the ASSOCIATE macro. For instance, VTO in M.H is listed in that manner.

The default value, OMITTED, is used by the simulator to force the calculation of a parameter's value during read-in. For instance, VTO is calculated from other values if it is not given a value. These calculations are built into the read-in and are fixed. It is recommended that parameters that are added by the user be given a normal default value and not be computed by using OMITTED.

Once the parameter has been added, the model structure becomes one parameter longer, and the read-in section of PSpice places a value in its entry. The parameter can now be used in the device code (e.g., MOS.C).

Changing the Device Equations

The device equations are in the file having the same name as the type of device (DIODE.C, BJT.C, JFET.C, MOS.C, GASFET.C). The code in these subroutines use the model parameters and the device's terminal voltages to calculate the branch currents and conductances, and, during transient analysis, the terminal charges and branch capacitances. These equations are neither simple nor easy. A good understanding of U.C. Berkeley's SPICE2G is recommended before making such a change. Two useful references are:

[1] L. W. Nagel, *SPICE2: A Computer Program to Simulate Semiconductor Circuits*, Memorandum No. M520, May 1975.

[2] Ellis Cohen, *Program Reference for SPICE2*, Memorandum No. M592, June 1976.

which are available by sending a check for $30.00 and $15.00, respectively, payable to *The Regents of the University of California* to this address:

Cindy Manly
EECS/ERL Industrial Support Office
497 Cory Hall
University of California
Berkeley, CA 94720

Eight weeks are required for delivery.

The code in each of the device source files is arranged into separate functional subsections. Each subsection occurs at least once, but can occur several times for devices that have more than one level. The subsections required are outlined in Table 5-1.

Table 5-1 *Functional Subsections of the Device Source File*

Subsection	Description
Initialization	This consists of locating and binding the device instance and its model, initializing any local variables, and obtaining appropriate values for the device branch voltages. The branch voltages (e.g., vds, vgs) are set differently depending upon whether there are user-specified initial conditions (using IC= or .IC), and on whether the present Newton Raphson cycle has finished or not.
Computing new nonlinear branch voltage:	This is needed to monitor progress towards a Newton Raphson solution.
Test if the solution has changed:	If there is not significant change bypass the rest of the computation. Otherwise, continue.
Limit any nonlinear branch voltages:	This code uses the macro PNJLIM() to insure that the branch voltages are in the appropriate operating region.
Compute currents and conductances:	This is the meat of the Device Equations code, and involves obtaining all the branch currents (e.g., ibs, ibd) as well as all the derivatives to be used in the conductance matrix.
Charge calculations:	Internal charges are calculated and updated.
Check convergence:	Check to see if the nonlinear device branches now have values that are within a small tolerance range of those obtained in the last repeat cycle, and set a return flag to signal whether the device converged.
Load the current vector and conductance matrix:	The macro Y_MATRIX () is used to obtain handles to the proper matrix elements, and the elements are assigned their values based on the present evaluation of the device equations and derivatives.

SPICE2G is written in FORTRAN, whereas PSpice is in C. For the device subroutines, as much correspondence as possible has been maintained between the two. Because of FORTRAN, SPICE kept integer and real numbers in different tables: NODPLC (indexed by LOC) and VALUE (indexed by LOCV or LOCM). In PSpice, these have been combined into one structure (e.g., struct m_).

The state vector information is constructed somewhat differently, though the overall pattern is similar. In SPICE the state vector information is kept in a set of vectors in VALUE. There is one vector for each time point "remembered" (from 4 to 7, depending on the order of the integration method). Each device's LOC table contains an offset, LX, to its portion of the information in each state vector. In PSpice the number of state vectors is fixed, and each device's state information is kept in its own device structure (e.g., struct m_). For instance, for MOSFETs the state vectors are an array, *`struct msv_def m_sv[MSTVCT]`* in struct m_. MSTVCT is the number of state vectors and is defined in TRAN.H to be equal to 4. The definition of msv_def (also in M.H) lists the various currents, conductances, charges, and capacitances that are in the state vector. Finally, M.H contains a set of #defines, which allows accessing of the entries to the state vectors by name. It is these (upper case) names which are then used in MOS.C. This may seem like a roundabout way of constructing the state vector information, but the actual usage (in MOS.C) is quite straightforward and is similar to that in SPICE.

Adding a New Device

The Device Equations option does not allow the addition of an entirely new device. However, in many cases the same thing can be achieved by making use of an existing device.

Suppose, for example, that a lightning arrester device is to be added. The lightning arrester has two terminals, therefore it can be built into the diode equations, because the diode also has two terminals. This means that in the circuit (.CIR) file the lightning arresters would use the letter "D" to start and would refer to a .MODEL statement of the type "D."

At first glance it appears that this would preclude using diodes in circuits, since they have been replaced by lightning arresters. This problem is avoided by keeping all the diode model parameters, adding the lightning arrester parameters, adding a LEVEL parameter, and giving the LEVEL parameter a default of 1. In the diode subroutine (in DIODE.C), a large "if" test would select all the old diode code if LEVEL=1 and all the new lightning arrester code otherwise. The new LEVEL parameter would switch between diode and lightning arrester.

This approach can be extended to as many devices as wanted. This could be:

LEVEL=1 as a diode,
LEVEL=2 as a lightning arrester,
LEVEL=3 as a gas discharge tube,

and so on. The restriction is that all of the devices added to the "diode" must have two terminals. If the device to be added has three terminals, it must be built into a three terminal device, such as the JFET. The highest number of terminals that can be modeled is four, using the MOSFET. There is not a good way to add devices, such as pentodes, that have five or more terminals.

Specifying Device Internal Structure

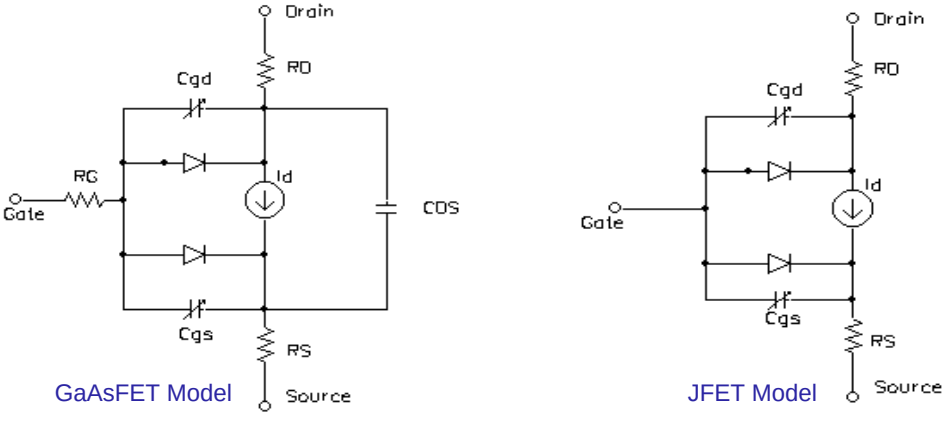
Sometimes it is desirable to change the topology of a device in order to accommodate a more elaborate set of parasitic resistances and/or capacitances. To do this requires that positions in the conductance matrix be assigned to include the terms that the additional equations generate. This requires five steps:

1. The device header file must have all of the new internal nodes and matrix conductance terms added to the device structure.
2. The new matrix elements must be allocated.
3. Handles must be provided to access the new matrix elements and bind the nodes to the branches.
4. Logic could be needed to support device model parameter checking and updating.

   The new device equations have to be added to the device code.

   This process can be illustrated by looking at PSpice's JFET and GaAsFET devices. The topologies of these two devices are nearly identical, except that the GaAsFET has an additional internal capacitance, CDS, between the source and drain, and an additional internal resistance, RG, at the gate. This gives the GaAsFET topology one additional node where RG joins the rest of the structure, and two additional internal branches. The following figure shows the internal schematic diagram.

Figure 5-1 *GaAsFET and JFET Device Models*



These differences are reflected in the device structure definitions in J.H and B.H. Each of the devices node is given a name and declared to be of type CKT_IDX.

The JFET device structure, j_, lists the two internal nodes j_d and j_s, while the GaAsFET device structure, b_, has three internal nodes b_d, b_s, and a new one, b_g. The two additional branches in the GaAsFET require three new matrix conductance terms.

The conductance terms are declared type MTX_IDX, and are listed immediately following the internal nodes.

The JFET has a term j_GG, which is the term that appears on the matrix diagonal for the external gate node.

The GaAsFET has an additional gate node which requires one additional matrix diagonal conductance term, b_gg, along with two off-diagonal conductance terms, b_Gg and b_gG. These are used by the source code in GASFET.C to designate where the conductance terms associated with RG go when the matrix is loaded. CDS doesn't need any additional nodes or matrix terms because the items required are already in place to accommodate the parallel current source, id.

With the nodes and conductance terms taken care of in the device header file, the first step is completed. Step two involves setting up memory allocation to properly incorporate the new equations into the conductance matrix. This is accomplished by modifying DEMATPTR.C. In this file are functions JMatPtr() and BMatPtr(). These functions call the function Reserve() once for each conductance matrix term that was declared in the header file. For instance, when b_gg, b_Gg, and b_gG are added for the GaAsFET, these require corresponding code in BMatPtr() as follows:

```
flag &= Reserve (ng,ng);
flag &= Reserve (nG,ng);
flag &= Reserve (ng,nG);
```

The arguments ng and nG are local variables which serve as aliases for the respective device nodes, b_g and b_G.

The mechanics of step three, binding the nodes and branches, are very similar to the mechanics of step two. This time DEMATLOC.C is modified. The functions of interest are JMatLoc() and BMatLoc(), and they now call Indxcl() instead of Reserve(). The GaAsFET again has three more lines of code:

```
flag &= Indxcl (&(bloc->b_gg),ng,ng);
```

```
flag &= Indxcl (&(bloc->b_Gg),nG,ng);
flag &= Indxcl (&(bloc->b_gG),ng,nG);
```

Step four, handling model parameters, is basically the same as it would be for a case not involving topology changes, with one significant exception. This requires handling the case where the parasitics associated with an internal node can be zero. In this case the node must be generated conditionally. An instance of this is the GaAsFET internal resistance RG. If RG is zero, the parasitic resistance between the internal node b_g and the external node b_G can be removed from the circuit. This is accomplished in the function B_AddInternalNodes() in DEMODCHK.C, using the following line of code:

```
INTERNAL_NODE(P->B_rg,b_g,b_G);
```

INTERNAL_NODE() is a macro which performs the required logic, depending on whether the model parameter B_rg is zero or not. The other two calls to this macro in B_AddInternalNodes() correspond to the RD and RS resistances that also exist for the JFET.

The final fifth step doesn't involve any further topological considerations, and is carried out just as it would be otherwise.

Recompiling/Linking the Device Equations Option

The object and source files necessary to make the Windows version of PSpice (PSPICE.EXE) are installed in a directory called DEVEQU. A project file, “PSPICE.MAK,” is included to compile and link the program.

The object code assumes that the Microsoft Visual C++ 32-bit edition, version 2.2, or 4.0 compiler (for Windows 95 or Windows NT) is being used.

To create a new PSPICE.EXE, simply load “PSPICE.MAK” into the Visual C++ development environment, and choose the “Build PSPICE.exe” menu item from the “Project” menu.

Once PSPICE.EXE is built, simply add it as an icon to the MicroSim program group. The PSpice directory should also contain the file “msim.iff.”

For information on how to obtain the Microsoft compiler, call or write:

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
Telephone: (800) 426-9400

Convergence and “Time Step Too Small Errors”

6

Overview

This chapter discusses common errors and convergence problems in PSpice.

Introduction

In order to calculate the bias point, DC sweep and transient analysis for analog devices PSpice must solve a set of nonlinear equations which describe the circuit's behavior. This is accomplished by using an iterative technique - the Newton-Raphson algorithm - which starts by having an initial approximation to the solution and iteratively improves it until successive voltages and currents converge to the same result.

In a few cases PSpice cannot find a solution to the nonlinear circuit equations. This is generally called a “convergence problem” because the symptom is that the Newton-Raphson repeating series cannot converge onto a consistent set of voltages and currents. The following discussion gives some background on the algorithms in PSpice and some guidelines for avoiding convergence problems.

The transient analysis has the additional possibility of being unable to continue because the time step required becomes too small from something in the circuit moving too fast. This is also discussed below.

The AC and noise analyses are linear and do not use an iterative algorithm. The following discussion does not apply to them. Digital devices are evaluated using boolean algebra and this discussion does not apply to them either.

Concepts

Newton-Raphson Requirements

The Newton-Raphson algorithm has the very nice property that *it is guaranteed to converge to a solution*. However, this nice property has some serious strings attached:

1. The nonlinear equations must have a solution
2. The equations must be continuous
3. The algorithm needs the equations' derivatives
4. The initial approximation must be close enough to the solution

Each of these can be taken in order. One must keep in mind that PSpice's algorithms are used in computer hardware that has finite precision and finite dynamic range which produce these limits:

- voltages and currents in PSpice are limited to +/-1e10 volts and amps,
- derivatives in PSpice are limited to 1e14, and
- the arithmetic used in PSpice is double precision and has 15 digits of accuracy.

Is There a Solution?

The answer is yes for any physically realistic circuit. However, it is not difficult to set up a circuit which does not have a solution within the limits of PSpice’s numerics. Consider, for example, a voltage source of one megavolt connected to a resistor of one micro-ohm. This circuit does not have a solution within the dynamic range of currents (+/- 1e10 amps). Here is a sneakier example:

```
V1                      1,    0     5v
D1                      1,    0     DMOD
.MODEL                              DMOD(IS=1e-16)
```

The problem here is that the diode model has no series resistance. Referring to **Chapter 2,Analog Devices**, it can be shown that the current through a diode is:

$$I = IS*e^{V/(N*k*T)}$$

N defaults to one and k*T at room temperature is about .025 volts. So, in this example the current through the diode would be:

$$I = 1e\text{-}16*e^{200} = 7.22e70 \text{ amps}$$

This circuit also does not have solution within the limits of the dynamic range of PSpice. In general, you should be careful of components without limits built into them. Extra care is needed when using the expressions for controlled sources (i.e., behavioral modeling). It is easy to write expressions whose values can be very large.

Are the Equations Continuous?

The device equations built into PSpice are continuous. The functions available for behavioral modeling are also continuous (there are several functions, such as int(x), which cannot be added because of this). So, for physically realistic circuits the equations can also be continuous. Exceptions that come are usually from exceeding the limits of the numerics in PSpice. Consider the following attempt to approximate an ideal switch using the diode model:

```
.MODEL DMOD(IS=1e-16 N=1e-6)
```

The current through this diode is:

$$I = 1e\text{-}16 * e^{V/(N*.025)} = 1e\text{-}16 * e^{V/25e\text{-}9}$$

Because the denominator in the exponential is so small, the current I is essentially zero for $V < 0$ and almost infinite for $V > 0$. Even if there are external components that limit the current the “knee” of the diode's I-V curve is so sharp that it is almost a discontinuity. The caution again is to avoid unrealistic model parameters. Behavioral modeling expressions need extra care.

Are the Derivatives Correct?

The device equations built into PSpice include the derivatives and these are correct. Depending on the device, the physical meaning of the derivatives is small-signal conductance, transconductance or gain. Unrealistic model parameters can exceed the limit of 1e14, but it requires some effort. The main thing to look at is the behavioral modeling expressions, especially those having denominators.

Is the Initial Approximation Close Enough?

It seems like a Catch-22: Newton-Raphson is guaranteed to converge only if the analysis is started close to the answer. Worse yet, there is no measurement that can tell how close is close enough.

PSpice gets around this by making heavy use of continuity. Each analysis starts from a known solution and uses a variable step size to find the next solution. If the next solution does not converge PSpice reduces the step size, falls back and tries again.

Bias point

The hardest part of the whole process is getting started. That is, finding the bias point. PSpice first tries with the power supplies set to 100%. A solution is not guaranteed, but most of the time the PSpice algorithm finds one. If not, then the power supplies are cut back to almost zero. They are cut to a level small enough that *all nonlinearities are turned off*. When the circuit is linear a solution can be found (very near zero, of course). Then, PSpice works its way back up to 100% power supplies using a variable step size.

Once a bias point is found the transient analysis can be run. It starts from a known solution (the bias point) and steps forward in time. The step size is variable and is reduced as needed to find further solutions.

DC sweep

The DC sweep uses a hybrid approach. It uses the bias point algorithm (varying the power supplies) to get started. For subsequent steps it uses the previous solution as the initial approximation. The sweep step is not variable, however. If a solution cannot be found at a step then the bias point algorithm is used for that step.

The whole process relies heavily on continuity. It also requires that the circuit be linear when the supplies are turned off.

STEPGMIN

An alterative algorithm is GMIN stepping. This is not obtained by default, and is enabled by specifying the circuit analysis option STEPGMIN (either using .OPTION STEPGMIN in the netlist, or by making the appropriate choice from the Analysis/Setup/Options menu). When enabled, the GMIN stepping algorithm is applied after the circuit fails to converge with the power supplies at 100 percent, and if GMIN stepping also fails, the supplies are then cut back to almost zero.

GMIN stepping attempts to find a solution by starting the repeating cycle with a large value of GMIN, initially 1.0e10 times the nominal value. If a solution is found at this setting it then reduces GMIN by a factor of 10, and tries again. This continues until either GMIN is back to the nominal value, or a repeating cycle fails to converge. In the latter case, GMIN is restored to the nominal value and the power supplies are stepped.

Bias Point and DC Sweep

Power supply stepping

As previously discussed, PSpice uses a proprietary algorithm which finds a continuous path from zero power supplies levels to 100%. It starts at almost zero (.001%) power supplies levels and works its way back up to the 100% levels. The minimum step size is 1e-6 (.0001%). The first repeating series of the first step *starts at zero for all voltages*.

Semiconductors

Model parameters

The first consideration for semiconductors is to avoid physically unrealistic model parameters. Remember that as PSpice steps the power supplies up it has to step carefully through the turn on transition for each device. In the diode example above, for the setting N=1e-6, the knee of the I-V curve would be too sharp for PSpice to maintain its continuity within the power supply step size limit of 1e-6.

Unguarded p-n junctions

A second consideration is to avoid “unguarded” p-n junctions (no series resistance). The above diode example also applies to the p-n junctions inside bipolar transistors, MOSFETs (drain-bulk and source-bulk), JFETs and GaAsFETs.

No leakage resistance

A third consideration is to avoid situations which could have an ideal current source pushing current into a reverse-biased p-n junction without a shunt resistance. Since p-n junctions in PSpice have (almost) no leakage resistance and would cause the junction's voltage to go beyond 1e10 volts.

The model libraries which are part of PSpice follow these guidelines.

Typos can cause unrealistic device parameters. The following MOSFET:

```
M1                3, 2, 1, 0  MMOD        L=5 W=3
```

has a length of five meters and a width of three meters instead of micrometers. It should have been:

```
M1                3, 2, 1, 0  MMOD        L=5u W=3u
```

PSpice flags an error for L too large, but cannot for W because power MOSFETs are so interdigitated (a zipper-like trace) that their effective W can be very high. The LIST option can show this kind of problem. When the devices are listed in the output file their values are shown in scientific notation making it easy to spot unusual values.

Switches

PSpice switches have gain in their transition region. If several are cascaded then the cumulative gain can easily exceed the derivative limit of 1e14. This can happen when modeling simple logic gates using totem-pole switches and there are several gates in cascaded in series. Usually a cascade of two switches works but three or more can cause trouble.

Behavioral Modeling Expressions

Range limits

Voltages and currents in PSpice are limited to the range +/- 1e10. Care must be taken that the output of expressions fall within this range. This is especially important when one is building an electrical analog of a mechanical, hydraulic or other type of system.

Source limits

Another consideration is that the controlled sources must turn off when the supplies are almost 0 (.001%). There is special code in PSpice which “squelches” the controlled sources in a continuous way near 0 supplies. However, care should still be taken using expressions that have denominators. Take, for example, a constant power load:

```
GLOAD                          3, 5          VALUE = {2Watts/V(3,5)}
```

The first repeating series starts with V(3,5) = 0 and the current through GLOAD would be infinite (actually, the code in PSpice which does the division clips the result to a finite value). The “squelching” code is required to be a smooth and well-behaved function.

Note *The “squelching” code cannot be “strong” enough to suppress dividing by 0.*

The result is that GLOAD does not turn off near 0 power supplies. A better way is described in the application note Modeling Constant Power Loads. The “squelching” code is sufficient for turning off all expressions except those having denominators. In general, though, it is good practice to constrain expressions having the LIMIT function to keep results within physically realistic bounds. For example, a first approximation to an opamp that has an open loop gain of 100,000 is:

```
VOPAMP                         3, 5          VALUE = {V(in+,in-)*1e5}
```

This has the undesirable property that there is no limit on the output. A better expression is:

```
VOPAMP                         3, 5          VALUE =
+ {LIMIT(V(in+,in-)*1e5,15v,-15v}
```

where the output is limited to +/- 15 volts.

Transient Analysis

The transient analysis starts using a known solution - the bias point. It then uses the most recent solution as the first guess for each new time point. If necessary, the time step is cut back to keep the new time point close enough that the first guess allows the Newton-Raphson repeating series to converge. The time step is also adjusted to keep the integration of charges and fluxes accurate enough.

In theory the same considerations which were noted for the bias point calculation apply to the transient analysis. However, in practice they show up during the bias point calculation first and, hence, are corrected before a transient analysis is run.

The transient analysis can fail to complete if the time step gets too small. This can have two different effects:

1. The Newton-Raphson iterations would not converge even for the smallest time step size, or
2. Something in the circuit is moving faster than can be accommodated by the minimum step size.

The message PSpice puts into the output file specifies which condition occurred.

Skipping the Bias Point

The SKIPBP option for the transient analysis skips the bias point calculation. In this case the transient analysis has no known solution to start from and, therefore, is not assured of converging at the first time point. Because of this, its use is not recommended. It inclusion in PSpice is to maintain compatibility with UC Berkeley SPICE. SKIPBP has the same meaning as UIC in Berkeley SPICE. UIC is not needed in order to specify initial conditions.

The Dynamic Range of TIME

TIME, the simulation time during transient analysis, is a double precision variable which gives it about 15 digits of accuracy. The

dynamic range is set to be 15 digits minus the number of digits of accuracy required by RELTOL. For a default value of RELTOL = .001 (.1% or 3 digits) this gives 15-3 = 12 digits. This means that the minimum time step is the overall run time (TSTOP) divided by 1e12. The dynamic range is large but finite.

It is possible to exceed this dynamic range in some circuits. Consider, for example, a timer circuit which charges up a 100uF capacitor to provide a delay of 100 seconds. At a certain threshold a comparator turns on a power MOSFET. The overall simulation time is 100 seconds. For default RELTOL this gives us a minimum time step of 100 picoseconds. If the comparator and other circuitry has portions that switch in a nanosecond then PSpice needs steps of less than 100 picoseconds to calculate the transition accurately.

Failure at the First Time Step

If the transient analysis fails at the first time point then usually there is an unreasonably large capacitor or inductor. Usually this is due to a typographical error. Consider the following capacitor:

```
C                       1      3, 0   1Ouf
```

“1O” (has the letter O) should have been “10.” This capacitor has a value of one farad, not 10 microfarads. An easy way to catch these is to use the LIST option (on the .OPTIONS command).

LIST

The LIST option can echo back all the devices into the output file *that have their values in scientific notation*.

That makes it easy to spot any unusual values. This kind of problem does not show up during the bias point calculation because capacitors and inductors do not participate in the bias point.

Similar comments apply to the parasitic capacitance parameters in transistor (and diode) models. These are normally echoed to the output file (the NOMOD option suppresses the echo but the default is to echo). As in the LIST output, the model parameters are echoed in scientific notation making it easy to spot unusual values. A further diagnostic is to ask for the detailed operating bias point (.TRAN/OP) information.

.TRAN/OP

This lists the small-signal parameters for each semiconductor device including the calculated parasitic capacitances.

Parasitic Capacitances

It is important that switching times be nonzero. This is assured if devices have parasitic capacitances. The semiconductor model libraries in PSpice have such capacitances. If switches and/or controlled sources are used, then care should be taken to assure that no sections of circuitry can try to switch in zero time. In practice this means that if any positive feedback loops exist (such as a Schmidt trigger built out of switches) then such loops should include capacitances.

Another way of saying all this is that during transient analysis the circuit equations must be continuous over time (just as during the bias point calculation the equations must be continuous with the power supply level).

Inductors and Transformers

While the impedance of capacitors gets lower at high frequencies (and small time steps) the impedance of inductors gets higher.

Note *The inductors in PSpice have an infinite bandwidth.*

Real inductors have a finite bandwidth due to eddy current losses and/or skin effect. At high frequencies the effective inductance drops. Another way to say this is that physical inductors have a frequency at which their Q begins to roll off. The inductors in PSpice have no such limit. This can lead to very fast spikes as transistors (and diodes) connected to inductors turn on and off. The fast spikes, in turn, can force PSpice to take unrealistically small time steps.

Comments

It is recommended that all inductors have a parallel resistor (series resistance is good for modeling DC effects but does not limit the inductor's bandwidth).

The parallel resistor gives a good model for eddy current loss and limits the bandwidth of the inductor. The size of resistor should be set to be equal to the inductor's impedance at the frequency at which its Q begins to roll off. For example, a common one millihenry iron core inductor begins to roll off at no less than 100KHz. A good resistor value to use in parallel is then R = 2*π*100e3*.001 = 628 ohms. Below the roll-off frequency the inductor dominates; above it the resistor does. This keeps the width of spikes from becoming unreasonably narrow.

Bipolar Transistors Substrate Junction

The UC Berkeley SPICE contains an unfortunate convention for the substrate node of bipolar transistors. The collector-substrate p-n junction has *no DC component*. If the capacitance model parameters are specified (e.g., CJS) then the junction has (voltage-dependent) capacitance but no DC current. This can lead to a sneaky problem: if the junction is inadvertently forward-biased it can create a very large capacitance. The capacitance goes as a power of the junction voltage. Normal junctions cannot sustain much forward voltage because a large current flows. The collector-substrate junction is an exception because it has no DC current.

If this happens it usually shows up at the first time step. It can be spotted turning on the detailed operating point information (.TRAN/OP) and looking at the calculated value of CJS for bipolar transistors. The whole problem can be prevented by using the PSpice model parameter ISS. This parameter “turns on” DC current for the substrate junction.

Diagnostics

If PSpice encounters a convergence problem it inserts into the output file a message that looks like the following.

```
ERROR -- Convergence problem in transient analysis at Time =  7.920E-03
       Time step =  47.69E-15, minimum allowable step size =  300.0E-15
These voltages failed to converge:
V(x2.23)  =         1230.23 / -68.4137
V(x2.25)  =        -1211.94 / 86.6888
These supply currents failed to converge:
I(X2.L1)   =        -36.6259 / 2.25682
I(X2.L2)   =        -36.5838 / 2.29898
These devices failed to converge:
X2.DCR3   X2.DCR4   x2.ktr    X2.Q1     X2.Q2
Last node voltages tried were:
NODE     VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE     NODE     VOLTAGE
(    1)   25.2000  (    3)    4.0000  (    4)     0.0000  (    6)   25.2030
(x2.23) 1230.2000  (X2.24)    9.1441  (x2.25) -1211.9000  (X2.26)  256.9700
(X2.28) -206.6100  (X2.29)   75.4870  (X2.30)   -25.0780  (X2.31)   26.2810
(X3.34) 1.771E-06  (X3.35)    1.0881  (X3.36)      .4279  (X2.XU1.6) 1.2636
```

The message always includes the banner (`ERROR -- convergence problem` ...) and the trailer (`Last node voltages tried were` ...). It cannot include all three of the middle blocks.

The "`Last node voltages tried...`" trailer shows the voltages tried at the last Newton-Raphson iteration. If any of the nodes have unreasonable large values this is a clue that these nodes are related to the problem. "These voltages failed to converge" lists the specific nodes which did not settle onto consistent values. It also shows their values for the last two iterations. "These supply currents failed converge" does the same for currents through voltage sources and inductors. If any of the listed numbers are +/- 1e10 then that is an indication that the value is being clipped from an unreasonable value. Finally, "These devices failed to converge" shows devices whose terminal currents or core fluxes did not settle onto consistent values.

The message gives a clue as to the part of the circuit which is causing the problem. Looking at those devices and/or nodes for the problems discussed above is recommended.

PSpice vs. SPICE

7

Overview

A discussion on the differences between PSpice and SPICE is in this chapter.

Comparison Between PSpice and SPICE

PSpice is a member of the SPICE family of circuit simulators. The programs in this family come from the SPICE2 circuit simulation program developed at the University of California at Berkeley during the early 1970's. The algorithms of SPICE2 were developed into more powerful and faster algorithms than those of the earlier versions. The general use and speed of SPICE2 led to its becoming the *de facto* standard for analog circuit simulation. PSpice uses the same general algorithms as SPICE2 and also conforms to its format for input and output files. For more information on SPICE2, see the references listed here, especially the thesis by Laurence Nagel.

[1]L. W. Nagel, *SPICE2: A Computer Program to Simulate Semiconductor Circuits*, Memorandum No. M520, May 1975.

[2]Ellis Cohen, *Program Reference for SPICE2*, Memorandum No. M592, June 1976.

PSpice, the first SPICE-based simulator available on the IBM-PC, started delivering in January of 1984.

Convergence and performance is what sets PSpice apart from all the others in the SPICE family. Many SPICE programs became available on the IBM-PC in mid-1985, after Microsoft released their FORTRAN compiler version 3.0. Most of these SPICE programs are modified little from the U.C. Berkeley code. In the area of convergence, PSpice has a two-year lead in improving convergence and a *customer base that is larger than all of the other SPICE simulators combined* (including those SPICEs offered for workstations and mainframes). This larger customer base provides more feedback and faster response, than any other SPICE program is likely to receive.

PSpice Differences

The version of SPICE2 referred to is SPICE2G.6 from the University of California at Berkeley.

PSpice runs any circuit which SPICE2 can run with these exceptions:

1. Circuits which use .DISTO (small-signal distortion) analysis. U.C. Berkeley SPICE supports the .DISTO analysis but contains errors. Also, the special distortion output variables (e.g., HD2 and DIM3) are not available. Instead of the .DISTO analysis we recommend running a transient analysis and looking at the output spectrum using the Fourier transform mode of *Probe*. This technique shows the distortion (spectral) products for both small-signal and large-signal distortion.
2. These options on the .OPTIONS statement are not available in PSpice:
   - LIMTIM: it is assumed to be 0
   - LVLCOD: no in-line machine code is generated
   - METHOD: a combination of trapezoidal and gear integration is always used
   - MAXORD: a combination of trapezoidal and gear integration is always used
   - LVLTIM: truncation error time step control is always used
   - ITL3: truncation error time step control is always used
3. The IN= option on the .WIDTH statement is not available. PSpice always reads the entire input file regardless of how long the input lines are.
4. Voltage coefficients for capacitors, and current coefficients for inductors must be put into a .MODEL statement instead of on the device statement.
5. PSpice does not allow the use of nested subcircuit definitions.

If this construct is used:

```
.SUBCKT ABC 1 2 3
...
        .SUBCKT DEF 4 5 6
        ...
        .ENDS
...
.ENDS
```

It is recommended that the definitions be separated into:

```
.SUBCKT ABC 1 2 3
...
X1 ... DEF
...
.ENDS

.SUBCKT DEF 4 5 6
...
.ENDS
```

Note *Subcircuit calls could be nested.*

6 The .ALTER command is not supported in PSpice. Use instead the .STEP command to modify specific parameters over multiple PSpice runs.

7 The syntax for the *one-dimensional* POLY form of E, F, G, and H devices is different. PSpice requires a dimension specification of the form POLY(1) while SPICE does not.

PSpice produces basically the same results as SPICE. There can be some small differences, especially for values crossing zero, due to the corrections made for convergence problems. The semiconductor device models are the same as in SPICE.

Using MicroSim Products on Unix

8

Overview

The following topics are covered:

Building Model Files Using Parts on page 8-2 discusses how to create new devices using the Parts utility program.

Stimulus Generation on page 8-16 leads you through the steps to generate stimuli for your circuit designs using the Stimulus Editor.

Stimulus Editor (StmEd) on page 8-21 describes the toolbar buttons and menu functions of the Stimulus Editor.

Building Model Files Using Parts

Overview

As already described in earlier chapters, PSpice is shipped with the Analog Library and the Digital Library containing device model and subcircuit definitions for thousands of off-the-shelf analog and digital parts. For most users, the parts described in these libraries should sufficiently meet their design goals. These pre-defined parts can be easily referenced by part name in the circuit file as long as the library files containing them are declared in .LIB statements.

For some users, however, it is necessary to define new parts. New parts can be manually created within your circuit file using the .MODEL and .SUBCKT/.ENDS commands. Better yet, the new part definitions can be placed in *user-defined library files* for reuse in many circuit files. As in the Analog Library and the Digital Library, a .LIB statement must be included in your circuit file identifying your user-defined library file.

The Parts utility program is available to ease the creation of new *analog* devices. Using Parts, .MODEL, and .SUBCKT definitions are automatically generated to external files by simply entering data sheet values for the wanted components. These can be manually incorporated into user-defined library files. This chapter focuses on using Parts to characterize selected analog device classes and to generate the model and subcircuit definitions.

Note *When a new part is generated (or an existing part is modified) the new data is stored in a file which is given the same name as the part. The name is case sensitive, and the characters in the file name must be used exactly as the name was originally entered. If the device is later accessed through Parts, the name must be entered in exactly the same way; otherwise a new file is created.*

Analog Device Creation using the Parts Program

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

The question could be asked, "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 guarantees when a part is bought. The device's operating characteristics can 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 show how most of the devices operate. Though Parts can work using measurements taken from a specific device, it is sufficient to use typical values from the data sheet for most of your simulation work. A best/worst case model can be created for checking your design.

Device Specification Screen

Once the type of device model is selected, a series of interactive screens are presented such as the one shown in the following figure for opamps.

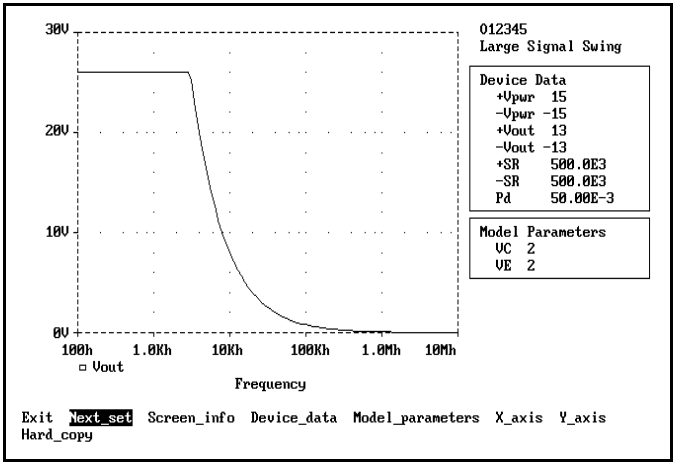


Figure 8-1 *Opamp specification screen for large signal swing*

Each screen describes a different device characteristic in terms of its data sheet information and its model parameters. In this case, large signal swing is depicted for opamp O12345. The data sheet information relevant to this characteristic is displayed in the upper list on the right. Relevant model parameters are displayed in the lower list on the right.

Data sheet information can be entered or modified for each characteristic (screen). For opamps, this information is described as Device Data. By selecting the Device_data menu item at the bottom of the screen, any or all of the data sheet parameters can be altered. (For other device types, data sheet information is presented as a Device Curve and can be modified by choosing the Device_curve menu item.) Once complete, the model parameters in the lower list are automatically updated, as is the graph.

Alternatively, model parameters can be altered by choosing the Model_parameters menu item. Changes are automatically reflected in the graph, only. The *estimated* model parameters can be recovered by re-entering one of the items in the upper list.

Figure 8-2 depicts the two ways in which model parameters can be modified and hence, update the graph of the device characteristic. Data sheet information can be entered and model parameters automatically calculated. Or, model parameters can be explicitly modified.

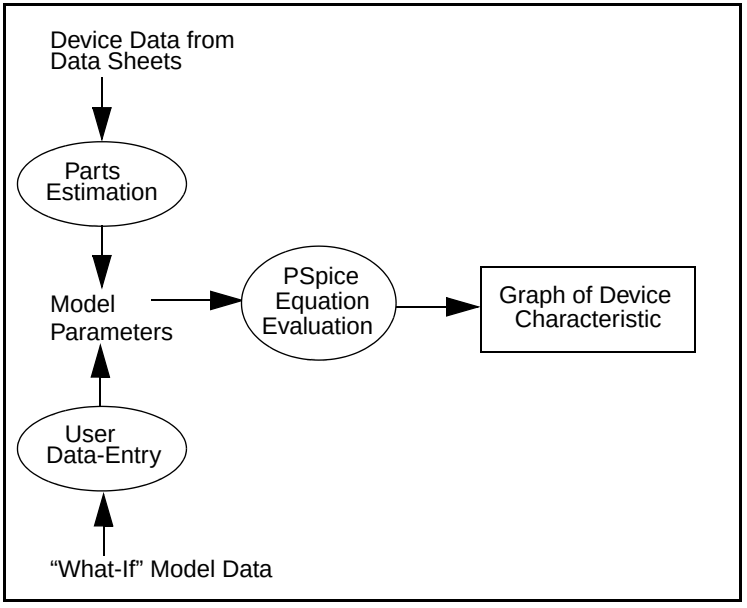


Figure 8-2 *Process and data flow in Parts*

Though *what-if* results can be explored by either means, it is customary to vary the data sheet information list to obtain new model parameters.

The device characteristic *screens* are presented in an order designed to model the most independent model parameters first. Later parameters depend on earlier ones, for example, the forward beta of a bipolar transistor affects its saturation voltage. In opamps, the parameters are modeled first for large signal swing, followed by open loop gain, then by open loop phase, and finally by the maximum output swing.

PSpice Model Files (.MOD)

Each .MODEL and .SUBCKT definition generated by Parts is output to a unique PSpice model file. The PSpice model file name is nominally composed of the part name using the .mod

extension. For devices intrinsic to PSpice (i.e., devices using template models built into PSpice), the file contains a .MODEL statement using all of the parameters generated by Parts in the correct format. The first letter in file name must be the letter representation of the device type. For example, the file for part 2N3904 is named `q2n3904.mod`, where `q` is the SPICE convention for bipolar transistors.

For more complex models, the file contains a .SUBCKT statement (subcircuit definition) using several lines of circuit description and model information. The nominal file name is used for subcircuits. If, in a single Parts session, the parameters for the 1N914 diode are extracted, the 2N3306 transistor, and the OP27 opamp, three files are generated named `d1n914.mod`, `q2n3306.mod`, and `op27.mod`. Note that the first two files have names starting with `d` and `q` respectively, that reflects the intrinsic device type for the model definition. The `op27.mod` file contains a subcircuit definition.

All of these files are saved to your default directory. The files are ordinary text files, which can be copied or changed using a text editor. Using the .LIB command, they can be referenced in your circuit file. However, it is recommended that these files be collected into a larger user-defined library file.

To collect all of the files into one library, use a text editor or refer to your operating system manual for copying several files together into one larger file. For example, the Unix cat (concatenate) command can be used to append the new models into the library file `myparts.lib` as follows:

```
cat *.mod >> myparts.lib
```

Parts Model Data Files (.MDT)

Every time a PSpice model file (.mod) is written to disk, a Parts model data file (.mdt) is also written to disk. This file contains information describing the modeling session, and in effect, *freezes* the session so it can be resumed later. If the same device name is used in a later Parts session, the data found in the .mdt file is automatically read in. Like the .mod file, the name of this file is the device name with a .mdt extension.

Converting Model Files to MDT Format

Conversions are supported for diodes, BJTs, JFETs, and level 3 MOSFETs.

Model files (.mod or .lib) can be converted to the .mdt format. Thus, existing device model definitions can be pulled into Parts in order to

- view performance curves,
- run *what-if* analyses, and
- create new model derivatives for use in simulation.

To run the conversion

1 Start Parts by typing the following at the shell tool prompt:

```
parts
```

This produces the menu list box of part types that can be modeled and an entry to run the conversion.

2 Select the `Model to MDT file converter` entry by typing `A` at the Select prompt.

3 At the prompt, enter the name of the .mod model file to be converted (exclude the .mod extension because it is automatically appended to the file name). This file must reside in the current working directory.

Note *The convertor expects only one device model per .mod file. If multiple model definitions are contained in the file, only the first definition is converted.*

Supported Device Types

As already mentioned, Parts 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. Hence, the .MODEL statement is used.

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. Below is a summary the device model types supported in Parts.

Table 8-1 *Device Models Supported in Parts*

Device Type	Definition Form	Prefix
Diode	.MODEL	"D"
Bipolar Transistor	.MODEL	"Q"
JFET	.MODEL	"J"
Power MOSFET	.MODEL	"M"
Operational Amplifier	.SUBCKT	"X"
Voltage Comparator	.SUBCKT	"X"
Nonlinear Magnetic Core (transformer core)	.MODEL	"K"
Voltage Regulator	.SUBCKT	"X"
Voltage Reference	.SUBCKT	"X"

Running Parts

Setup Requirements

Parts requires the presence of a device file. If no device file is specified, Parts looks for the device file pspice.dev. StmEd also uses this device file. Therefore, if your system is already setup for StmEd, nothing more is needed to set it up for Parts. Otherwise, your device file must be set up manually or use the SetupDev utility.

Refer to the MicroSim installation guide for details on setting up the device file.

Foreground, background, and trace colors can be configured for Parts in the color configuration file pspice.clr. This file is also used by StmEd. If the file is **not** found, a default configuration is used: six trace colors (green, red, blue, yellow, magenta, and cyan), black background, and white foreground.

Refer to the MicroSim installation guide for details on customizing this file.

Manual Start-up

To run Parts, simply type:

```
parts [options]*
```

where

options are the command line options such as command file `(-c <file name>)`, log file `(-l <file name>)`.

Specifying a New Device

This section presents the steps for modeling a simple diode device. To get started, execute the command

```
parts
```

to run the program. Parts is a stand-alone program and does **not** depend on PSpice or Probe for files (though it can share the same device file with other programs).

Below the Parts logo message, there is a menu list box of the devices that Parts can model, as shown in Figure 8-3. All Parts commands and selections are activated through the menu.

Each item in the menu has a number or, if there are more than ten items, a letter preceding the item. After the last menu item, is a suggested command entry. This is the default and is selected if the Enter key is pressed. For this example, pressing `1` or Enter selects the same menu item.

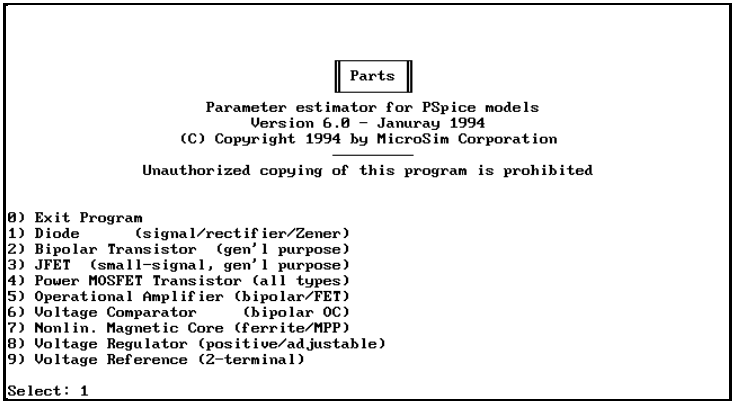


Figure 8-3 *Select device display for Parts*

To model a diode, press `1`. Now a device name is required. This can be the part number, such as 1N4935.

Type `1n4935` followed by Enter to move to the first modeling screen. A screen similar to that of Figure 8-4 appears for modeling the forward current of a diode.

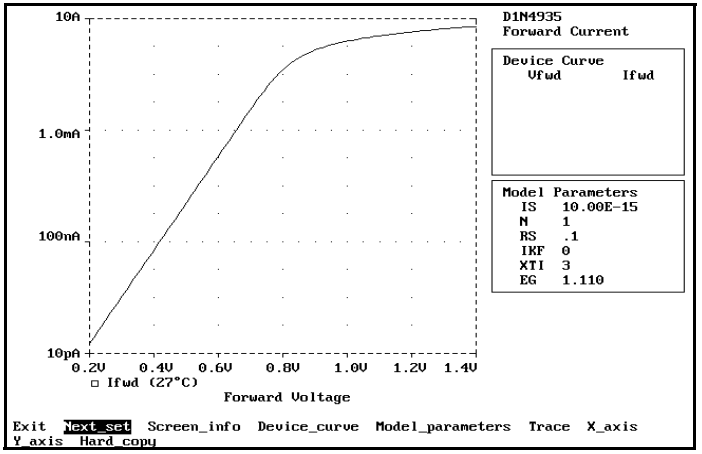


Figure 8-4 *Initial screen for diode model*

When a modeling session begins, the following occurs:

- the screen goes into graphics mode
- a graph for the current device characteristic is displayed
- a command menu is displayed (bottom)
- the device name and screen name are displayed (upper right)
- lists using data sheet information and model parameters are displayed (right side)

The device name is the name that must be used in your PSpice circuit files to refer to this device. It is also the name of the file that Parts creates when this device is fully characterized. In this example, the file is named using the format `d<device name>.mod` where `d` is the PSpice intrinsic device type character for a diode and the `<device name>` is 1N4935. This file has all of the model parameters recorded in it for use by PSpice.

Note *Device names and the associated Parts file names are case sensitive.*

The screen name is also the label for the graph, which is similar to the graphs on a data sheet. Each screen name is unique to a device, and is used as a guide through this tutorial.

A command menu is provided at the bottom of the screen.

To set the X axis to the range .6 to 1.6 volts

Select the command X_axis followed by Set_range, and type `.6,1.6` followed by Enter in the range text box.

To add values for the device curve

The values in this list come from measurements reported on the data sheet. Select the command Device_curve, and immediately start entering data points as pairs of numbers. In this case, enter the forward voltage (Vfwd) first, followed by Enter. Then enter the forward current (Ifwd) followed by Enter. Use the following values:

```
Vfwd = 1
Ifwd = 1
```

After the last Enter, a menu appears which allows changes to the data curve. At this point, select Exit, or press Esc.

The screen changes and is similar to that of Figure 8-5. The lower data list has model parameters using values for the PSpice model being worked on. These parameters are *fitted* whenever the device data is complete enough to do a fit. The model parameters are re-fitted whenever the Fit command is selected, or when a value is changed in the device data. (Parts does **not** keep track of the value that is changed, so entering the same value still activates a fit.)

The graph shows the effect of the changed model parameter values. At this point, the example diode has been given a new value for the model parameter IS only, since there is only one data point.

To add another data point to fit values for both IS and N

Select the command Device_curve, then select Add, and use the following values:

```
Vfwd = .8
Ifwd = .15
```

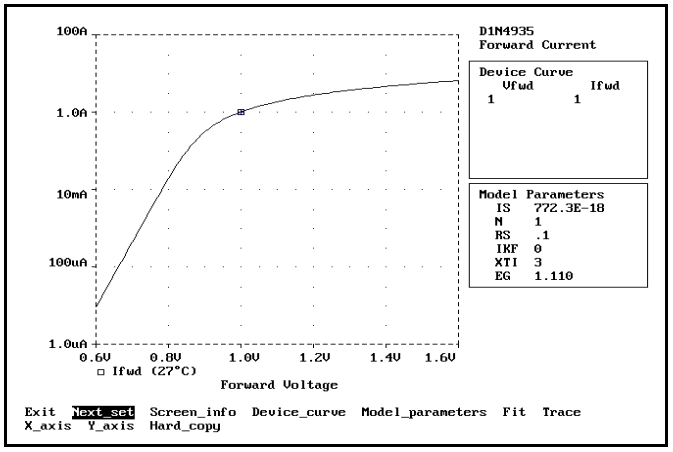


Figure 8-5 *Diode model screen after entering one data point in the device curve*

Notice that the list keeps its values in order. Select Exit or press Esc. The screen changes and is similar to that of Figure 8-7.

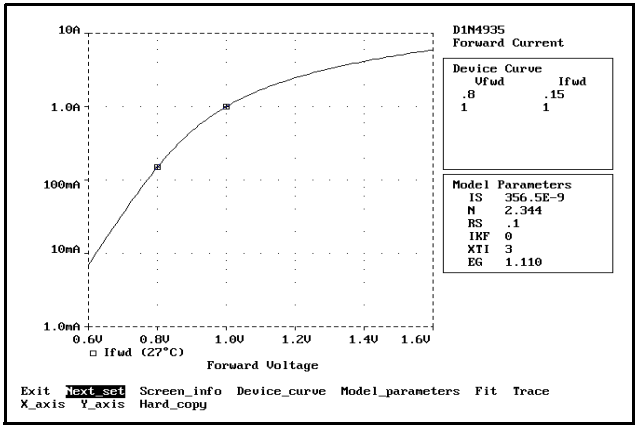


Figure 8-6 *Diode model screen after entering two data points in the device curve*

This time, both IS and N have new values.

To add another data point to fit values for IS, N, and RS

Choose the command Device_curve, then select Add, and use the following values:

```
Vfwd = 1.4
Ifwd = 10
```

Select Exit, or press Esc.

The screen changes once again and is similar to that of Figure 8-7. This time, IS, N, and RS have new values. Look at these values closely, to remember them, and then select the Fit command. The model parameters should change slightly, adjusting to make a better fit. In regular use, selecting the Fit command can continue until the model parameters stop changing.

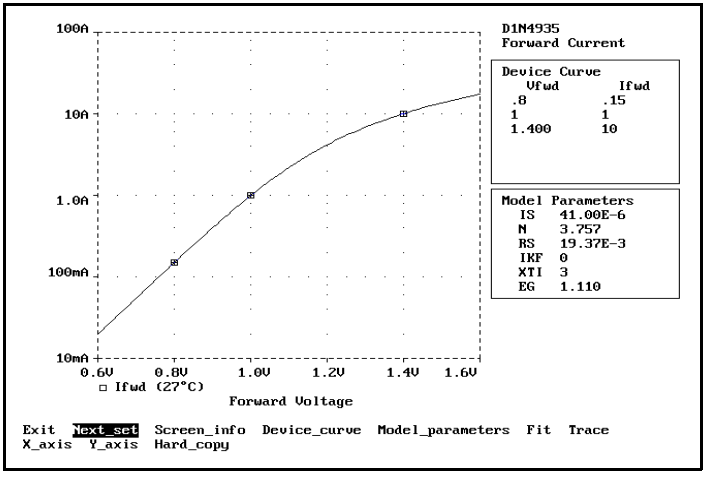


Figure 8-7 *Diode model screen after entering three data points in the device curve*

Choosing Next_set displays the screen for the next device characteristic and its associated parameters. But before trying that, stop Parts by select Exit to get back to the Parts logo screen (some disk activity can be noticed) followed by Exit Program to quit.

Now, use a directory command to look for the PSpice model file just created (`d1n4935.mod`). Print the file to your screen or printer. It should look like this:

```
.model d1n4935 D (Is=39.14u ...
```

Look for the Parts model data file just created (`d1n4935.mdt`). Restart Parts, select Diode, and enter the same diode name, `1n4935`. The first graphics screen should reappear using all of the modeling data from the last time Parts was run for this model. All of the session data came from the Parts model data (.mdt) file. Continue modeling this device on your own.

Stimulus Generation

Overview

Several device types are available for generating stimuli for your circuit.

Table 8-2 *Stimulus Device Summary*

Device Type	Mode	Description
V	Analog	Independent voltage source
I	Analog	Independent current source
U STIM	Digital	Stimulus generator
U FSTIM	Digital	File stimulus device

The analog devices can generate waveforms of these types:

- sine wave
- repeating pulse
- exponential pulse
- single-frequency FM
- piecewise linear

The piecewise linear waveform can approximate any given waveform.

Digital stimuli can produce complex timing relations using repeating segments.

Stimulus devices can be defined manually by editing your circuit file for those device instances (with the exception of the *digital file stimulus device*, which reads input data from a file). The process is partly automated using the interactive stimulus device design aid, the Stimulus Editor (StmEd).

Stimulus Generation using StmEd

StmEd can quickly set up and verify the analog and digital stimulus (input) waveforms for a simulation. Menu prompts provide the necessary parameters as guides, such as the rise time, fall time, and period of an analog repeating pulse, or the complex timing relations with repeating segments of a digital stimulus. Graphical feedback allows for quick verification of the waveform characteristics.

The stimuli and all the other components are saved in the circuit file. Therefore, when many similar simulations are run, it is only necessary to set up the input waveforms once.

Running StmEd

StmEd requires the presence of a device file. If no device file is specified, StmEd looks for the device file `pspice.dev`. The Parts program also uses this file. Therefore, if your `pspice.dev` file is already set up for Parts, nothing more is needed to set it up for StmEd. If not, set up the device file manually or use the SetupDev utility.

Refer to the MicroSim installation guide for details on the SetupDev utility.

Foreground, background, and trace colors can be configured for StmEd in the color configuration file `pspice.clr`. This is the same file used by Parts. If the file is **not** found, the default configuration is used: six trace colors (green, red, blue, yellow, magenta, and cyan), black background, and white foreground

Refer to the MicroSim installation guide for details on customizing this file.

StmEd can be run from the system prompt using the command:

```
stmed [options]* [<circuit file>]
```

where

`options`	are the command line options for running StmEd such as command file (-c <`file name`>), log file (-l <`file name`>)
`circuit file`	is the name of a new or existing circuit file.

StmEd Tutorial for an Analog Stimulus

Use StmEd to edit the stimuli in the circuit file `example1.cir` that is shipped with PSpice. First start StmEd by typing:

```
stmed example1.cir
```

A plot similar to that of Figure 8-8 should display. It shows the transient waveform for the independent voltage source VIN. This plot is the only independent source having a transient behavior specification of SIN(0 0.1 5MEG).

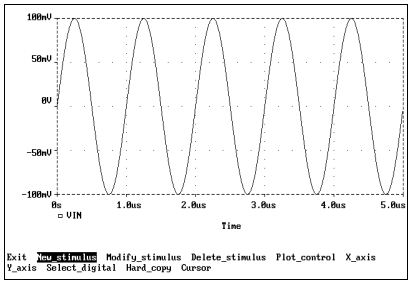


Figure 8-8 *Typical StmEd waveform display*

Note *To add the other stimuli to the plot for modification, use the Plot_control command and then the Display_stimulus command to list VCC and VEE as choices for display and modification.*

To change the transient waveform of VIN, select the Modify_stimulus command from the Main Menu. In the bottom portion of the display, similar to that of Figure 8-9, notice that VIN has a transient specification type of SIN (sinusoidal) and a list of values for each of the parameters that describe the waveform.

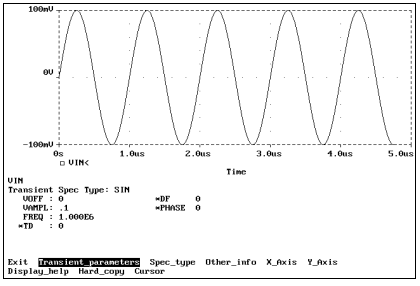


Figure 8-9 *Modifying a sinusoidal waveform*

To change the peak amplitude from .1 volts to .2 volts

1. choose the Transient_parameters command.
2. Select VAMPL from the list of parameter names.
3. Enter .2 and press Enter.
4. Select Exit.

   The waveform should now be redrawn using a peak amplitude of .2 volts.
5. Select Exit again to return to the Main Menu.

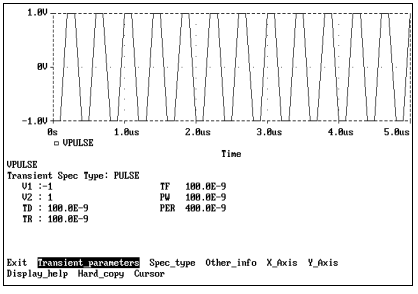


Figure 8-10 *Display of VPULSE from EXAMPLE1*

To create a new stimulus

1. Select the New_stimulus command from the Main Menu.
2. When asked for the name, type `VPULSE` and press Enter.
3. Select PULSE from the Transient Parameter Menu to create a pulsed waveform.

4 Type `-1v` at the prompt to define the waveform and press Enter.

5 Repeat steps 1-4 for `1v`, `.1u`, `.1u`, `.1u`, `.1u`, and `.4u` (the initial voltage, pulsed voltage, delay, rise time, fall time, pulse width, and period, respectively).

To exit the program

Repeatedly select the Exit command until the Exit_program command appears.

Exit_program writes the changes made to VIN and the new stimulus VPULSE into `example1.cir`. Select Abort_program if saving the changes in `example1.cir` is **not** wanted.

StmEd Tutorial for a Digital Stimulus

StmEd can be used to edit the digital stimulus in the circuit file `digital.cir` that is shipped with MicroSim software platforms supporting mixed analog/digital design. Start StmEd by typing:

```
stmed digital.cir
```

This time, plots are presented for both the analog stimulus VIN, and the digital stimulus U2 as shown in Figure 8-11.

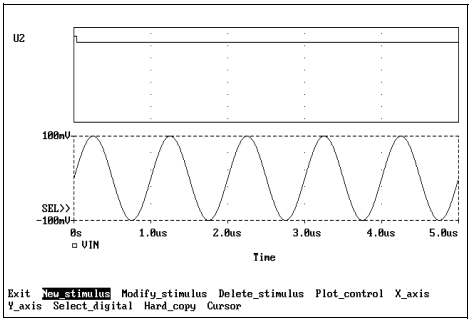


Figure 8-11 *Analog and digital stimuli in digital.cir*

To modify U2

1 Select the Modify_stimulus command from the Main Menu.

2 Mark U2 as the current stimulus to be edited by pointing to its name (to the left of the digital area) using the mouse, and clicking the left mouse button.

A < symbol appears to the right of U2 indicating that it is current. Type Enter to display the edit screen for U2.

Notice that the U2 device is described by digital state value changes at specified time intervals.

3 Select the Add_command item, followed by the Set selection.

4 Type `900.00E-9` for the time value as shown in Figure 8-12.

Then type `1` for the digital state value.

Alternatively, press Tab+Space to highlight U2 and mark it as

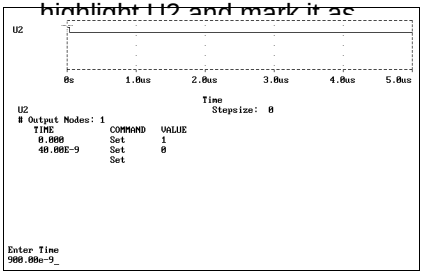


Figure 8-12 *Editing the digital stimulus*

5 To stop adding digital stimulus commands, type Esc followed by selecting the Exit from the menu.

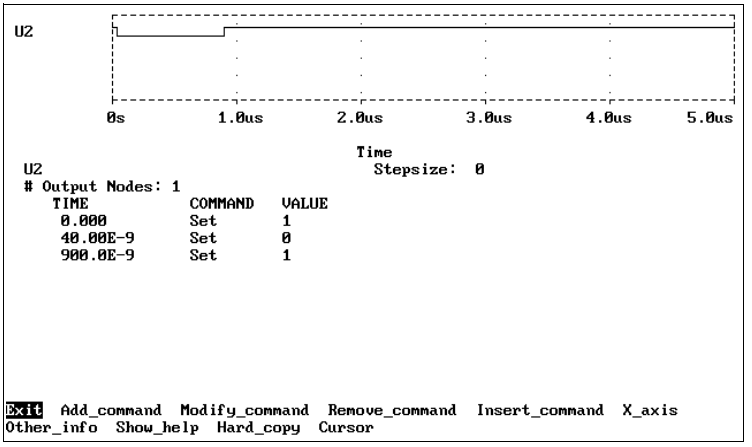


Figure 8-13 *Modified U2 digital stimulus*

Changes can now be saved to `digital.cir` by exiting the program.

Stimulus Editor (StmEd)

Overview

StmEd lets you quickly set up and verify the stimulus (input) waveforms for a simulation. Menu prompts guide you to provide the necessary parameters, such as the rise time, fall time, and period of a repeating pulse. You also have the capability to enter waveforms graphically using a mouse. Graphical feedback lets you quickly verify the waveform. The available analog waveforms are: sine wave, repeating pulse, exponential pulse, single-frequency FM, and piecewise linear. The piecewise linear waveform consists of a series of (<time>,<value>) pairs and can be used to approximate any given waveform.

User Interface

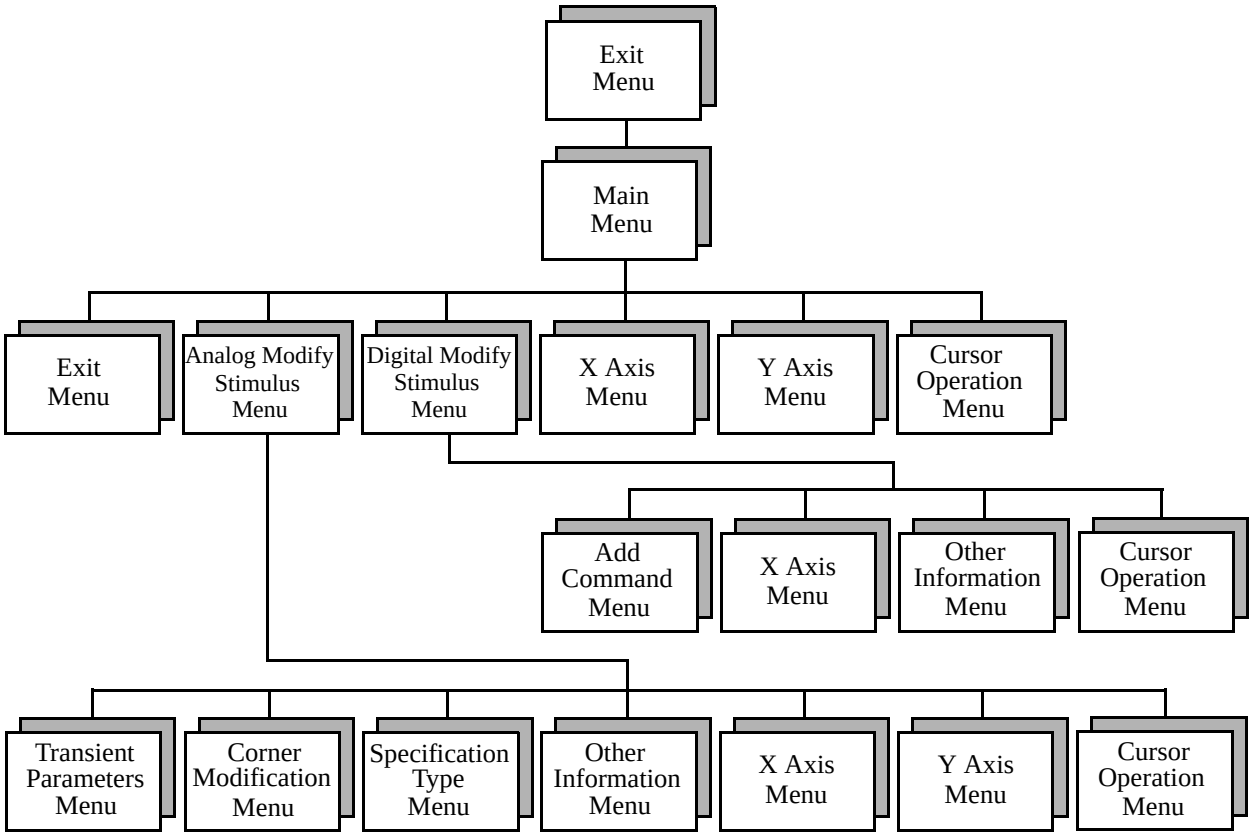
Toolbar Buttons

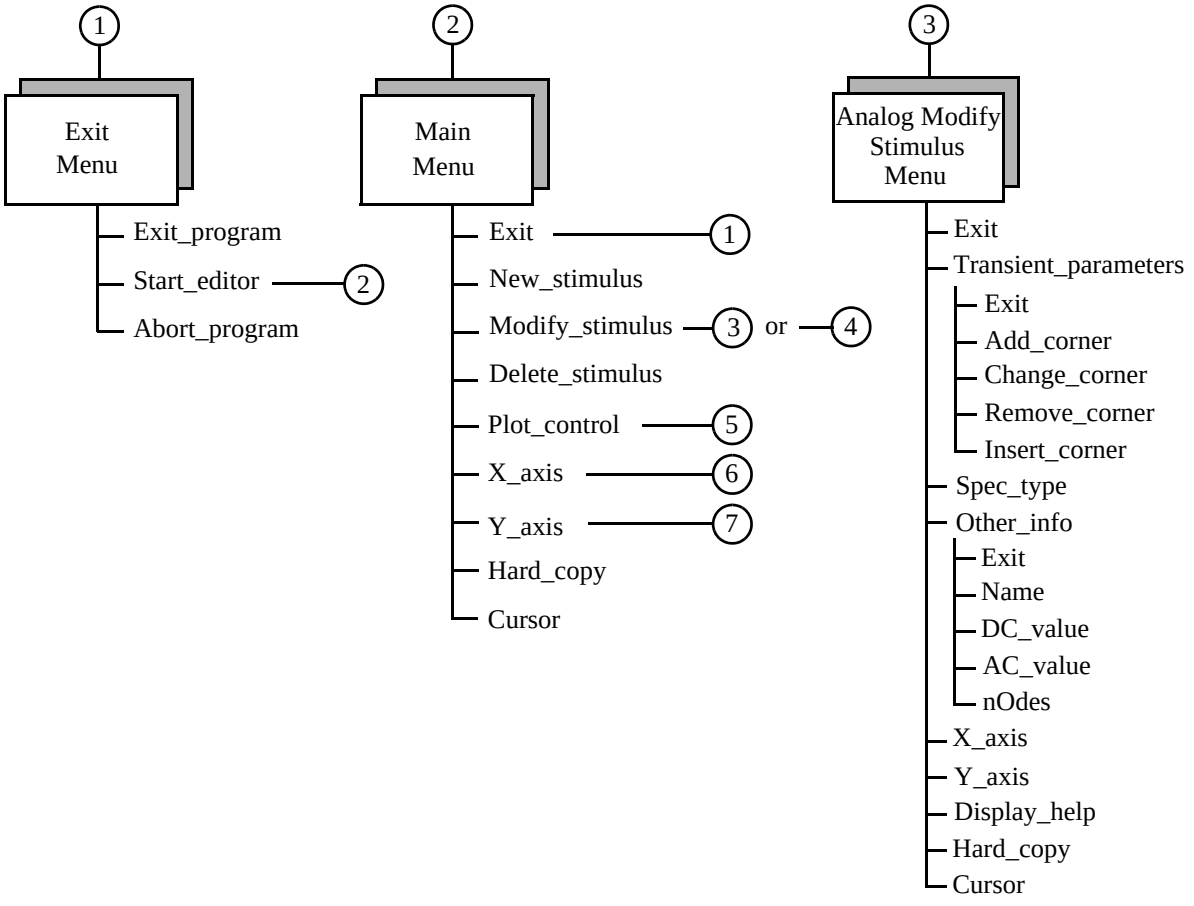
Toolbar buttons provide shortcuts for initiating common stimulus editing functions. Choosing a toolbar button (by a click), invokes the same response as would selecting the corresponding menu item. Below is a list of the StmEd toolbar buttons and the equivalent menu item or action taken in StmEd.

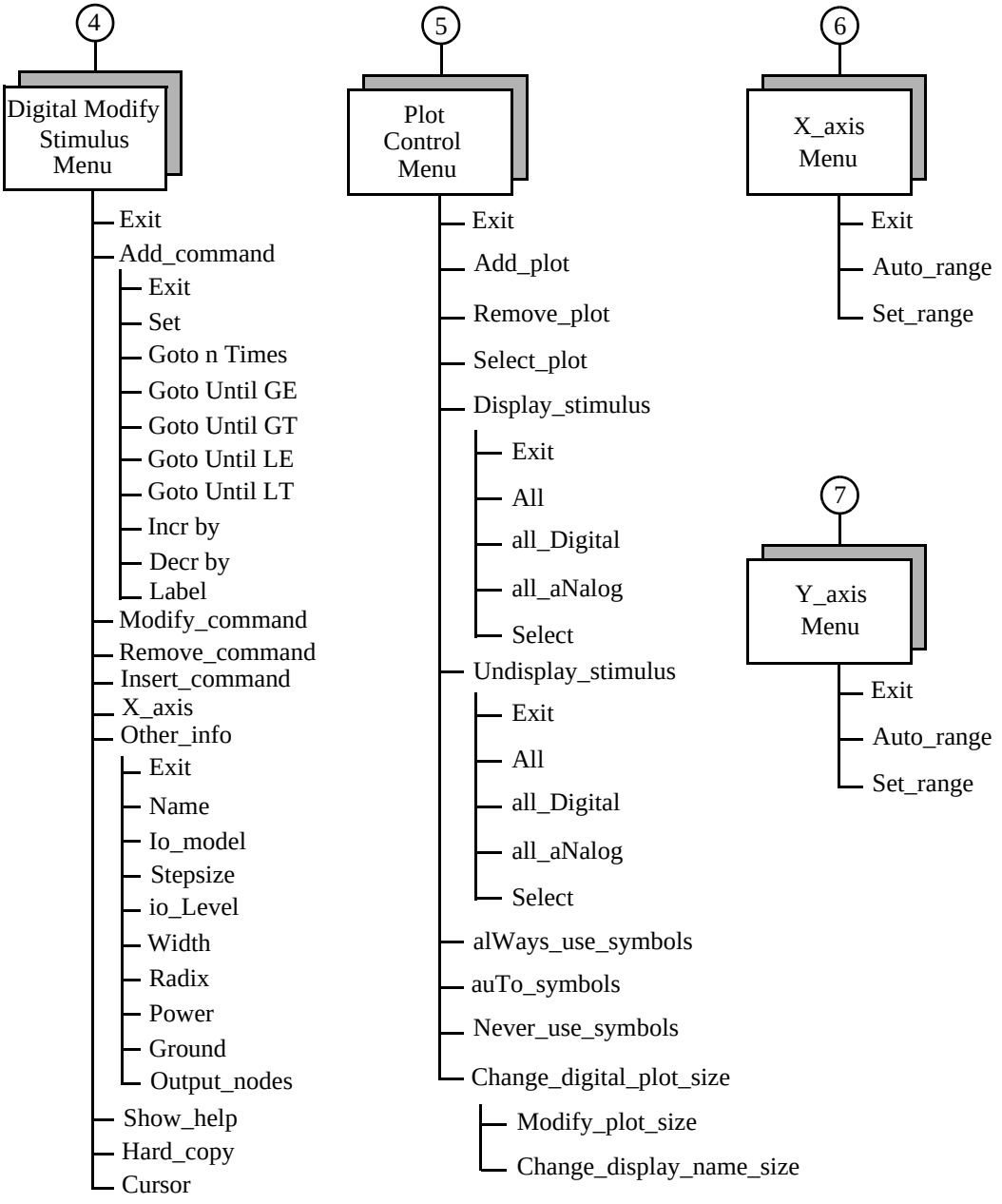
Table 8-3 *Stimulus Editor Toolbar Buttons*

Button	Equivalent Menu Item or Action
	File/New
	File/Open
	File/Save
	Immediate hard copy print
	Stimulus/New Stimulus
	Stimulus/Get Stimulus
	Edit/Attributes
	Edit/Add
	View/In
	View/Out
	View/Area
	View/Fit
	Plot/Axis Setting

Menu Structure




1
Exit Menu
Exit_program
Start_editor
2
Abort_program
2
Main Menu
Exit
1
New_stimulus
Modify_stimulus
3
or
4
Delete_stimulus
Plot_control
5
X_axis
6
Y_axis
7
Hard_copy
Cursor
3
Analog Modify Stimulus Menu
Exit
Transient_parameters
Exit
Add_corner
Change_corner
Remove_corner
Insert_corner
Spec_type
Other_info
Exit
Name
DC_value
AC_value
nOdes
X_axis
Y_axis
Display_help
Hard_copy
Cursor


4
Digital Modify Stimulus Menu
Exit
Add_command
Exit
Set
Goto n Times
Goto Until GE
Goto Until GT
Goto Until LE
Goto Until LT
Incr by
Decr by
Label
Modify_command
Remove_command
Insert_command
X_axis
Other_info
Exit
Name
Io_model
Stepsize
io_Level
Width
Radix
Power
Ground
Output_nodes
Show_help
Hard_copy
Cursor
5
Plot Control Menu
Exit
Add_plot
Remove_plot
Select_plot
Display_stimulus
Exit
All
all_Digital
all_aNalog
Select
Undisplay_stimulus
Exit
All
all_Digital
all_aNalog
Select
alWays_use_symbols
auTo_symbols
Never_use_symbols
Change_digital_plot_size
Modify_plot_size
Change_display_name_size
6
X_axis Menu
Exit
Auto_range
Set_range
7
Y_axis Menu
Exit
Auto_range
Set_range

Menu Function Definition

Exit Menu

This menu appears when you are about to exit StmEd.

Command	Description
Exit_program	Saves any stimuli you have created and any changes you have made into your circuit file and then exits the StmEd program. If you were editing an existing file, a copy of the original unedited file is saved under the same file name with a .BAK extension for backup purposes.
Start_editor	Transfers control to the Main menu.
Abort_program	Exits the StmEd program without saving any changes you have made since starting the program.

Main Menu

This menu appears when you start StmEd. When you are in this menu, one or more plots are displayed on the screen. If there are multiple plots displayed at one time, one plot will be marked as the selected plot by the characters SEL>>. Some commands, such as New_stimulus and Modify_stimulus, only work with one plot at a time. In the case where there are multiple plots displayed at one time, the commands will only effect the selected plot.

If you are editing an existing circuit file, StmEd will read in the file and display the first four stimuli (I- or V-devices) that have a transient specification. Stimuli that are not initially displayed may be added to a plot by using the commands in the Plot Control menu.

Exit

Exits the Main menu and returns to the Exit menu.

New_stimulus

Adds a new stimulus (either an independent current or voltage source, or digital stimulus) to the currently selected plot. You will be prompted to enter a device name. Enter a name beginning with V to add a voltage source, I to add a current source, or U to add a digital stimulus.

Modify_stimulus

Allows you to change an existing stimulus that is displayed on the currently selected plot. After selecting Modify_Stimulus, you will be prompted to select the stimulus to modify. Either the Analog Modify Stimulus menu or Digital Modify Stimulus menu will be displayed, depending on the type of stimulus chosen.

Delete_stimulus

Enters the Delete Stimulus menu, which allows you to delete one or all of the stimuli that are displayed in the currently selected plot. Deleting will also remove the stimuli from the circuit file.

Exit

Exits the Delete Stimulus menu and returns to the Main menu.

All

Removes all analog and digital stimuli from the currently selected plot.

all_Digital

Removes all of the digital stimuli from the currently selected plot.

all_aNalog

Removes all of the analog stimuli from the currently selected plot.

Select

Allows you to select one or more analog and/or digital stimuli to be removed. The stimulus selection process is accomplished by highlighting the desire stimlui and then pressing Enter to remove them.

After selecting Select, the first analog stimulus is highlighted. If only digital stimuli are displayed, then the first digital stimulus is highlighted. If the currently highlighted stimulus is one to be removed, use the Space to mark that stimulus for removal. The highlight may be moved among the analog stimuli using the ← and →, and the digital stimuli using the ↑ and ↓. The ↑ and ↓ may also be used to scroll the digital stimuli, if there are more displayed then can fit on the digital plot. If both analog and digital stimuli exist, the highlight can be moved from the analog to the digital part or from the digital to the analog part by using the Tab key.

You can also use the left mouse button to select the name of the stimulus to be removed.

Undelete

Redisplays the last stimuli deleted since entering the Delete Stimulus menu. Undelete only appears in the menu if traces have been deleted since entering the Delete Stimulus menu.

Plot_control

Displays the Plot Control menu.

X_axis

Displays the X Axis menu. Since all plots on the display share the same X axis, these commands affect every plot.

Y_axis

Displays the Y Axis menu.

Hard_copy

Prints or plots the display onto a hard copy device. The Hard_copy command works the same as in Print in Probe; see the Print command under the File menu in the Probe chapter.

The hard copy devices available are determined by the information in the device file. Refer to the MicroSim installation guide for details.

Cursor

Displays the Cursor Operation menu and activates two cursors which may be attached to one or two traces:

Exit

Exits from the Cursor Operation menu and returns to the Main menu. You can also press Esc to exit.

Hard_copy

Allows you to print the cursor and their values as shown on the display.

To change the trace on which the first cursor is positioned within a plot, press Ctrl → or Ctrl ←. Change the trace for the second cursor with Shift Ctrl → and Shift Ctrl ←.

Move the first cursor along the trace using ← and →. To move to the beginning of a trace, press the Home key; to move to the end, press the End key. To move the second cursor along the trace with ⇧ Shift +Home and ⇧ Shift +End.

On Sun keyboards, use the Home and End keys to move to the beginning or end of the trace. On older Sun keyboards, use <R1> and <R3>.

The mouse can also be used to move the position of the cursor on all platforms. Pressing the left mouse button moves the first cursor to the closest X position on the selected trace. The right mouse button performs the same function with the second cursor. If the trace has multiple Y values for a given X value, the cursor moves to the Y value closest to the cursor position.

The mouse can also be used to select the trace to be used with that cursor by clicking on the trace symbol in the legend.

The X,Y value for each cursor is continuously displayed, as well as the difference of each as they are moved.

Analog Modify Stimulus Menu

This menu appears when the Modify_stimulus command is chosen from the Main menu, and a voltage or current source is chosen. The stimulus name and the current values of the

transient waveform specification for the chosen stimulus will be shown at the bottom of the display. The top of the display will contain a plot which shows the waveform.

Exit

Exits the Analog Modify Stimulus menu and returns to the Main menu.

Transient_parameters

For non-piecewise linear specifications, a menu of the available parameters is shown. You may change the value of a parameter by choosing that item from the menu and entering a new value. Numeric values are input in the same form for PSpice/PLogic, except that suffixes MEG and MIL are not available. This exception is handled with the m/ M scale suffix for numbers: m means milli (1E-3) whereas M means mega (1E+6).

Note *Note: The waveform display is re-drawn only after you exit the Transient Parameters menu.*

Values may also be expressions if enclosed by { and }. If an existing circuit file is being edited which contains.PARAM parameters, these parameters may be used in expressions. The value of the expression is displayed when the expression is being used, and the expression itself is displayed when you go to modify the value of the expression.

For piecewise linear waveforms, the list of corners that define the waveform are shown. A corner is a pair of values

```
(<time>, <value>)
```

where `<time>` is the time at which the waveform will be equal to `<value>`. A + in front of the topmost corner indicates that there are previous corners that are not being displayed. If the bottommost corner has a + in front of it, there are more corners beyond this one that are not being displayed. To see the undisplayed corners, use the Change_corner command. The Corner Modification menu will be displayed with the following commands.

Exit

Returns to the Analog Modify Stimulus menu.

Add_corner

Adds one or more corners to the list of corners. When you have entered all the corners you want, press Enter or Esc to return to the Corner Modification menu.

Change_corner

Lets you change the time and/or value of a corner. You will be prompted to select a corner to change. Use the ↑ and ↓ keys to highlight the corner you wish to change or use the left mouse button. Press Enter to change the time and/or value of the highlighted corner, or Esc to return to the Corner Modification menu without changing it.

The key combinations Ctrl Page Up and Ctrl PageDown move to the start and end of the list.

Remove_corner

Deletes one or more corners from the list. You will be prompted to select a corner to remove. Use the ↑ and ↓ keys until the corner you wish to delete is highlighted or use the left mouse button. Press Enter to delete the highlighted corner or Esc to return to the Corner Modification menu.

The key combinations Ctrl Page Up and Ctrl PageDown move to the start and end of the list.

Insert_corner

Adds one or more corners anywhere in the list. You will be prompted to select a corner above which you wish to add the new corner(s). Use the ↑ and ↓ keys to highlight the desired corner or use the left mouse button. Press Enter to insert the new corner(s) at the current location, or Esc to return to the Corner Modification menu without making any modifications.

The key combinations Ctrl Page Up and Ctrl PageDown move to the start and end of the list.

Either the mouse or the keyboard can be used to add or insert corners for piecewise linear specifications. If a mouse is present, whenever you are prompted for the time and value of a corner, a cross hair will appear near the center of the plot. The time and value at the cross hair will be shown at the bottom of the screen. Whenever the mouse is moved, the cross hair will move, and the current time and voltage (or current) values will be updated. To add a corner with the mouse, click the left mouse button. To scroll the plot, move the mouse just outside the plot and click the left mouse button.

Spec_type

Changes the type of transient waveform the stimulus will produce. The Specification Type menu showing the available types is displayed.

Other_info

Enters the Other Information menu and displays the non-transient parameter information associated with an I- or V-device.

Exit

Exits the Other Information menu and returns to the Analog Modify Stimulus menu.

Name

Lets you change the name of the stimulus device.

DC_value

Lets you set the DC value for the stimulus device.

AC_value

Lets you set the AC value(s) for the stimulus device.

nOdes

Allows you to set the + and - node names for the stimulus device.

X_axis

Enters the X Axis menu.

Y_axis

Enters the Y Axis menu.

Display_help

Displays the definition of each of the parameters for the transient specification being shown.

Hard_copy

PRINTS or plots the display onto a hard copy device.

Cursor

See the Cursor command in the Main menu section.

Digital Modify Stimulus Menu

This menu appears after the Modify_stimulus command is chosen from the Main menu, and a digital stimulus is chosen. The name and transient specification for the digital stimulus that was chosen are shown in the lower portion of the display. The corresponding waveform is shown in the plot at the top of the display.

The transient specification for a digital stimulus consists of a series of commands that describe the output of the stimulus over time. If the topmost command is preceded by a +), there are more commands prior to it that are not being displayed. If the bottommost command is preceded by a +, there are more commands following that are not being displayed. To see commands that are not being displayed, use the Modify_command command.

Exit

Returns to the Main menu

Add_command

Enters the Add Command menu allowing a stimulus to be defined.

The following commands are available:

Exit

Returns to the Digital Modify Stimulus menu.

Set

See the description of radix under the Other_info command for more information.

Prompts you to enter a time and a value. At the given time, the output of the stimulus changes to value. The format of value depends on the radix of the stimulus. When you have entered all the desired SET commands, press Esc to return to the Add Command menu.

Goto n Times

Prompts you to enter a time, a label name, and the number of times you want to repeat the loop. When the goto is encountered, the statement following the <label name> is immediately executed.

Goto until GE | GT | LT | LE

Prompts you to enter a time, a label name, and a <value>. The goto is done until the value of the stimulus is GE (greater than or equal to), GT (greater than), LT (less than), or LE (less than or equal to) the value specified.

This command only appears if the stimulus has more than one output node

Incr by

Prompts you for a <time> and a <value>. The output of the stimulus is incremented by <value> each time this command is executed.

This command only appears if the stimulus has more than one output node

Decr by

Prompts you for a <time> and a <value>. The output of the stimulus is decremented by <value> each time this command is executed.

This command only appears if the stimulus has more than one output node

Label

Prompts you for a label name. This allows you to define the labels that are used in “goto” commands.

Modify_command

Allows you to select one of the commands displayed and change it. Use the ↑ and ↓ keys to highlight the command you wish to modify or select it with the left mouse button. (The Ctrl Page Up and Ctrl PageDown key combinations move to the start and end of the list.) Press Enter to change the command or Esc to return to the Digital Modify Stimulus menu without changing the command.

The Ctrl Page Up and Ctrl PageDown key combinations move to the start and end of the list.

Remove_command

Allows you to delete one or more of the commands. Use the ↑ and ↓ keys to highlight the command you wish to remove or select it with the left mouse button. Press Enter to remove the command or Esc to return to the Digital Modify Stimulus menu without removing the command.

Insert_command

Allows you to add a command somewhere other than at the bottom of the list. Use the ↑ and ↓ keys to highlight the command above which you wish to insert one or more commands or select it with the left mouse button. (The Ctrl Page Up and Ctrl PageDown key combinations move to the start and end of the list.) Press J to insert commands or E to return to the Digital Modify Stimulus menu.

X_axis

Displays the X Axis menu. Since all plots on the display share the same X axis, these commands affect every plot.

Other_info

Displays the Other Information menu which displays the non-transient parameter information associated with a digital stimulus.

The available commands are as follows:

Exit

Exits the Other Information menu and returns to the Digital Modify Stimulus menu.

Name

Is the device name which must start with a U to indicate that this is a digital stimulus.

Io_model

The name of the model which describes the device's loading and driving characteristics.

Stepsize

The number of seconds that a step represents. If you suffix the time for a command with a C, it is interpreted as steps, and the actual time is equal to the number of steps multiplied by the stepsize.

Width

The number of output nodes connected to the stimulus.

Radix

Specifies the value format. It is a sequence of digits which specifies the number of signals that the corresponding digit in a value represents. Each digit of value is assumed to be in base 2n, where n is the corresponding digit in the radix. The sum of the digits in the radix must be equal to the width. Each digit must be either a 1, 3, or 4 (that is, binary, octal, or hexadecimal).

Power

The digital power node, with a default value of $G_DPWR.

Ground

The digital ground node, with a default value of $G_DGND.

Output_nodes

Displays the list of digital nodes that are connected to the stimulus. For stimuli that have more than one output node (a bus), the first node name entered is associated with the high-order bit, and the last node name is associated with the low-order bit.

Show_help

Displays an explanation of the available commands.

Hard_copy

Prints or plots the display onto a hard copy device. The hard copy devices available are determined by the information in the device file.

Cursor

See the Cursor command in the Main menu section.

Plot Control Menu

Exit

Exits the Plot Control menu and returns to the Main menu.

Add_plot

Adds an analog plot to the display, and that plot then becomes the currently selected plot. The display can contain up to 20 plots. If the display is already at its maximum capacity, this command does not appear. The new plot is added at the top of the display, condensing the existing plots to make room. All plots share the same X axis, but each has its own Y axis.

Remove_plot

Removes the selected plot from the display. If the display has only one plot, this command does not appear. After the plot is removed, the other plots are expanded to fill the display and the topmost plot becomes the new selected plot.

Select_plot

Allows you to change which plot is selected. If the display has only one plot, this command does not appear. After you choose this command, you use the up and down row keys or mouse to move the select characters SEL>>. When the SEL>> characters point to the desired plot, select Exit to return to the Plot Control menu.

Display_stimulus

Allows you to add one or all existing stimuli to the currently selected plot.

Undisplay_stimulus

Allows you to remove one or all stimuli from the currently selected plot without deleting it from the circuit file.

always_Use_symbols

Enables the drawing of trace symbols. Each trace is marked with a symbol matching that of the associated variable.

auTo_symbols

Allows StmEd to draw symbols only when the following criteria apply:

- More traces are displayed then there are colors for drawing traces
- There are fewer than nine traces. StmEd starts with auTo_symbols as the default.

Never_use_symbols

Disables the drawing of trace symbols by the auTo_symbols selection.

Change_digital_plot_size

Displays the Change Digital Plot Size menu.

Modify_plot_size

Allows you to specify the percentage of the plot that should be for digital traces. You will be prompted to select a percentage within the displayed valid range. The default is 33 percent.

Change_display_name_size

Allows you to specify the length of the digital signal’s display name.

X Axis Menu

This menu appears after the X_axis command is selected from either the Main menu, Analog Modify Stimulus menu, or Digital Modify Stimulus menu.

Exit

Returns to the previous menu (Main menu, Analog Modify Stimulus menu, or Digital Modify Stimulus menu).

Auto_range

Sets the range of the axis to be the range of its stimuli, rounded to a *nice* value. The range of the axis will be automatically adjusted as its stimuli are changed. If the axis is already auto-ranged, this command does not appear.

If an existing file with a .TRAN statement is being edited, the range will use the final time value of the .TRAN statement.

Set_range

Sets the range of the axis to your specification. Once set, the range is not affected by changes in the stimuli being displayed. It can only be changed by an Auto_range or another Set_range command.

Y Axis Menu

This menu appears after the Y_axis command is chosen from either the Main menu or the Analog Modify Stimulus menu.

Exit

Returns to the previous menu (Main menu or the Analog Modify Stimulus menu).

Auto_range

Sets the range of the axis to be the range of its stimuli, rounded to a *nice* value. The range of the axis will be automatically adjusted as its stimuli are changed. If the axis is already auto-ranged, this command does not appear.

Set_range

Sets the range of the axis to your specification. Once set, the range is not affected by changes in the stimuli being displayed. It can only be changed by an Auto_range or another Set_range command.

Glossary

AKO "A Kind Of" symbol. Symbols must either contain graphics or refer to an AKO symbol. The AKO defines the symbol in terms of the graphics and pins of another part. Both must exist in the same Symbol Library file.

alias An alias relates local schematic names for parts and signals to netlist names (simulation devices and nodes). An alias is an exact electrical equivalent that can be used to reference a symbol. A command that sets up equivalences between pin names or net names and node names. As a command, it is the setup equivalences between node names and pin names or net names.

annotation Annotation is a means by which parts are labeled when they are placed, either automatically or manually.

annotation symbol An annotation symbol has no electrical significance, and is used to clarify, point out, or define items on the schematic.

argument A value or an expression used with an operator or passed to a subprogram (subroutine, procedure, or function).

attributes Attributes are special characteristics (a name and an associated value) contained in a part instance or definition. For example, a MOSFET may contain specific length and width parameters which are represented as attributes on the symbol or part. Attributes may be changed through the Schematic Editor and/or the Symbol Editor.

block A block is a user defined rectangle placed on a schematic. It is used to represent or hold the place for a collection of circuitry. The block is treated as a "black box" by Schematics. Schematics is aware of the connections going into and out of the block, but ignores the contents of the block until netlisting.

bus A bus is a collection of homogeneously named signals.

call To transfer a program execution to some section of code (usually a subroutine of some sort), while saving the necessary information to allow execution to resume at the calling point when the call section has completed execution.

circuit A circuit is a configuration of electrically connected components or devices.

comment A statement written into a program for documentation purposes only and not for any functionality purposes.

compiler Translates between high-level computer language understood by humans and machine language that is understood by computers.

component A device or part employed in a circuit to obtain some desired action. See *package*.

connector A connector is a physical device that is used for external connections to a circuit board.

construct A computer program statement that produces a predetermined effect.

current source A current source can be an ideal current source (no limit on the supply voltage) or a voltage source with a series resistor.

defined function A computer instruction specifying the operation to be done with predetermined limits.

declarative statement A computer source program instruction specifying the size, format, and kind of data elements and variables in a program for a complier.

device A simple or complex discrete electronic component. Sometimes, a subsystem employed as a unit and, therefore, thought of as a single component. See *package*.

DIBL drain-induced barrier lowering (MOSFET device)

"dot" command A type of formatting command typed into a document that is preceded by a period (dot) to distinguish from other syntax text.

doping tail A changing amount of impurity in a semiconductor device. It is observed as a change in the bulk resistance of the semiconductor material.

ELSE An operation used in BASIC computer programing. It specifies the operation to be performed if the conditions given in the same program line didn't occur.

flicker noise	A repeating low-frequency noise.
Fourier analysis	A mathematical method of transforming a function in such a way that the data of the function is retained but the representation of that data is changed. It is used to simplify the reduction of the data.
FSTIM	digital file stimulus device
gate	A gate is a subset of a package, and corresponds to a part instance. An electronic switch that follows a rule of Boolean logic.
glitch	An unwanted transient that recurs irregularly in the system.
global temperature	Universally applied temperature (to all elements of a circuit)
global parameter	Universally applied parameter (to all elements of a circuit)
icon	A small graphics image displayed on the screen to represent an object that can be manipulated by the user.
IF	An operation used in BASIC computer programing. It specifies an IF-THEN operation to be performed when a condition has changed from what was expected in a program line.
ISAS	independent current source and stimulus
included file	A smaller file that is read into a larger source-code file at a specific spot and becomes part of a statement within the larger source-code file.
instance name	A name of an object in an object oriented programing. It is a unique name for a part instance.
instantiate	To create an instance of a class in object oriented programing.
invocation	To start a software program by invoking an initial power from a higher power
invoke	To call or activate; used in reference to commands and subroutines.
ionization knee	A bend in the response curve where ionization starts.
IS temperature	The temperature of the JFET and other transistor types junction saturation current or the input leakage current
iteration	A repeating series of arithmetic operations to arrive at a solution.
Jiles-Atherton model	A state equation model rather than an explicit function for an inductor
junction	A junction graphically indicates that wires, buses, and/or pins are electrically connected.
keyword	The significant word in a syntax statement that directs the process of the operation.
labels	Is a word or symbol used to identify a file or other element defined in a computer program.

LIBPATH A variable that specifies the directory that the model library is in, and is first set in the msim.ini file.

link A branch instruction, or an address in such an instruction, used to leave a subroutine to return to some point in the main program.

lot tolerance The tolerance of a group of items taken as one unit.

lsb least significant bit

menu A list of options in a window environment that can be selected for performing the listed action. Sometimes clicking the mouse button on one menu item will cascade from one menu to another until the final action is obtained.

metafile A file that contains or defines other files.

mobility movement of electrons in semiconductor devices such as MOSFETs

model library consists of analog models of off-the-shelf parts that can be used directly in circuits that are being developed

mouse A common pointing device used in a windows environment. The physical movement of the mouse will move the pointer (cursor) on the screen.

msb most significant bit

msim.ini The (MicroSim) simulator initiation file that has the default elements that are used to complete a simulation.

nesting The embedding of one construct (such as a table in a database; a data structure, a control structure) inside another-for example a nested procedure is a procedure declared within a procedure.

NETLIST The netlist provides the circuit definition and connectivity information in simulation netlist format.

NODESET A nodeset symbol contains one or two pins, permitting you to initialize a node voltage for simulation.

NOREUSE flag A piece of information that tells the simulator that the automatic saving and restoring of bias point information between different temperatures, Monte Carlo runs, worst-case runs, or parametric analyses is suppressed. It is one of the options in the .OPTIONS command.

NOSUBCKT A variable that tells the simulator not to save the node voltages and inductor currents for subcircuits.

NUMDGT An option that tells the simulator the number of digits that will be printed for the analog values. It is one of the options in the .OPTIONS command.

object A variable comprising both routines and data that is treated as a discrete entity, in object-oriented programing.

operator A symbol (mathematical, as an example) or other character indicating an operation that acts on one or more elements.

OUTPUT ALL An option that ask for an output from the sensitivity runs, after the nominal (first) run. The output from any run is governed by the .PRINT, .PLOT, and .PROBE command in the file. If OUTPUT ALL is omitted, then only the nominal and worst-case runs produce output. OUTPUT ALL ensures that all sensitivity information is saved for Probe.

package A package is an enclosure for an electronic device or subsystem. A physical device consisting of one or more gates.

page A page may contain both parts (represented by symbols), port instances, connectors, and annotation symbols. A page may or may not have a title. Each schematic page represents a single page of a circuit design.

parameter A value that is given to a variable for programing.

part A part is an electrical component which is represented by a schematic symbol. By “part” it is referred to being the logical rather than the physical component.

part definition (See “symbol definition.”)

part instance A part instance refers to an occurrence of a symbol in a schematic.

pin Pins are contained in parts, ports, and offpage connectors. Parts can contain multiple pins. Each part contains specific pin names associated with the part. Pins may connect to a wire, a bus, or another pin.

pin current The current that flows into or out-of a defined pin.

POLY specifies the number of dimensions of the polynomial

port A port provides connectivity across schematic pages. A port provides the anchor for a single pin. Ports are chosen from library files, placed, moved, and deleted in the same way as are parts. Ports may have multiple connections. Ports consist of three types: global, interface, and offpage (defined in this Appendix).

run The execution of a computer routine or operation.

SCBE substrate current induced body effect (MOSFET device)

schematic A schematic consists of the following components: one or more pages, a set of symbols representing local part definitions or parts in a library file, and/or text.

setpoint A setpoint provides a graphical way of introducing .IC or .NODESET commands for each instance of a symbol. These commands set one or more node voltages for the bias point calculation.

SIMLIBPATH A variable that defines the environment that the simulator is working in (path to the directory that the library is in).

simulation The use of a mathematical model to represent a physical device or process.

skipbp (skip bias point)

statement The smallest executable entity within a programming language. In general, each line of a program is an individual statement and is considered an individual instruction. (Examples: command statements, option statements, control statements, assignment statements, comment statements.)

Statz model GaAsFET model

subcircuit A small collection of components working together to perform a task.

symbol A symbol consists of the graphical representation of a logical or physical electronic part on the schematic page, and its definition. Symbols can be created either for a specific schematic or extracted from a library file, and may contain schematic pages nested within them.

syntax The grammar of a particular computer language, with rules that govern the structure and content of the statement.

TEXTINT A function which returns a text string which is the integer value closest to the value of the <value or expression>; (<value or expression> is a floating-point value)

tick number The number generated from a regular recurring signal emitted by a clocking circuit, or from the interrupt generated by this signal.

TOM model GaAsFET device

VARY BOTH The default option is VARY BOTH. When VARY BOTH is used, sensitivity to parameters using both DEV and LOT specifications is checked only with respect to LOT variations. The parameter is then maximized or minimized using both DEV and LOT tolerances for the worst-case. All devices referencing the model have the same parameter values for the worst-case simulation.

VARY DEV See VARY BOTH

VARY LOT See VARY BOTH

VTO temperature The temperature of the JFET or MOSFET device when there is zero-bias threshold (“pinchoff”) voltage.

window A graphical computer interface. An area on the screen that contains instructional documentation or a message.

Windows An operational system introduced by Microsoft Corporation. It is a graphical user interface environment that runs on MS-DOS-based computers.

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z