

OrCAD PSpice® A/D

User's Guide

Copyright © 1998 OrCAD, Inc. All rights reserved.

Trademarks

OrCAD, OrCAD Layout, OrCAD Express, OrCAD Capture, OrCAD PSpice, and OrCAD PSpice A/D are registered trademarks of OrCAD, Inc. OrCAD Capture CIS, and OrCAD Express CIS are trademarks of OrCAD, Inc.

Microsoft, Visual Basic, Windows, Windows NT, and other names of Microsoft products referenced herein are trademarks or registered trademarks of Microsoft Corporation.

All other brand and product names mentioned herein are used for identification purposes only, and are trademarks or registered trademarks of their respective holders.

Part Number 60-30-632

First edition 30 November 1998

Technical Support	(503) 671-9400
Corporate offices	(503) 671-9500
OrCAD Japan K.K.	81-45-621-1911
OrCAD UK Ltd.	44-1256-381-400
Fax	(503) 671-9501
General email	info@orcad.com
Technical Support email	techsupport@orcad.com
World Wide Web	http://www.orcad.com
OrCAD Design Network (ODN)	http://www.orcad.com/odn



9300 SW Nimbus Ave.
Beaverton, OR 97008 USA

Contents

Before you begin xxvii

Welcome to OrCAD	xxvii
OrCAD PSpice A / D overview	xxviii
How to use this guide	xxix
Typographical conventions	xxix
Related documentation	xxx
Online Help	xxx
If you don't have the standard PSpice A / D package	xxxii
If you have PSpice A / D Basics	xxxii
If you have the demo CD-ROM	xxxv
OrCAD demo CD-ROM	xxxv
What's New	xxxvi

Part one Simulation primer

Chapter 1 Things you need to know 41

Chapter overview	41
What is PSpice A / D?	42
Analyses you can run with PSpice A / D	43
Basic analyses	43
DC sweep & other DC calculations	43
AC sweep and noise	44
Transient and Fourier	45
Advanced multi-run analyses	46
Parametric and temperature	46
Monte Carlo and sensitivity / worst-case	47
Analyzing waveforms with PSpice A / D	48
What is waveform analysis?	48
Using PSpice A / D with other OrCAD programs	49
Using Capture to prepare for simulation	49

What is the Stimulus Editor?	49
What is the Model Editor?	50
Files needed for simulation	50
Files that Capture generates	50
Netlist file	51
Circuit file	51
Other files that you can configure for simulation	51
Model library	52
Stimulus file	53
Include file	53
Configuring model library, stimulus, and include files	53
Files that PSpice A/D generates	54
Waveform data file	54
PSpice output file	54

Chapter 2 Simulation examples 55

Chapter overview	55
Example circuit creation	56
Finding out more about setting up your design	61
Running PSpice A/D	62
Performing a bias point analysis	62
Using the simulation output file	64
Finding out more about bias point calculations	65
DC sweep analysis	66
Setting up and running a DC sweep analysis	66
Displaying DC analysis results	68
Finding out more about DC sweep analysis	71
Transient analysis	72
Finding out more about transient analysis	76
AC sweep analysis	77
Setting up and running an AC sweep analysis	77
AC sweep analysis results	79
Finding out more about AC sweep and noise analysis	81
Parametric analysis	82
Setting up and running the parametric analysis	83
Analyzing waveform families	85
Finding out more about parametric analysis	88
Performance analysis	89
Finding out more about performance analysis	91

Part two Design entry

Chapter 3 Preparing a design for simulation 95

Chapter overview	95
Checklist for simulation setup	96
Typical simulation setup steps	96
Advanced design entry and simulation setup steps	97
When netlisting fails or the simulation	
does not start	98
Things to check in your design	98
Things to check in your system configuration	99
Using parts that you can simulate	100
Vendor-supplied parts	101
Part naming conventions	101
Finding the part that you want	102
Passive parts	104
Breakout parts	105
Behavioral parts	106
Using global parameters and expressions for values	107
Global parameters	107
Declaring and using a global parameter	107
Expressions	109
Specifying expressions	109
Defining power supplies	114
For the analog portion of your circuit	114
For A/D interfaces in mixed-signal circuits	114
Default digital power supplies	114
Custom digital power supplies	114
Defining stimuli	116
Analog stimuli	116
Using VSTIM and ISTIM	117
If you want to specify multiple stimulus types	118
Digital stimuli	119
Things to watch for	120
Unmodeled parts	120
Do this if the part in question is from the OrCAD libraries	120
Check for this if the part in question is custom-built	122
Unconfigured model, stimulus, or include files	122
Check for this	123
Unmodeled pins	123
Check for this	124

Missing ground	124
Check for this	124
Missing DC path to ground	125
Check for this	125

Chapter 4 Creating and editing models 127

Chapter overview	127
What are models?	129
Models defined as model parameter sets	129
Models defined as subcircuit netlists	129
How are models organized?	130
Model libraries	130
Model library configuration	131
Global vs. design models and libraries	131
Nested model libraries	132
OrCAD-provided models	132
Tools to create and edit models	133
Ways to create and edit models	134
Using the Model Editor to	
edit models	135
Ways to use the Model Editor	136
Model Editor-supported device types	137
Ways To Characterize Models	138
Creating models from data sheet information	138
Analyzing the effect of model parameters	
on device characteristics	139
How to fit models	139
Running the Model Editor alone	141
Starting the Model Editor	141
Enabling and disabling automatic part creation	142
Saving global models (and parts)	142
Running the Model Editor from the schematic page editor	143
What is an instance model?	143
Starting the Model Editor	144
Saving design models	144
What happens if you don't save the instance model	145
The Model Editor tutorial	146
Creating the half-wave rectifier design	146
Using the Model Editor to edit the D1 diode model	147
Entering data sheet information	147
Extracting model parameters	149

Adding curves for more than one temperature	150
Completing the model definition	151
Editing model text	152
Editing .MODEL definitions	152
Editing .SUBCKT definitions	153
Changing the model name	153
Starting the Model Editor	
from the schematic page editor in Capture	153
What is an instance model?	154
Starting the Model Editor	154
Saving design models	155
Example: editing a Q2N2222 instance model	156
Starting the Model Editor	156
Editing the Q2N2222-X model instance	156
Saving the edits and updating the schematic	157
Using the Create Subcircuit command	157
Changing the model reference to an existing model definition	159
Reusing instance models	160
Reusing instance models in the same schematic	160
Making instance models available to all designs	161
Configuring model libraries	162
The Libraries and Include Files tabs	162
How PSpice A / D uses model libraries	163
Search order	163
Handling duplicate model names	164
Adding model libraries to the configuration	164
Changing design and global scope	165
Changing model library search order	166
Changing the library search path	167

Chapter 5 Creating parts for models 169

Chapter overview	169
What's different about parts used for simulation?	171
Ways to create parts	
for models	171
Preparing your models for part creation	172
Using the Model Editor to create parts	173
Starting the Model Editor	173
Setting up automatic part creation	174
Basing new parts on a custom set of parts	175
Editing part graphics	177

How Capture places parts	177
Defining grid spacing	178
Grid spacing for graphics	178
Grid spacing for pins	178
Attaching models to parts	180
MODEL	180
Defining part properties needed for simulation	181
PSPICETEMPLATE	182
PSPICETEMPLATE syntax	182
PSPICETEMPLATE examples	185
IO_LEVEL	189
MNTYMXDLY	190
PSPICEDEFAULTNET	191

Chapter 6 Analog behavioral modeling 193

Chapter overview	193
Overview of analog behavioral modeling	194
The ABM.OLB part library file	195
Placing and specifying ABM parts	196
Net names and device names in ABM expressions	196
Forcing the use of a global definition	197
ABM part templates	198
Control system parts	199
Basic components	201
Limiters	202
Chebyshev filters	203
Integrator and differentiator	206
Table look-up parts	206
Laplace transform part	210
Math functions	213
ABM expression parts	214
An instantaneous device example: modeling a triode	217
PSpice A/D-equivalent parts	220
Implementation of PSpice A/D-equivalent parts	221
Modeling mathematical or instantaneous relationships	222
EVALUE and GVALUE parts	222
EMULT, GMULT, ESUM, and GSUM	224
Lookup tables (ETABLE and GTABLE)	225
Frequency-domain device models	227
Laplace transforms (LAPLACE)	227
Frequency response tables (EFREQ and GFREQ)	229

Cautions and recommendations for simulation and analysis	232
Instantaneous device modeling	232
Frequency-domain parts	233
Laplace transforms	233
Non-causality and Laplace transforms	235
Chebyshev filters	237
Frequency tables	237
Trading off computer resources for accuracy	238
Basic controlled sources	239
Creating custom ABM parts	239

Chapter 7 **Digital device modeling 241**

Chapter overview	241
Introduction	242
Functional behavior	243
Digital primitive syntax	246
Timing characteristics	251
Timing model	251
Treatment of unspecified propagation delays	252
Treatment of unspecified timing constraints	253
Propagation delay calculation	254
Inertial and transport delay	255
Inertial delay	255
Transport delay	256
Input/Output characteristics	257
Input/Output model	257
Defining Output Strengths	262
Configuring the strength scale	263
Determining the strength of a device output	263
Controlling overdrive	264
Charge storage nets	264
Creating your own interface subcircuits for additional technologies	266
Creating a digital model using the PINDLY and LOGICEXP primitives	271
Digital primitives	272
Logic expression (LOGICEXP primitive)	273
Pin-to-pin delay (PINDLY primitive)	275
BOOLEAN	276
PINDLY	277
Constraint checker (CONSTRAINT primitive)	278

Setup_Hold	279
Width	280
Freq	280
74160 example	280

Part three **Setting Up and Running Analyses**

Chapter 8 Setting up analyses and starting simulation 287

Chapter overview	287
Analysis types	288
Setting up analyses	289
Execution order for standard analyses	290
Output variables	292
Modifiers	293
Starting a simulation	299
Starting a simulation from Capture	299
Starting a simulation outside of Capture	300
Setting up batch simulations	300
Multiple simulation setups within one circuit file	300
Running simulations with multiple circuit files	301
The PSpice A/D simulation window	301

Chapter 9 DC analyses 305

Chapter overview	305
DC Sweep	306
Minimum requirements to run a DC sweep analysis	306
Overview of DC sweep	308
Setting up a DC stimulus	310
Nested DC sweeps	311
Curve families for DC sweeps	313
Bias point	315
Minimum requirements to run a bias point analysis	315
Overview of bias point	315
Small-signal DC transfer	317
Minimum requirements to run a small-signal DC transfer analysis	317
Overview of small-signal DC transfer	318
DC sensitivity	320
Minimum requirements to run a DC sensitivity analysis	320
Overview of DC sensitivity	321

Chapter 10 AC analyses 323

Chapter overview	323
AC sweep analysis	324
Setting up and running an AC sweep	324
What is AC sweep?	324
Setting up an AC stimulus	325
Setting up an AC analysis	327
AC sweep setup in example.opj	329
How PSpice A/D treats nonlinear devices	331
What's required to transform a device into a linear circuit	331
What PSpice A/D does	331
Example: nonlinear behavioral modeling block	331
Noise analysis	333
Setting up and running a noise analysis	333
What is noise analysis?	334
How PSpice A/D calculates total output and input noise	334
Setting up a noise analysis	335
Analyzing Noise in the Probe window	337
About noise units	338
Example	338

Chapter 11 Transient analysis 341

Chapter overview	341
Overview of transient analysis	342
Minimum requirements to run a transient analysis	342
Minimum circuit design requirements	342
Minimum program setup requirements	342
Defining a time-based stimulus	344
Overview of stimulus generation	344
The Stimulus Editor utility	346
Stimulus files	346
Configuring stimulus files	347
Starting the Stimulus Editor	347
Defining stimuli	349
Example: piecewise linear stimulus	349
Example: sine wave sweep	350
Creating new stimulus symbols	352
Editing a stimulus	353
To edit an existing stimulus	353
To edit a PWL stimulus	353

To select a time and value scale factor for PWL stimuli	353
Deleting and removing traces	354
Manual stimulus configuration	354
To manually configure a stimulus	354
Transient (time) response	356
Internal time steps in transient analyses	358
Switching circuits in transient analyses	359
Plotting hysteresis curves	359
Fourier components	361

Chapter 12 Parametric and temperature analysis 363

Chapter overview	363
Parametric analysis	364
Minimum requirements to run a parametric analysis	364
Overview of parametric analysis	365
RLC filter example	366
Entering the design	366
Running the simulation	367
Using performance analysis to plot overshoot and rise time	367
Example: frequency response vs. arbitrary parameter	370
Setting up the circuit	370
Temperature analysis	373
Minimum requirements to run a temperature analysis	373
Overview of temperature analysis	374

Chapter 13 Monte Carlo and sensitivity / worst-case analyses 375

Chapter overview	375
Statistical analyses	376
Overview of statistical analyses	376
Output control for statistical analyses	377
Model parameter values reports	377
Waveform reports	378
Collating functions	379
Temperature considerations in statistical analyses	380
Monte Carlo analysis	381
Reading the summary report	383
Example: Monte Carlo analysis of a pressure sensor	385
Drawing the schematic	385
Defining part values	386
Setting up the parameters	387
Using resistors with models	388

Saving the design	389
Defining tolerances for the resistor models	389
Setting up the analyses	391
Running the analysis and viewing the results	392
Monte Carlo Histograms	393
Chebyshev filter example	393
Creating models for Monte Carlo analysis	394
Setting up the analysis	394
Creating histograms	395
Worst-case analysis	398
Overview of worst-case analysis	398
Inputs	399
Procedure	399
Outputs	400
Caution: An important condition for correct worst-case analysis	400
Worst-case analysis example	401
Tips and other useful information	405
VARY BOTH, VARY DEV, and VARY LOT	405
Gaussian distributions	406
YMAX collating function	406
RELTOL	406
Sensitivity analysis	406
Manual optimization	406
Monte Carlo analysis	407

Chapter 14 Digital simulation 409

Chapter overview	409
What is digital simulation?	410
Steps for simulating digital circuits	410
Concepts you need to understand	411
States	411
Strengths	412
Defining a digital stimulus	413
Using the DIGSTIMn part	414
Defining input signals using the Stimulus Editor	414
Defining clock transitions	414
Defining signal transitions	415
Defining bus transitions	417
Adding loops	420
Using the DIGCLOCK part	422

Using STIM1, STIM4, STIM8 and STIM16 parts	422
Using the FILESTIMn parts	424
Defining simulation time	426
Adjusting simulation parameters	427
Selecting propagation delays	428
Circuit-wide propagation delays	428
Part instance propagation delays	428
Initializing flip-flops	429
Starting the simulation	429
Analyzing results	430
Adding digital signals to a plot	431
Adding buses to a waveform plot	433
Tracking timing violations and hazards	435
Persistent hazards	435
Simulation condition messages	437
Output control options	440
Severity levels	440

Chapter 15 Mixed analog / digital simulation 443

Chapter overview	443
Interconnecting analog and digital parts	444
Interface subcircuit selection by PSpice A / D	445
Level 1 interface	446
Level 2 interface	447
Setting the default A / D interface	448
Specifying digital power supplies	449
Default power supply selection by PSpice A / D	449
Creating custom digital power supplies	450
Overriding CD4000 power supply voltage throughout a design	452
Creating a secondary CD4000, TTL, or ECL power supply	453
Interface generation and node names	454

Chapter 16 Digital worst-case timing analysis 457

Chapter overview	457
Digital worst-case timing	458
Starting worst-case timing analysis	459
Simulator representation of timing ambiguity	459
Propagation of timing ambiguity	461
Identification of timing hazards	462
Convergence hazard	462
Critical hazard	463

Cumulative ambiguity hazard	464
Reconvergence hazard	466
Glitch suppression due to inertial delay	468
Methodology	469

Part four Viewing results

Chapter 17 Analyzing waveforms 475

Chapter overview	475
Overview of waveform analysis	476
Elements of a plot	477
Elements of a Probe window	478
Managing multiple Probe windows	479
Printing multiple windows	479
Setting up waveform analysis	480
Setting up colors	480
Editing display and print colors in the PSPICE.INI file	480
Configuring trace color schemes	482
Viewing waveforms	483
Setting up waveform display from Capture	483
Viewing waveforms while simulating	484
Configuring update intervals	485
Interacting with waveform analysis during simulation	485
Pausing a simulation and viewing waveforms	486
Using schematic page markers to add traces	487
Limiting waveform data file size	490
Limiting file size using markers	490
Limiting file size by excluding internal subcircuit data	492
Limiting file size by suppressing the first part of simulation output	492
Using simulation data from multiple files	493
Appending waveform data files	493
Adding traces from specific loaded waveform data files	494
Saving simulation results in ASCII format	495
Analog example	497
Running the simulation	497
Displaying voltages on nets	499
Mixed analog / digital tutorial	500
About digital states	500
About the oscillator circuit	501
Setting up the design	501

Running the simulation	502
Analyzing simulation results	502
User interface features for waveform analysis	505
Zoom regions	505
Scrolling traces	507
Sizing digital plots	508
Modifying trace expressions and labels	509
Moving and copying trace names and expressions	510
Copying and moving labels	511
Tabulating trace data values	512
Using cursors	513
Displaying cursors	513
Moving cursors	514
Example: using cursors	515
Tracking digital simulation messages	517
Message tracking from the message summary	517
The Simulation Message Summary dialog box	517
Persistent hazards	518
Message tracking from the waveform	519
Trace expressions	519
Basic output variable form	520
Output variable form for device terminals	521
Analog trace expressions	527
Trace expression aliases	527
Arithmetic functions	527
Rules for numeric values suffixes	529
Digital trace expressions	530

Chapter 18 Other output options 533

Chapter overview	533
Viewing analog results in the PSpice window	534
Writing additional results to the PSpice output file	535
Generating plots of voltage and current values	535
Generating tables of voltage and current values	536
Generating tables of digital state changes	537
Creating test vector files	538

Appendix A Setting initial state 541

Appendix overview	541
Save and load bias point	542
Save bias point	542

Load bias point	543
Setpoints	544
Setting initial conditions	546

Appendix B Convergence and “time step too small errors” 547

Appendix overview	547
Introduction	548
Newton-Raphson requirements	548
Is there a solution?	549
Are the Equations Continuous?	550
Are the derivatives correct?	550
Is the initial approximation close enough?	551
Bias point and DC sweep	553
Semiconductors	553
Switches	554
Behavioral modeling expressions	555
Transient analysis	556
Skipping the bias point	557
The dynamic range of TIME	557
Failure at the first time step	558
Parasitic capacitances	559
Inductors and transformers	559
Bipolar transistors substrate junction	560
Diagnostics	561

Index 563

Figures

Figure 1	User-configurable data files that PSpice A / D reads	51
Figure 2	Diode clipper circuit.	56
Figure 3	Connection points.	59
Figure 4	PSpice A / D simulation output window.	62
Figure 5	Simulation output file.	64
Figure 6	DC sweep analysis settings.	67
Figure 7	Probe window.	68
Figure 8	Clipper circuit with voltage marker on net Out.	69
Figure 9	Voltage at In, Mid, and Out.	69
Figure 11	Trace legend with cursors activated.	70
Figure 12	Trace legend with V(Mid) symbol outlined.	70
Figure 13	Voltage difference at V(In) = 4 volts.	71
Figure 14	Diode clipper circuit with a voltage stimulus.	72
Figure 15	Stimulus Editor window.	74
Figure 16	Transient analysis simulation settings.	74
Figure 17	Sinusoidal input and clipped output waveforms.	75
Figure 18	Clipper circuit with AC stimulus.	77
Figure 19	AC sweep and noise analysis simulation settings.	78
Figure 20	dB magnitude curves for “gain” at Mid and Out.	80
Figure 21	Bode plot of clipper’s frequency response.	81
Figure 22	Clipper circuit with global parameter Rval.	82
Figure 23	Parametric simulation settings.	84
Figure 24	Small signal response as R1 is varied from 100 Ω to 10 k Ω	85
Figure 25	Small signal frequency response at 100 and 10 k Ω input resistance.	87
Figure 26	Performance analysis plots of bandwidth and gain vs. Rval.	90
Figure 27	Relationship of the Model Editor to Capture and PSpice A / D.	135
Figure 28	Process and data flow for the Model Editor.	138
Figure 29	Model Editor workspace with data for a bipolar transistor.	139
Figure 30	Design for a half-wave rectifier.	146
Figure 31	Model characteristics and parameter values for DbreakX.	147
Figure 32	Assorted device characteristic curves for a diode.	150
Figure 33	Forward Current device curve at two temperatures.	151

Figure 34	Rules for pin callout in subcircuit templates.	188
Figure 35	LOPASS filter example.	203
Figure 36	HIPASS filter part example.	204
Figure 37	BANDPASS filter part example.	205
Figure 38	BANDREJ filter part example.	205
Figure 39	FTABLE part example.	208
Figure 40	LAPLACE part example one.	211
Figure 41	Viewing gain and phase characteristics of a lossy integrator.	211
Figure 42	LAPLACE part example two.	211
Figure 43	ABM expression part example one.	215
Figure 44	ABM expression part example two.	215
Figure 45	ABM expression part example three.	216
Figure 46	ABM expression part example four.	216
Figure 47	Triode circuit.	217
Figure 48	Triode subcircuit producing a family of I-V curves.	219
Figure 49	EVALUE part example.	223
Figure 50	GVALUE part example.	223
Figure 51	EMULT part example.	224
Figure 52	GMULT part example.	225
Figure 53	EFREQ part example.	231
Figure 54	Voltage multiplier circuit (mixer).	232
Figure 55	Elements of a digital device definition	247
Figure 56	Level 1 and 0 strength determination.	263
Figure 57	PSpice A / D simulation window	303
Figure 58	Example schematic EXAMPLE.OPJ.	309
Figure 59	Curve family example schematic.	313
Figure 60	Device curve family.	314
Figure 61	Operating point determination for each member of the curve family.	314
Figure 62	Circuit diagram for EXAMPLE.OPJ.	329
Figure 63	AC analysis setup for EXAMPLE.OPJ.	330
Figure 64	Device and total noise traces for EXAMPLE.DSN.	339
Figure 65	Transient analysis setup for EXAMPLE.OPJ.	356
Figure 66	Example schematic EXAMPLE.OPJ.	357
Figure 67	ECL-compatible Schmitt trigger.	359
Figure 68	Netlist for Schmitt trigger circuit.	360
Figure 69	Hysteresis curve example: Schmitt trigger.	361
Figure 70	Passive filter schematic.	366
Figure 71	Current of L1 when R1 is 1.5 ohms.	368
Figure 72	Rise time and overshoot vs. damping resistance.	369
Figure 73	RLC filter example circuit.	370
Figure 74	Plot of capacitance versus bias voltage.	372
Figure 75	Example schematic EXAMPLE.OPJ.	374

Figure 76	Example schematic EXAMPLE.DSN.	380
Figure 77	Monte Carlo analysis setup for EXAMPLE.DSN.	382
Figure 78	Summary of Monte Carlo runs for EXAMPLE.OPJ.	383
Figure 79	Parameter values for Monte Carlo pass three.	384
Figure 80	Pressure sensor circuit.	385
Figure 81	Model definition for RMonte1.	390
Figure 82	Pressure sensor circuit with RMonte1 and RTherm model definitions.	391
Figure 83	Chebyshev filter.	394
Figure 84	1 dB bandwidth histogram.	397
Figure 85	Center frequency histogram.	398
Figure 86	Simple biased BJT amplifier.	401
Figure 87	Amplifier netlist and circuit file.	402
Figure 88	YatX Goal Function.	403
Figure 89	Correct worst-case results.	404
Figure 90	Incorrect worst-case results.	404
Figure 91	Schematic using VARY BOTH.	405
Figure 92	Circuit file using VARY BOTH.	405
Figure 93	FILESTIM1 used on a schematic page.	425
Figure 94	Circuit with a timing error	436
Figure 95	Circuit with a timing ambiguity hazard	436
Figure 96	Mixed analog / digital circuit before and after interface generation.	455
Figure 97	Simulation output for mixed analog / digital circuit.	456
Figure 98	Timing ambiguity example one.	460
Figure 99	Timing ambiguity example two.	461
Figure 100	Timing ambiguity example three.	461
Figure 101	Timing ambiguity example four	461
Figure 102	Timing hazard example.	462
Figure 103	Convergence hazard example.	463
Figure 104	Critical hazard example.	463
Figure 105	Cumulative ambiguity hazard example one.	464
Figure 106	Cumulative ambiguity hazard example two.	464
Figure 107	Cumulative ambiguity hazard example three.	465
Figure 108	Reconvergence hazard example one.	466
Figure 109	Reconvergence hazard example two.	466
Figure 110	Glitch suppression example one.	468
Figure 111	Glitch suppression example two.	468
Figure 112	Glitch suppression example three.	469
Figure 113	Analog and digital areas of a plot.	477
Figure 114	Two Probe windows.	478
Figure 115	Trace legend symbols.	494
Figure 116	Section information message box.	495
Figure 117	Example schematic EXAMPLE.OPJ.	497

Figure 118	Waveform display for EXAMPLE.DAT.	498
Figure 119	Mixed analog / digital oscillator design	501
Figure 120	Voltage at net 1 with y-axis added.	503
Figure 121	Mixed analog / digital oscillator results,	504
Figure 122	Cursors positioned on a trough and peak of V(1)	515
Figure 123	Waveform display for a persistent hazard.	518
Figure A-1	Setpoints.	544

Tables

Table 1	DC analysis types	43
Table 2	AC analysis types	44
Table 3	Time-based analysis types	45
Table 4	Parametric and temperature analysis types	46
Table 5	Statistical analysis types	47
Table 2-1		65
Table 10	Association of cursors with mouse buttons.	70
Table 2-1		71
Table 2-1		76
Table 2-2		81
Table 2-3		88
Table 2-4		91
Table 5		98
Table 6		99
Table 7	Passive parts	104
Table 8	Breakout parts	105
Table 9	Operators in expressions	110
Table 10	Functions in arithmetic expressions	111
Table 11	System variables	113
Table 12		114
Table 13		115
Table 14		116
Table 15		118
Table 16		119
Table 17		119
Table 18		121
Table 19	Models supported in the Model Editor	137
Table 1	Sample diode data sheet values	148
Table 2	Part names for custom part generation.	175
Table 3		181
Table 4		183
Table 5		184

Table 6		188
Table 7		189
Table 8		190
Table 9	Control system parts	199
Table 1	ABM math function parts	213
Table 2	ABM expression parts	214
Table 1	PSpice A / D-equivalent parts	220
Table 1	Basic controlled sources in ANALOG.OLB	239
Table 2	Digital primitives summary	243
Table 3	Digital I/O model parameters	260
Table 4	Classes of PSpice A / D analyses	288
Table 5	Execution order for standard analyses	291
Table 6	PSpice A / D output variable formats	294
Table 7	Element definitions for 2-terminal devices	295
Table 8	Element definitions for 3- or 4-terminal devices	296
Table 9	Element definitions for transmission line devices	297
Table 10	Element definitions for AC analysis specific elements	298
Table 11	DC sweep circuit design requirements	306
Table 12		310
Table 13		310
Table 14		311
Table 1	Curve family example setup	313
Table 2		325
Table 3		325
Table 4		326
Table 5		326
Table 6		328
Table 7		334
Table 8		336
Table 9		338
Table 10	Stimulus symbols for time-based input signals	344
Table 1	Parametric analysis circuit design requirements	364
Table 1	Collating functions used in statistical analyses	379
Table 1		386
Table 2		387
Table 3		392
Table 1	Digital states	411
Table 2		413
Table 3		415
Table 4		418
Table 5		419
Table 6		422

Table 7	STIMn part properties	423
Table 8	FILESTIMn part properties	424
Table 9		432
Table 10		433
Table 11		433
Table 12	Simulation condition messages—timing violations	438
Table 13	Simulation condition messages—hazards	439
Table 14	Simulation message output control options	440
Table 15	Interface subcircuit models	446
Table 16	Default digital power / ground pin connections	450
Table 17	Digital power supply parts in SPECIAL.OLB	451
Table 18	Digital power supply properties	451
Table 19	Default waveform viewing colors.	481
Table 20		482
Table 21		484
Table 22		486
Table 23		488
Table 24		489
Table 1		507
Table 2	Mouse actions for cursor control	514
Table 3	Key combinations for cursor control	514
Table 4		520
Table 5		521
Table 6	Output variable formats	521
Table 7	Examples of output variable formats	523
Table 8	Output variable AC suffixes	524
Table 9	Device names for two-terminal device types	524
Table 10	Terminal IDs by three & four-terminal device type	525
Table 11	Noise types by device type	526
Table 12	Analog arithmetic functions for trace expressions	528
Table 13	Output units for trace expressions	529
Table 14		531
Table 15	Digital logical and arithmetic operators	531
Table 16	Signal constants for digital trace expressions	532
Table 17		532
Table 18		535
Table 19		536
Table 20		538

Before you begin

Welcome to OrCAD

OrCAD® offers a total solution for your core design tasks: schematic- and VHDL-based design entry; FPGA and CPLD design synthesis; digital, analog, and mixed-signal simulation; and printed circuit board layout. What's more, OrCAD's products are a suite of applications built around an engineer's design flow--not just a collection of independently developed point tools. PSpice A/D is just one element in OrCAD's total solution design flow.

With OrCAD's products, you'll spend less time dealing with the details of tool integration, devising workarounds, and manually entering data to keep files in sync. Our products will help you build better products faster, and at lower cost.

OrCAD PSpice A / D overview

OrCAD PSpice A / D simulates analog-only, mixed analog / digital, and digital-only circuits. PSpice A / D's analog and digital algorithms are built into the same program so that mixed analog / digital circuits can be simulated with tightly-coupled feedback loops between the analog and digital sections without any performance degradation.

After you prepare a design for simulation, OrCAD Capture generates a circuit file set. The circuit file set, containing the circuit netlist and analysis commands, is read by PSpice A / D for simulation. PSpice A / D formulates these into meaningful graphical plots, which you can mark for display directly from your schematic page using markers.

How to use this guide

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

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

Typographical conventions

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

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

Notation	Examples	Description
<code>Ctrl+R</code>	Press <code>Ctrl+R</code>	A specific key or key stroke on the keyboard.
monospace font	Type VAC . . .	Commands / text entered from the keyboard.

Related documentation

Documentation for OrCAD products is available in both printed and online forms. To access an online manual instantly, you can select it from the Help menu in its respective program (for example, access the Capture User’s Guide from the Help menu in Capture).

Note *The documentation you receive depends on the software configuration you have purchased.*

The following table provides a brief description of those manuals available in both printed and online forms.

This manual...	Provides information about how to use...
OrCAD Capture User’s Guide	OrCAD Capture, which is a schematic capture front-end program with a direct interface to other OrCAD programs and options.
OrCAD Layout User’s Guide	OrCAD Layout, which is a PCB layout editor that lets you specify printed circuit board sturcture, as well as the components, metal, and graphics required for fabrication.
OrCAD PSpice A / D & Basics User’s Guide	PSpice A / D with Probe, the Stimulus Editor, and the Model Editor, which are circuit analysis programs that let you create, simulate, and test analog and digital circuit designs. This manual provides examples on how to specify simulation parameters, analyze simulation results, edit input signals, and create models. (PSpice A / D Basics is a limited version that does not include the Stimulus Editor.)
OrCAD PSpice User’s Guide	OrCAD PSpice with Probe is a circuit analysis program that lets you create, simulate, and test analog-only circuit designs.
OrCAD PSpice Optimizer User’s Guide	OrCAD PSpice Optimizer, which is an analog performance optimization program that lets you fine-tune your analog circuit designs.

The following table provides a brief description of those manuals available online *only*.

This online manual...	Provides this...
OrCAD PSpice A/D Online Reference Manual	Reference material for PSpice A/D. Also included: detailed descriptions of the simulation controls and analysis specifications, start-up option definitions, and a list of device types in the analog and digital model libraries. User interface commands are provided to instruct you on each of the screen commands.
OrCAD Application Notes Online Manual	A variety of articles that show you how a particular task can be accomplished using OrCAD's products, and examples that demonstrate a new or different approach to solving an engineering problem.
OrCAD PSpice Library List	A complete list of the analog and digital parts in the model and part libraries.

Online Help

Choosing Search for Help On from the Help menu displays an extensive online help system.

The online help includes:

- step-by-step instructions on how to set up PSpice A/D simulations and analyze simulation results
- reference information about PSpice A/D
- Technical Support information

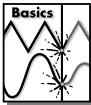
If you are not familiar with Windows (NT or 95) Help system, choose How to Use Help from the Help menu.

If you don't have the standard PSpice A / D package

If you have PSpice A/D Basics

PSpice A / D Basics provides the basic functionality needed for analog and mixed-signal design without the advanced features in the full PSpice A / D package. Because this guide is for both PSpice A / D Basics and PSpice A / D users, there are some features described here that are not available to PSpice A / D Basics users.

Note *Not supported in PSpice A/D Basics.*



The Basics icon (shown in the sidebar) is used throughout this user's guide to mark each section or paragraph which describes a feature *not available* to PSpice A / D Basics users. If an entire section describes a “non-Basics” feature, the icon is placed next to the section title. If an individual paragraph describes a “non-Basics” feature, the icon is placed next to the paragraph.

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

Feature	PSpice A/D (standard)	PSpice A/D Basics
Benefits of integration with OrCAD Capture		
graphical design entry (schematic capture)	yes	yes
simulation setup using dialog boxes	yes	yes
cross-probing	yes	yes
multi-window analysis of PSpice data sets	yes	yes
marching waveforms in PSpice	yes	yes
board layout package interfaces	yes	yes
Notable PSpice analysis and simulation features		
DC sweep, AC sweep, transient analysis	yes	yes
noise, Fourier, temperature analysis	yes	yes
parametric analysis	yes	no

Note *For expert PSpice A/D users, these are the PSpice circuit file commands that are not available in the Basics package:*

- .STIMULUS
- .STIMLIB
- .SAVEBIAS
- .LOADBIAS

Feature	PSpice A/D (standard)	PSpice A/D Basics
Monte Carlo, sensitivity / worst-case analysis	yes	no
analog behavioral modeling (ABM)	yes	yes
propagation delay modeling	yes	no
constraint checking (such as setup and hold timing)	yes	no
digital worst-case timing	yes	no
charge storage on digital nets	yes	no
Stimulus Editor	yes	no
Parts utility	yes	no
performance analysis (goal functions)	yes	no
save/load bias point	yes	no
Notable PSpice devices and library models		
GaAsFETs: Curtice, Statz, TriQuint, Parker-Skellern	all	Statz
MOSFETs: SPICE3 (1-3) with charge conservation, BSIM1, BSIM3.1 (version 3), EKV (version 2.6)	yes	yes
IGBTs	yes	no
JFETs, BJTs	yes	yes
resistor, capacitor, and inductor .MODEL support	yes	yes
ideal, non-ideal lossy transmission lines	all	ideal
coupled inductors	yes	yes
coupled transmission lines	yes	no
nonlinear magnetics	yes	no
voltage- and current-controlled switches	yes	yes
analog model library	10,200+	10,200+ *
Notable PSpice devices and library models, continued		
digital primitives	all	most**
digital model library	1600+	1600+

Feature	PSpice A / D (standard)	PSpice A / D Basics
Purchase options		
OrCAD Layout	yes	yes
OrCAD PSpice Optimizer	yes	no
Device Equations	yes	no
network licensing	yes	no
Miscellaneous specifications		
unlimited circuit size	yes	yes

* PSpice A / D Basics package includes all libraries except IGBTs, SCRs, thyristors, PWMs, magnetic cores, and transmission lines.

** PSpice A / D Basics does not include bidirectional transfer gates.

If you have the demo CD-ROM

OrCAD demo CD-ROM

The OrCAD demo CD-ROM has the following limitations for PSpice A/D:

- circuit simulation limited to circuits with up to 64 nodes, 10 transistors, two operational amplifiers or 65 digital primitive devices, and 10 transmission lines (ideal or non-ideal) with not more than 4 pairwise coupled lines
- device characterization using the Model Editor limited to diodes
- stimulus generation limited to sine waves (analog) and clocks (digital)
- sample library of approximately 39 analog and 134 digital parts
- displays only simulation data created using the demo version of the simulator
- PSpice Optimizer limited to one goal, one parameter and one constraint
- designs created in Capture can be saved if they have no more than 30 part instances

What's New

To find out more, see [Analyzing waveforms on page -475](#).

New PSpice interface with integrated waveform analysis functionality Release 9 of PSpice A/D includes all of Probe's features and adds to them. Included in one screen are tabbed windows for viewing plots, text windows for viewing output files or other text files, and a simulation status and message window. Also included is a new, self-documenting analysis setup dialog for creating simulation profiles (see below). PSpice A/D now provides an editable simulation queue which shows you how many files are currently in line to be simulated. You can edit or re-order the list as needed. And the plotting features have been improved by providing user-controlled grid settings, grid and trace properties (style and color) and metafile format copy and paste functions.

Simulation profiles PSpice A/D Release 9 introduces the concept of simulation profiles. Each simulation profile refers to one schematic in a design and includes one analysis type (AC, DC, or Transient) with any options (sensitivity, temperature, parametric, Monte Carlo, etc.). You can define as many profiles as you need for your design and you can set up multiple analyses of the same type. Simulation profiles help you keep your analysis results separate, so you can delete one without losing the rest.

New OrCAD Capture front-end Release 9 integrates OrCAD Capture as the front-end schematic entry tool for PSpice A/D. Capture provides a professional design entry environment with many advanced capabilities that now work hand-in-hand with PSpice A/D. These include a project manager, a new property editor spreadsheet, right mouse button support, and many other time-saving features.

New Model Editor interface The Model Editor (formerly known as Parts) has been improved and modernized for Release 9. It now provides a unified application for editing models either in text form or by modifying their specifications. The Model Editor now also supports Darlington modeling.

To find out more, see [Creating and editing models on page -127](#).

EKV version 2.6 MOSFET model The EKV model is a scalable and compact model built on fundamental physical properties of the device. Use this model to design low-voltage, low-current analog, and mixed analog-digital circuits that use sub-micron technologies.

To find out more, refer to MOSFET devices in the *Analog Devices* chapter of the online *OrCAD PSpice A/D Reference Manual*.

Version 2.6 models the following:

- geometrical and process related aspects of the device (oxide thickness, junction depth, effective channel length and width, and so on)
- effects of doping profile and substrate effects
- weak, moderate, and strong inversion behavior
- mobility effects due to vertical and lateral fields and carrier velocity saturation
- short-channel effects such as channel-length modulation, source and drain charge sharing, and the reverse short channel effect
- thermal and flicker noise modeling
- short-distance geometry and bias-dependent device matching for Monte Carlo analysis

Enhanced model libraries The model libraries supplied with PSpice A/D Release 9 have been enhanced to include the latest models from various vendors, as well as models for popular optocouplers, Darlingtons, and DAC and ADC devices.

Part one

Simulation primer

Part one provides basic information about circuit simulation including examples of common analyses.

- [Chapter 1, Things you need to know](#), provides an overview of the circuit simulation process including what PSpice A / D does, descriptions of analysis types, and descriptions of important files.
- [Chapter 2, Simulation examples](#), presents examples of common analyses to introduce the methods and tools you'll need to enter, simulate, and analyze your design.

Things you need to know

1

Chapter overview

This chapter introduces the purpose and function of the OrCAD® PSpice A / D circuit simulator.

- [What is PSpice A/D? on page 1-42](#) describes PSpice A / D capabilities.
- [Analyses you can run with PSpice A/D on page 1-43](#) introduces the different kinds of basic and advanced analyses that PSpice A / D supports.
- [Using PSpice A/D with other OrCAD programs on page 1-49](#) presents the high-level simulation design flow.
- [Files needed for simulation on page 1-50](#) describes the files used to pass information between OrCAD programs. This section also introduces the things you can do to customize where and how PSpice A / D finds simulation information.
- [Files that PSpice A/D generates on page 1-54](#) describes the files that contain simulation results.

What is PSpice A/D?

Because the analog and digital simulation algorithms are built into the same program, PSpice A/D simulates mixed-signal circuits with no performance degradation because of tightly coupled feedback loops between the analog and digital sections.

OrCAD PSpice A/D is a simulation program that models the behavior of a circuit containing any mix of analog and digital devices. Used with OrCAD Capture for design entry, you can think of PSpice A/D as a software-based breadboard of your circuit that you can use to test and refine your design before ever touching a piece of hardware.

Run basic and advanced analyses PSpice A/D can perform:

- DC, AC, and transient analyses, so you can test the response of your circuit to different inputs.
- Parametric, Monte Carlo, and sensitivity / worst-case analyses, so you can see how your circuit's behavior varies with changing component values.
- Digital worst-case timing analysis to help you find timing problems that occur with only certain combinations of slow and fast signal transmissions.

The range of models built into PSpice A/D include not only those for resistors, inductors, capacitors, and bipolar transistors, but also these:

- transmission line models, including delay, reflection, loss, dispersion, and crosstalk
- nonlinear magnetic core models, including saturation and hysteresis
- six MOSFET models, including BSIM3 version 3.1 and EKV version 2.6
- five GaAsFET models, including Parker-Skellern and TriQuint's TOM2 model
- IGBTs
- digital components with analog I/O models

Use parts from OrCAD's extensive set of libraries

The model libraries feature over 11,300 analog and 1,600 digital models of devices manufactured in North America, Japan, and Europe.

Vary device characteristics without creating new parts PSpice A/D has numerous built-in models with parameters that you can tweak for a given device. These include independent temperature effects.

Model behavior PSpice A/D supports analog and digital behavioral modeling, so you can describe functional blocks of circuitry using mathematical expressions and functions.

Analyses you can run with PSpice A/D

See [Chapter 2, Simulation examples](#), for introductory examples showing how to run each type of analysis.

See [Part three, Setting Up and Running Analyses](#), for a more detailed discussion of each type of analysis and how to set it up.

Basic analyses

DC sweep & other DC calculations

These DC analyses evaluate circuit performance in response to a direct current source. [Table 1](#) summarizes what PSpice A/D calculates for each DC analysis type.

Table 1 *DC analysis types*

For this DC analysis...	PSpice A/D computes this...
DC sweep	Steady-state voltages, currents, and digital states when sweeping a source, a model parameter, or temperature over a range of values.
Bias point detail	Bias point data in addition to what is automatically computed in any simulation.
DC sensitivity	Sensitivity of a net or part voltage as a function of bias point.
Small-signal DC transfer	Small-signal DC gain, input resistance, and output resistance as a function of bias point.

AC sweep and noise

These AC analyses evaluate circuit performance in response to a small-signal alternating current source. **Table 2** summarizes what PSpice A / D calculates for each AC analysis type.

Table 2 AC analysis types

For this AC analysis...	PSpice A / D computes this...
AC sweep	Small-signal response of the circuit (linearized around the bias point) when sweeping one or more sources over a range of frequencies. Outputs include voltages and currents with magnitude and phase; you can use this information to obtain Bode plots.
Noise	For each frequency specified in the AC analysis: • Propagated noise contributions at an output net from every noise generator in the circuit. • RMS sum of the noise contributions at the output. • Equivalent input noise.

Note To run a noise analysis, you must also run an AC sweep analysis.

Transient and Fourier

These time-based analyses evaluate circuit performance in response to time-varying sources. [Table 3](#) summarizes what PSpice A/D calculates for each time-based analysis type.

Table 3 *Time-based analysis types*

For this time-based analysis...	PSpice A/D computes this...
Transient	Voltages, currents, and digital states tracked over time. For digital devices, you can set the propagation delays to minimum, typical, and maximum. If you have enabled digital worst-case timing analysis, then PSpice A/D considers all possible combinations of propagation delays within the minimum and maximum range.
Fourier	DC and Fourier components of the transient analysis results.

Note *To run a Fourier analysis, you must also run a transient analysis.*

Advanced multi-run analyses

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

Parametric and temperature

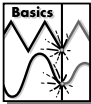
For parametric and temperature analyses, PSpice A / D steps a circuit value in a sequence that you specify and runs a simulation for each value.

Table 4 shows the circuit values that you can step for each kind of analysis.

Table 4 Parametric and temperature analysis types

For this analysis...	You can step one of these...
Parametric	global parameter model parameter component value DC source operational temperature
Temperature	operational temperature

Note Parametric analysis is not supported in PSpice A/D Basics.



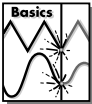
Monte Carlo and sensitivity / worst-case

Monte Carlo and sensitivity / worst-case analyses are statistical. PSpice A / D changes device model parameter values with respect to device and lot tolerances that you specify, and runs a simulation for each value.

Table 5 summarizes how PSpice A / D runs each statistical analysis type.

Table 5 *Statistical analysis types*

For this statistical analysis...	PSpice A / D does this...
Monte Carlo	For each simulation, <i>randomly</i> varies all device model parameters for which you have defined a tolerance.
Sensitivity / worst-case	Computes the probable worst-case response of the circuit in two steps: 1 Computes component sensitivity to changes in the device model parameters. This means PSpice A / D <i>nonrandomly</i> varies device model parameters for which you have defined a tolerance, one at a time for each device and runs a simulation with each change. 2 Sets <i>all</i> model parameters for <i>all</i> devices to their worst-case values (assumed to be at one of the tolerance limits) and runs a final simulation.



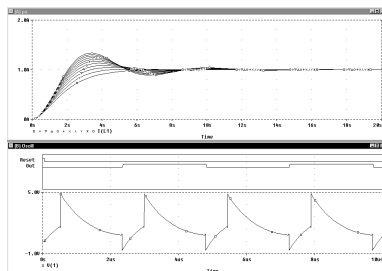
Note Monte Carlo/Worst Case Analysis is not supported in PSpice A/D Basics.

Analyzing waveforms with PSpice A/D

What is waveform analysis?

After completing the simulation, PSpice A/D plots the waveform results so you can visualize the circuit's behavior and determine the validity of your design.

Taken together, simulation and waveform analysis is an iterative process. After analyzing simulation results, you can refine your design and simulation settings and then perform a new simulation and waveform analysis.



Perform post-simulation analysis of the results This means you can plot additional information derived from the waveforms. What you can plot depends on the types of analyses you run. Bode plots, phase margin, derivatives for small-signal characteristics, waveform families, and histograms are only a few of the possibilities. You can also plot other waveform characteristics such as rise time versus temperature, or percent overshoot versus component value.

Pinpoint design errors in digital circuits When PSpice A/D detects setup and hold violations, race conditions, or timing hazards, a detailed message appears along with corresponding waveforms. PSpice A/D also helps you locate the problem in your design.

Using PSpice A/D with other OrCAD programs

Using Capture to prepare for simulation

Capture is a design entry program you need to prepare your circuit for simulation. This means:

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

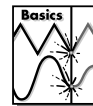
Capture is also the control point for running other programs used in the simulation design flow.

What is the Stimulus Editor?

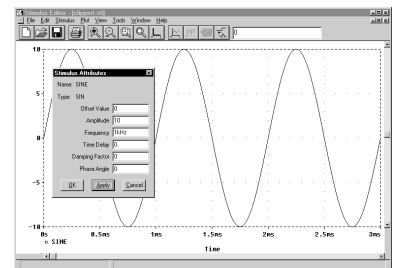
The Stimulus Editor is a graphical input waveform editor that lets you define the shape of time-based signals used to test your circuit's response during simulation. Using the Stimulus Editor, you can define:

- analog stimuli with sine wave, pulse, piecewise linear, exponential pulse, single-frequency FM shapes, and
- digital stimuli that range from simple clocks to complex pulse patterns and bus sequences.

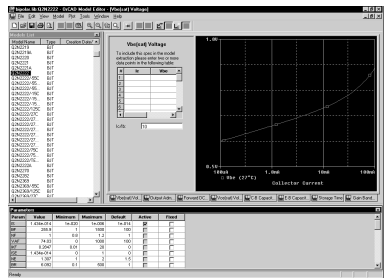
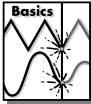
The Stimulus Editor lets you draw analog piecewise linear and all digital stimuli by clicking at the points along the timeline that correspond to the input values that you want at transitions.



Note The Stimulus Editor is not included in PSpice A/D Basics.



Note *The Model Editor is not included in PSpice A/D Basics.*



What is the Model Editor?

The Model Editor is a model extractor that generates model definitions for PSpice A / D to use during simulation. All the Model Editor needs is information about the device found in standard data sheets. As you enter the data sheet information, the Model Editor displays device characteristic curves so you can verify the model-based behavior of the device. When you are finished, the Model Editor automatically creates a part for the model so you can use the modeled part in your design immediately.

Files needed for simulation

To simulate your design, PSpice A / D needs to know about:

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

This information is provided in various data files. Some of these are generated by Capture, others come from libraries (which can also be generated by other programs like the Stimulus Editor and the Model Editor), and still others are user-defined.

Files that Capture generates

When you begin the simulation process, Capture first generates files describing the parts and connections in your circuit. These files are the netlist file and the circuit file that PSpice A / D reads before doing anything else.

Netlist file

The netlist file contains a list of device names, values, and how they are connected with other devices. The name that Capture generates for this file is DESIGN_NAME.NET.

Refer to the online *OrCAD PSpice A/D Reference Manual* for the syntax of the statements in the netlist file and the circuit file.

Circuit file

The circuit file contains commands describing how to run the simulation. This file also refers to other files that contain netlist, model, stimulus, and any other user-defined information that apply to the simulation. The name that Capture generates for this file is DESIGN_NAME.CIR.

Other files that you can configure for simulation

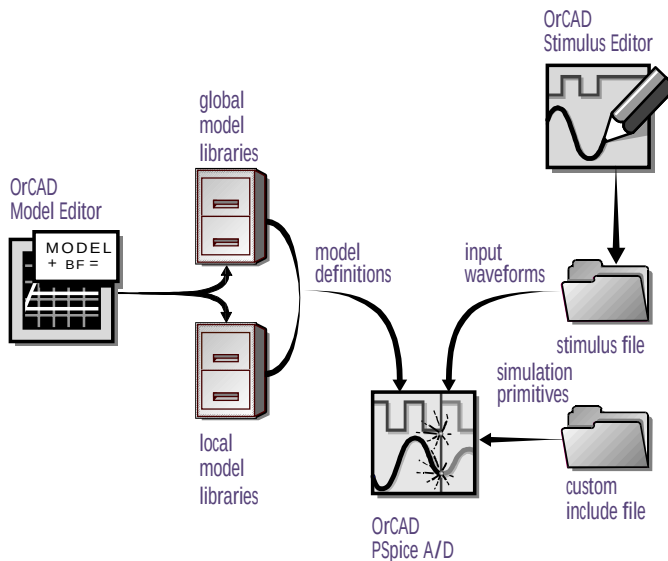


Figure 1 User-configurable data files that PSpice A/D reads

Before starting simulation, PSpice A/D needs to read other files that contain simulation information for your circuit. These are model files, and if required, stimulus files and include files.

The circuit file (.CIR) that Capture generates contains references to the other user-configurable files that PSpice A/D needs to read.

Note *The Stimulus Editor is not included in PSpice A/D Basics.*



You can create these files using OrCAD programs like the Stimulus Editor and the Model Editor. These programs automate file generation and provide graphical ways to verify the data. You can also use the Model Text view in the Model Editor (or another text editor like Notepad) to enter the data manually.

Model library

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

These definitions take the form of either a:

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

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

If needed, however, you can create your own models and libraries, either:

- manually using the Model Text view in the Model Editor (or another text editor like Notepad), or
- automatically using the Model Editor.

See [What is the Model Editor?](#) on page 1-50 for a description.

A subcircuit, sometimes called a *macromodel*, is analogous to a procedure call in a software programming language.

Stimulus file

A stimulus file contains time-based definitions for analog and /or digital input waveforms. You can create a stimulus file either:

- manually using the Model Text View of the Model Editor (or a standard text editor) to create the definition (a typical file extension is .STM), or
- automatically using the Stimulus Editor (which generates a .STL file extension).

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

See [What is the Stimulus Editor? on page 1-49](#) for a description.

Include file

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

- PSpice commands, or
- supplemental text comments that you want to appear in the PSpice output file (see page [1-54](#)).

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

Example: An include file that contains definitions, using the PSpice .FUNC command, for functions that you want to use in numeric expressions elsewhere in your design.

Configuring model library, stimulus, and include files

PSpice A/D searches model libraries, stimulus files, and include files for any information it needs to complete the definition of a part or to run a simulation.

The files that PSpice A/D searches depend on how you configure your model libraries and other files. Much of the configuration is set up for you automatically, however, you can do the following yourself:

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

More on libraries...

Configuration for model libraries is similar to that for other libraries that Capture uses, including part libraries. To find out more, refer to your Capture user's guide.

Files that PSpice A/D generates

After reading the circuit file, netlist file, model libraries, and any other required inputs, PSpice A/D starts the simulation. As simulation progresses, PSpice A/D saves results to two files—the data file and the PSpice output file.

For a description of how to display simulation results, see [Part four, Viewing results](#).

For a description of the waveform analyzer program, see [What is waveform analysis? on page 1-48](#).

There are two ways to add waveforms to the display:

- From within PSpice A/D, by specifying trace expressions.
- From within Capture, by cross-probing.

Example: Each instance of a VPRINT1 symbol placed in your schematic causes PSpice A/D to generate a table of voltage values for the connecting net, and to write the table to the PSpice output file.

Waveform data file

The data file contains simulation results that can be displayed graphically. PSpice A/D reads this file automatically and displays waveforms reflecting circuit response at nets, pins, and parts that you marked in your schematic (cross-probing). You can set up your design so PSpice A/D displays the results as the simulation progresses or after the simulation completes.

After PSpice A/D has read the data file and displays the initial set of results, you can add more waveforms and to perform post-simulation analysis of the data.

PSpice output file

The PSpice output file is an ASCII text file that contains:

- the netlist representation of the circuit,
- the PSpice command syntax for simulation commands and options (like the enabled analyses),
- simulation results, and
- warning and error messages for problems encountered during read-in or simulation.

Its content is determined by:

- the types of analyses you run,
- the options you select for running PSpice A/D, and
- the simulation control symbols (like VPRINT1 and VPLOT1) that you place and connect to nets in your design.

Simulation examples

2

Chapter overview

The examples in this chapter provide an introduction to the methods and tools for creating circuit designs, running simulations, and analyzing simulation results. All analyses are performed on the same example circuit to clearly illustrate analysis setup, simulation, and result-analysis procedures for each analysis type.

This chapter includes the following sections:

- [Example circuit creation on page 2-56](#)
- [Performing a bias point analysis on page 2-62](#)
- [DC sweep analysis on page 2-66](#)
- [Transient analysis on page 2-72](#)
- [AC sweep analysis on page 2-77](#)
- [Parametric analysis on page 2-82](#)
- [Performance analysis on page 2-89](#)

Example circuit creation

This section describes how to use OrCAD Capture to create the simple diode clipper circuit shown in Figure 2.

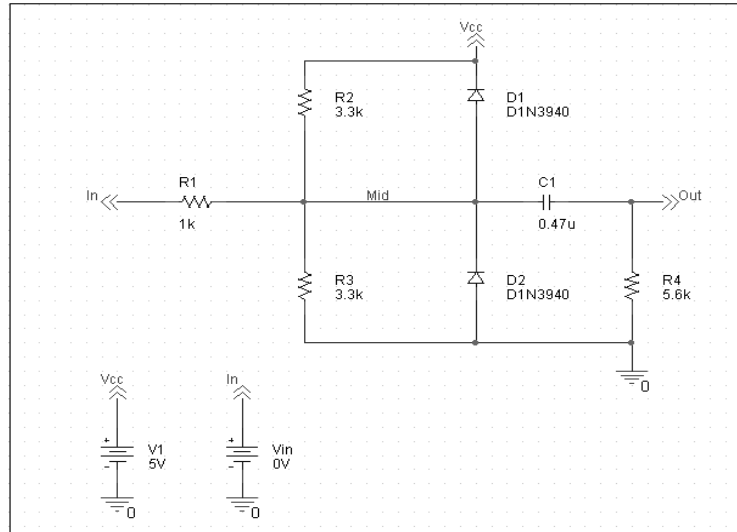


Figure 2 Diode clipper circuit.

To create a new PSpice project

- 1 From the Windows Start menu, choose the OrCAD Release 9 program folder and then the Capture shortcut to start Capture.
- 2 In the Project Manager, from the File menu, point to New and choose Project.
- 3 Select Analog or Mixed-Signal Circuit Wizard.
- 4 In the Name text box, enter the name of the project (CLIPPER).
- 5 Click OK, then click Finish.

No special libraries need to be configured at this time. A new page will be displayed in Capture and the new project will be configured in the Project Manager.

To place the voltage sources

- 1 In Capture, switch to the schematic page editor.

- 2 From the Place menu, choose Part to display the Place Part dialog box.
- 3 Add the library for the parts you need to place:
 - a Click the Add Library button.
 - b Select SOURCE.OLB (from the PSpice library) and click Open.
- 4 In the Part text box, type VDC.
- 5 Click OK.
- 6 Move the pointer to the correct position on the schematic page (see Figure 2) and click to place the first part.
- 7 Move the cursor and click again to place the second part.
- 8 Right-click and choose End Mode to stop placing parts.



Note There are two sets of library files supplied with Capture and PSpice A/D. The standard schematic part libraries are found in the directory `Capture\Library`. The part libraries that are designed for simulation with PSpice A/D are found in the sub-directory `Capture\Library\PSpice`. In order to have access to specific parts, you must first configure the library in Capture using the Add Library function.

To place the diodes

- 1 From the Place menu, choose Part to display the Place Part dialog box.
- 2 Add the library for the parts you need to place:
 - a Click the Add Library button.
 - b Select DIODE.OLB (from the PSpice library) and click Open.
- 3 In the Part text box, type D1N39 to display a list of diodes.
- 4 Select D1N3940 and click OK.
- 5 Press **[R]** to rotate the diode to the correct orientation.
- 6 Click to place the first diode (D1), then click to place the second diode (D2).
- 7 Right-click and choose End Mode to stop placing parts.



When placing parts:

- Leave space to connect the parts with wires.
- You will change part names and values that do not match those shown in Figure 2 later in this section.

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

- 1 Click the text to select it, then drag the text to a new location.

To place the other parts



- 1 From the Place menu, choose Part to display the Place Part dialog box.
- 2 Add the library for the parts you need to place:
 - a Click the Add Library button.
 - b Select ANALOG.OLB (from the PSpice library) and click Open.
- 3 Follow similar steps as described for the diodes to place the parts listed below, according to Figure 2. The part names you need to type in the Part name text box of the Place Part dialog box are shown in parentheses:
 - resistors (R)
 - capacitor (C)



- 4 To place the off-page connector parts (OFFPAGELEFT-R), click the Place Off-Page Connector button on the tool palette.
- 5 Add the library for the parts you need to place:
 - a Click the Add Library button.
 - b Select CAPSYM.OLB (from the Capture library) and click Open.



- 6 Place the off-page connector parts according to Figure 2.
- 7 To place the ground parts (0), click the GND button on the tool palette.
- 8 Add the library for the parts you need to place:
 - a Click the Add Library button.
 - b Select SOURCE.OLB (from the PSpice library) and click Open.
- 9 Place the ground parts according to Figure 2.

To connect the parts

- 1 From the Place menu, choose Wire to begin wiring parts.
The pointer changes to a crosshair.
- 2 Click the connection point (the very end) of the pin on the off-page connector at the input of the circuit.
- 3 Click the nearest connection point of the input resistor R1.
- 4 Connect the other end of R1 to the output capacitor.
- 5 Connect the diodes to each other and to the wire between them:
 - a Click the connection point of the cathode for the lower diode.
 - b Move the cursor straight up and click the wire between the diodes. The wire ends, and the junction of the wire segments becomes visible.
 - c Click again on the junction to continue wiring.
 - d Click the end of the upper diode's anode pin.
- 6 Continue connecting parts until the circuit is wired as shown in Figure 2 on page 2-56.



To stop wiring, right-click and choose End Wire. The pointer changes to the default arrow.

Clicking on any valid connection point ends a wire. A valid connection point is shown as a *box* (see Figure 3).

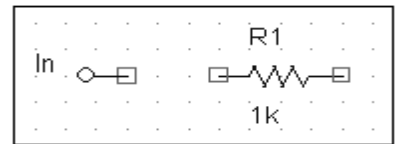


Figure 3 Connection points.

If you make a mistake when placing or connecting components:

- 1 From the Edit menu, choose Undo, or click .

To assign names (labels) to the nets

- 1 From the Place menu, choose Net Alias to display the Place Net Alias dialog box.
- 2 In the Name text box, type Mid.
- 3 Click OK.
- 4 Place the net alias on any segment of the wire that connects R1, R2, R3, the diodes, and the capacitor. The lower left corner of the net alias must touch the wire.
- 5 Right-click and choose End Mode to quit the Net Alias function.

To assign names (labels) to the off-page connectors

Label the off-page connectors as shown in Figure 2 on page 2-56.

- 1 Double-click the name of an off-page connector to display the Display Properties dialog box.
- 2 In the Name text box, type the new name.
- 3 Click OK.
- 4 Select and relocate the new name as desired.

To assign names to the parts

A more efficient way to change the names, values and other properties of several parts in your design is to use the Property Editor, as follows:

- 1 Select all of the parts to be modified by pressing **Ctrl** and clicking each part.
- 2 From the Edit menu, choose Properties.
- 3 Type in the new name Vin.
- 4 Click Apply to update the changes to the part, then close the spreadsheet.
- 5 Continue naming the remaining parts until your schematic looks like Figure 2 on page 2-56.

The Parts Spreadsheet appears.

Change the entries in as many of the cells as needed, and then click Apply to update all of the changes at once.

To change the values of the parts

- 1 Double-click the voltage label (0V) on V1 to display the Display Properties dialog box.
- 2 In the Value text box, type 5V.
- 3 Click OK.
- 4 Continue changing the Part Value properties of the parts until all the parts are defined as in Figure 2 on page 2-56.

Your schematic page should now have the same parts, wiring, labels, and properties as Figure 2 on page 2-56.

To save your design



- 1 From the File menu, choose Save.

Finding out more about setting up your design

About setting up a design for simulation

For a checklist of all of the things you need to do to set up your design for simulation, and how to avoid common problems, see [Chapter 3, Preparing a design for simulation](#).

Running PSpice A/D

When you perform a simulation, PSpice A/D generates an output file (*.OUT).

You can set up a simulation profile to run one analysis at a time. To run multiple analyses (for example, both DC sweep and transient analyses), set up a batch simulation. For more information, see [Chapter 8, Setting up analyses and starting simulation](#).

While PSpice A/D is running, the progress of the simulation appears and is updated in the PSpice A/D simulation output window (see Figure 4).

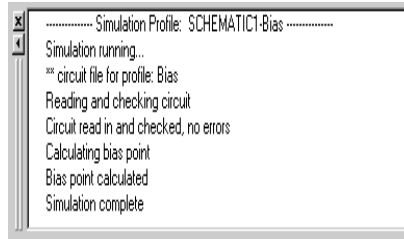


Figure 4 PSpice A/D simulation output window.

Performing a bias point analysis

To set up a bias point analysis in Capture

- 1 In Capture, switch to CLIPPER.OPJ in the schematic page editor.
- 2 From the PSpice menu, choose New Simulation Profile to display the New Simulation dialog box.
- 3 In the Name text box, type Bias.
- 4 From the Inherit From list, select None, then click Create.

The Simulation Settings dialog box appears.

- 5 From the Analysis type list, select Bias Point.
- 6 Click OK to close the Simulation Settings dialog box.

The root schematic listed is the schematic page associated with the simulation profile you are creating.

To simulate the circuit from within Capture

- 1 From the PSpice menu, choose Run.



PSpice A/D simulates the circuit and calculates the bias point information.

Note *Because waveform data is not calculated during a bias point analysis, you will not see any plots displayed in the Probe window for this simulation. To find out how to view the results of this simulation, see Using the simulation output file below.*

Using the simulation output file

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

To view the simulation output file

- 1 From PSpice’s View menu, choose Output File.

Figure 5 shows the results of the bias point calculation as written in the simulation output file.

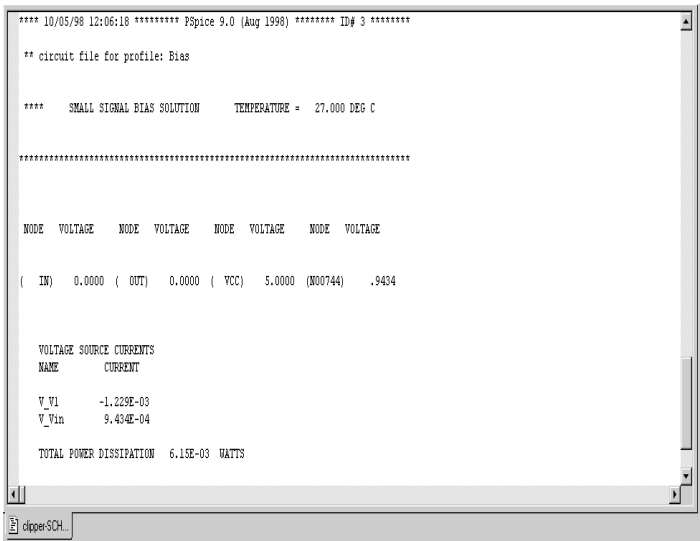


Figure 5 Simulation output file.

- 2 When finished, close the window.

PSpice A/D measures the current through a two terminal device into the first terminal and out of the second terminal. For voltage sources, current is measured from the positive terminal to the negative terminal; this is opposite to the positive current flow convention and results in a negative value in the output file.

Finding out more about bias point calculations

Table 2-1

To find out more about this...	See this...
bias point calculations	Bias point on page 9-315

DC sweep analysis

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

Setting up and running a DC sweep analysis

To set up and run a DC sweep analysis

- 1 In Capture, from the PSpice menu, choose New Simulation Profile.
The New Simulation dialog box appears.
- 2 In the Name text box, type DC Sweep.
- 3 From the Inherit From list, select Schematic1-Bias, then click Create.

The Simulation Settings dialog box appears.

- 4 Click the Analysis tab.
- 5 From the Analysis type list, select DC Sweep and enter the values shown in Figure 6.

Note *The default settings for DC Sweep simulation are Voltage Source as the swept variable type and Linear as the sweep type. To use a different swept variable type or sweep type, choose different options under Sweep variable and Sweep type.*

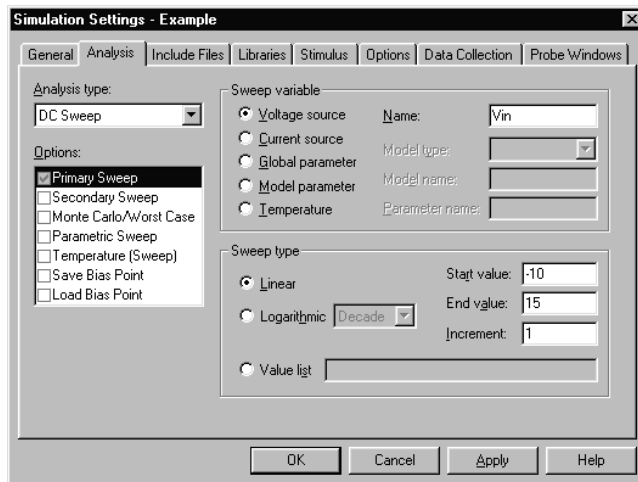


Figure 6 *DC sweep analysis settings.*

- 6 Click OK to close the Simulation Settings dialog box.
- 7 From the File menu, choose Save.
- 8 From the PSpice menu, choose Run to run the analysis.



Displaying DC analysis results

Probe windows can appear during or after the simulation is finished.

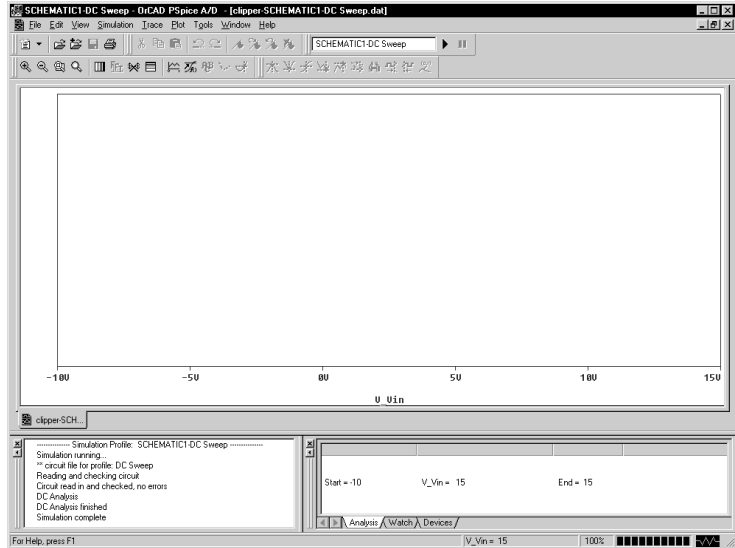


Figure 7 Probe window.

To plot voltages at nets In and Mid

press **Insert**



- 1 From PSpice's Trace menu, choose Add Trace.
- 2 In the Add Traces dialog box, select V(In) and V(Mid).
- 3 Click OK.

To display a trace using a marker

press **Ctrl+M**

- 1 From Capture's PSpice menu, point to Markers and choose Voltage Level.
- 2 Click to place a marker on net Out, as shown in Figure 8.

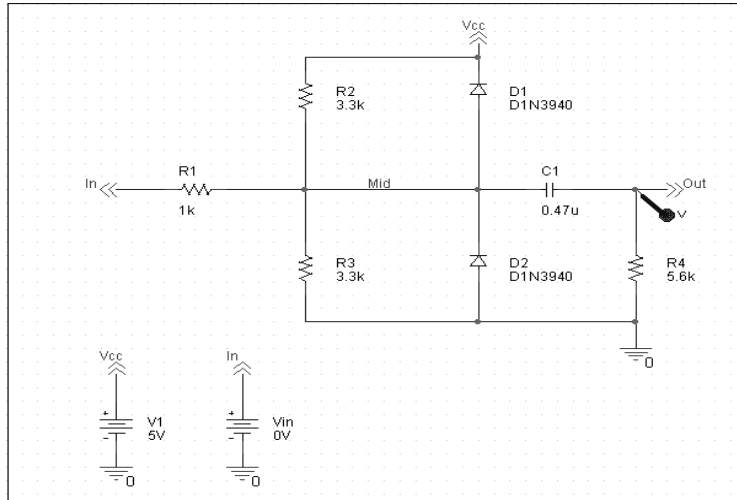


Figure 8 Clipper circuit with voltage marker on net Out.

- 3 Right-click and choose End Mode to stop placing markers.
- 4 From the File menu, choose Save.
- 5 Switch to PSpice. The V(Out) waveform trace appears, as shown in Figure 9.

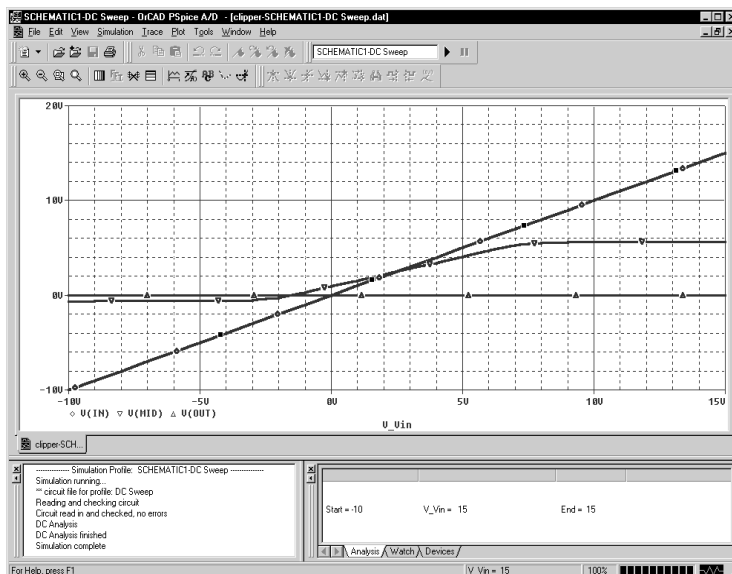


Figure 9 Voltage at In, Mid, and Out.

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

Table 10 Association of cursors with mouse buttons.

cursor 1	left mouse button
cursor 2	right mouse button

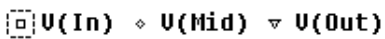


Figure 11 Trace legend with cursors activated.

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

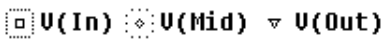


Figure 12 Trace legend with V(Mid) symbol outlined.

To place cursors on V(In) and V(Mid)

- 1 From PSpice’s Trace menu, point to Cursor and choose Display.

Two cursors appear for the first trace defined in the legend below the x-axis—V(In) in this example. The Probe Cursor window also appears.
- 2 To display the cursor crosshairs:
 - a Position the mouse anywhere inside the Probe window.
 - b Click to display the crosshairs for the first cursor.
 - c Right-click to display the crosshairs for the second cursor.
In the trace legend, the part for V(In) is outlined in the crosshair pattern for each cursor, resulting in a dashed line as shown in Figure 11.
- 3 Place the first cursor on the V(In) waveform:
 - a Click the portion of the V(In) trace in the proximity of 4 volts on the x-axis. The cursor crosshair appears, and the current X and Y values for the first cursor appear in the cursor window.
 - b To fine-tune the cursor location to 4 volts on the x-axis, drag the crosshairs until the x-axis value of the A1 cursor in the cursor window is approximately 4.0. You can also press  and  for tighter control.
- 4 Place the second cursor on the V(Mid) waveform:
 - a Right-click the trace legend part (diamond) for V(Mid) to associate the second cursor with the Mid waveform. The crosshair pattern for the second cursor outlines the V(Mid) trace part as shown in Figure 12.
 - b Right-click the portion on the V(Mid) trace that is in the proximity of 4 volts on the x-axis. The X and Y values for the second cursor appear in the cursor window along with the difference (dif) between the two cursors’ X and Y values.

- c To fine-tune the location of the second cursor to 4 volts on the x-axis, drag the crosshairs until the x-axis value of the A2 cursor in the cursor window is approximately 4.0. You can also press **Shift**+**→** and **Shift**+**←** for tighter control.

Figure 13 shows the Probe window with both cursors placed.

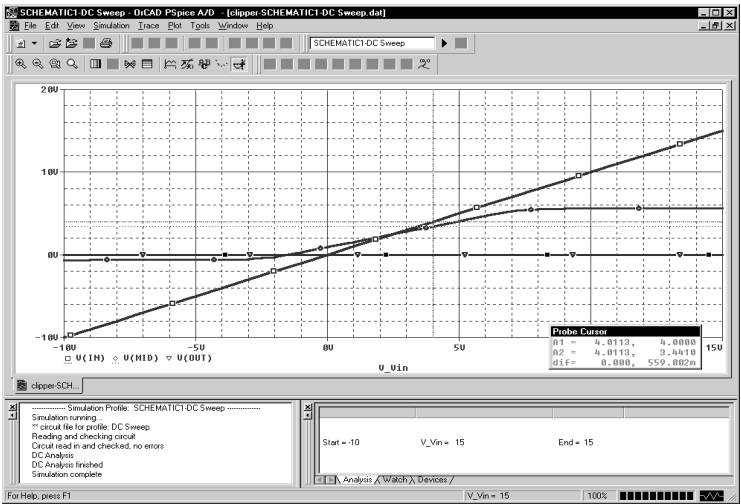


Figure 13 Voltage difference at $V(In) = 4$ volts.

To delete all of the traces

- 1 From the Trace menu, choose Delete All Traces.

At this point, the design has been saved. If needed, you can quit Capture and PSpice and complete the remaining analysis exercises later using the saved design.

There are also ways to display the difference between two voltages as a trace:

- In PSpice, add the trace expression $V(In)-V(Mid)$.
- In Capture, from the PSpice menu, point to Markers and choose Voltage Differential. Place the two markers on different pins or wires.

You can also delete an individual trace by selecting its name in the trace legend and then pressing **Delete**.

Example: To delete the $V(In)$ trace, click the text, $V(In)$, located under the plot's x-axis, and then press **Delete**.

Finding out more about DC sweep analysis

Table 2-1

To find out more about this...	See this...
DC sweep analysis	DC Sweep on page 9-306

Transient analysis

This example shows how to run a transient analysis on the clipper circuit. This requires adding a time-domain voltage stimulus as shown in Figure 14.

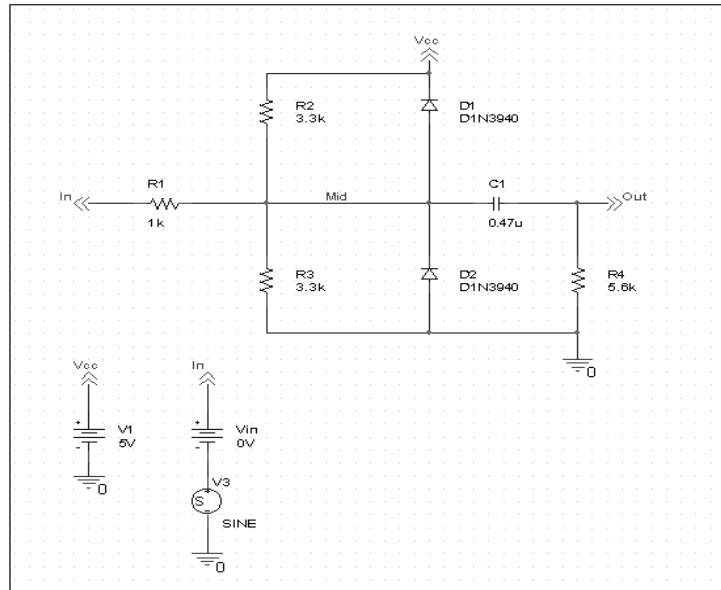


Figure 14 Diode clipper circuit with a voltage stimulus.

To add a time-domain voltage stimulus

- 1 From Capture's PSpice menu, point to Markers and choose Delete All.
- 2 Select the ground part beneath the VIN source.
- 3 From the Edit menu, choose Cut.
- 4 Scroll down (or from the View menu, point to Zoom, then choose Out).
- 5 Place a VSTIM part (from the PSpice library SOURCESTM.OLB) as shown in Figure 14.
- 6 From the Edit menu, choose Paste.
- 7 Place the ground part under the VSTIM part as shown in Figure 14.
- 8 From the View menu, point to Zoom, then choose All.
- 9 From the File menu, choose Save to save the design.



or press **Ctrl** + **V**

To set up the stimulus

- 1 Select the VSTIM part (V3).
- 2 From the Edit menu, choose PSpice Stimulus.
The New Stimulus dialog box appears.
- 3 In the New Stimulus dialog box, type SINE.
- 4 Click SIN (sinusoidal), then click OK.
- 5 In the SIN Attributes dialog box, set the first three properties as follows:

Offset Voltage = 0

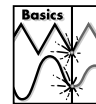
Amplitude = 10

Frequency = 1kHz

- 6 Click Apply to view the waveform.

The Stimulus Editor window should look like Figure 15.

Note The Stimulus Editor is not included in PSpice A/D Basics.



If you do not have the Stimulus Editor

- 1 Place a VSIN part instead of VSTIM and double-click it.
- 2 In the Edit Part dialog box, click User Properties.
- 3 Set values for the VOFF, VAMPL, and FREQ properties as defined in step 5. When finished, click OK.

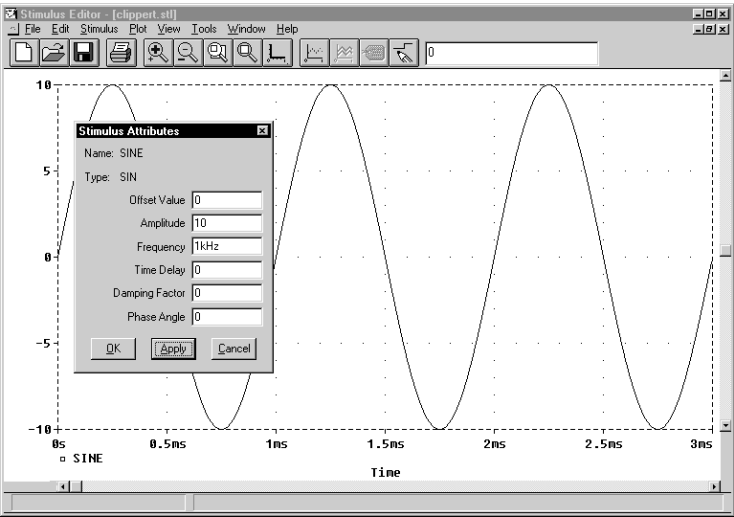


Figure 15 *Stimulus Editor window.*

- 7 Click OK.
- 8 From the File menu, choose Save to save the stimulus information. Click Yes to update the schematic.
- 9 From the File menu, choose Exit to exit the Stimulus Editor.

press **Shift** + **F12**



To set up and run the transient analysis

- 1 From Capture’s PSpice menu, choose New Simulation Profile.
The New Simulation dialog box appears.
- 2 In the Name text box, type Transient.
- 3 From the Inherit From list, select Schematic1-DC Sweep, then click Create.
The Simulation Settings dialog box appears.
- 4 Click the Analysis tab.
- 5 From the Analysis list, select Time Domain (Transient) and enter the settings shown in Figure 16.

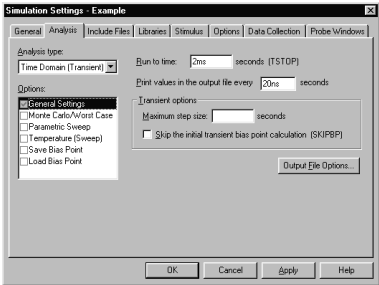


Figure 16 *Transient analysis simulation settings.*

- 6 Click OK to close the Simulation Settings dialog box.
- 7 From the PSpice menu, choose Run to perform the analysis.

PSpice A/D uses its own internal time steps for computation. The internal time step is adjusted according to the requirements of the transient analysis as it proceeds. PSpice A/D saves data to the waveform data file for each internal time step.



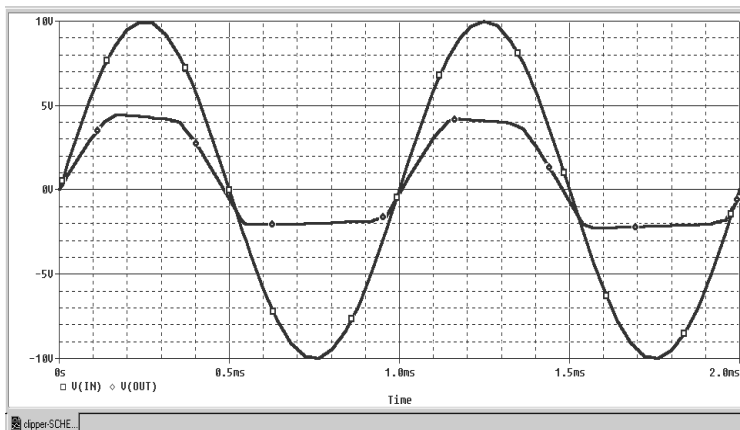
Note The internal time step is different from the Print Step value. Print Step controls how often optional text format data is written to the simulation output file (*.OUT).

To display the input sine wave and clipped wave at V(Out)

- 1 From PSpice's Trace menu, choose Add Trace.
- 2 In the trace list, select V(In) and V(Out) by clicking them.
- 3 Click OK to display the traces.
- 4 From the Tools menu, choose Options to display the Probe Options dialog box.
- 5 In the Use Symbols frame, click Always if it is not already enabled.
- 6 Click OK.



or press **Insert**



These waveforms illustrate the clipping of the input signal.

Figure 17 Sinusoidal input and clipped output waveforms.

Finding out more about transient analysis

Table 2-1

To find out more about this...	See this...
transient analysis for analog and mixed-signal designs*	Chapter 11, Transient analysis
transient analysis for digital designs*	Chapter 14, Digital simulation

* Includes how to set up time-based stimuli using the Stimulus Editor.

AC sweep analysis

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

Setting up and running an AC sweep analysis

In this example, you will set up the clipper circuit for AC analysis by adding an AC voltage source for a stimulus signal (see Figure 18) and by setting up AC sweep parameters.

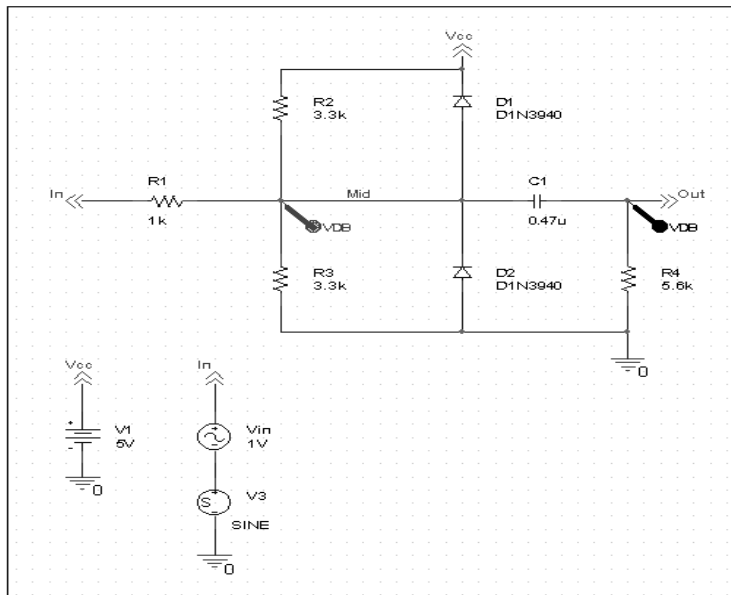


Figure 18 Clipper circuit with AC stimulus.

To change Vin to include the AC stimulus signal

- 1 In Capture, open CLIPPER.OPJ.
- 2 Select the DC voltage source, Vin, and press Delete to remove the part from the schematic page.

- 3 From the Place menu, choose Part.
- 4 In the Part text box, type VAC (from the PSpice library SOURCE.OLB) and click OK.
- 5 Place the AC voltage source on the schematic page, as shown in Figure 17.
- 6 Double-click the VAC part (0V) to display the Parts spreadsheet.
- 7 Change the Reference cell to V_{in} and change the ACMAG cell to 1V.
- 8 Click Apply to update the changes and then close the spreadsheet.

To set up and run the AC sweep simulation

Note PSpice simulation is not case-sensitive, so both *M* and *m* can be used as “milli,” and *MEG*, *Meg*, and *meg* can all be used for “mega.” However, waveform analysis treats *M* and *m* as mega and milli, respectively.

- 1 From Capture’s PSpice menu, choose New Simulation Profile.
- 2 In the Name text box, enter AC Sweep, then click create.

The Simulation Settings dialog box appears.

- 3 Click the Analysis tab.
- 4 From the Analysis type list, select AC Sweep / Noise and enter the settings shown in Figure 19.

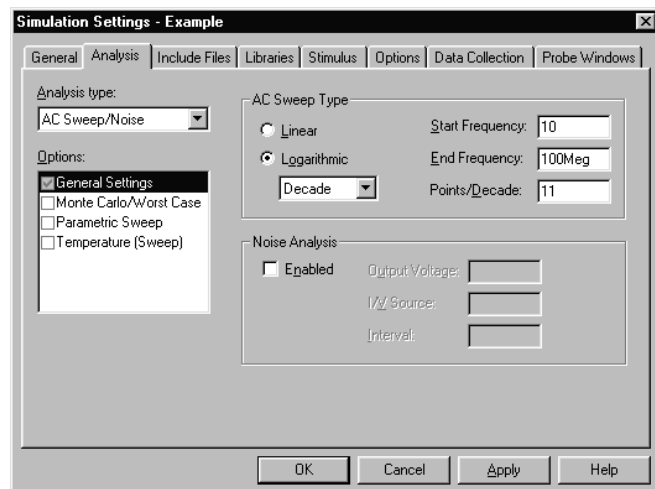


Figure 19 AC sweep and noise analysis simulation settings.

- 5 Click OK to close the Simulation Settings dialog box.
- 6 From the PSpice menu, choose Run to start the simulation.



PSpice A/D performs the AC analysis.

To add markers for waveform analysis

- 1 From Capture's PSpice menu, point to Markers, point to Advanced, then choose db Magnitude of Voltage.
- 2 Place one Vdb marker on the Out net, then place another on the Mid net.
- 3 From the File menu, choose Save to save the design.

Note You must first define a simulation profile for the AC Sweep/Noise analysis in order to use advanced markers.

AC sweep analysis results

PSpice displays the dB magnitude ($20\log_{10}$) of the voltage at the marked nets, Out and Mid, in a Probe window as shown in Figure 20 below. VDB(Mid) has a lowpass response due to the diode capacitances to ground. The output capacitance and load resistor act as a highpass filter, so the overall response, illustrated by VDB(out), is a bandpass response. Because AC is a linear analysis and the input voltage was set to 1V, the output voltage is the same as the gain (or attenuation) of the circuit.

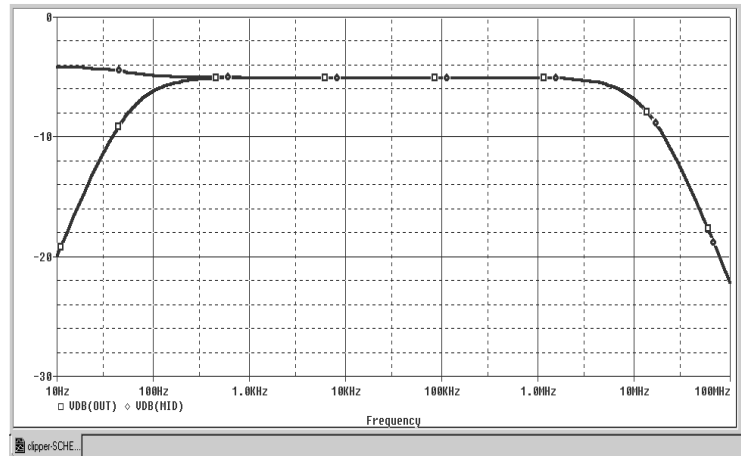


Figure 20 dB magnitude curves for “gain” at Mid and Out.

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

For more information on Probe windows and trace expressions, see [Chapter 17, Analyzing waveforms](#).

press **Ctrl** + **X**



press **Ctrl** + **V**



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

- 1 From Capture’s PSpice menu, point to Markers, point to Advanced and choose Phase of Voltage.
- 2 Place a Vphase marker on the output next to the Vdb marker.
- 3 Delete the Vdb marker on Mid.
- 4 Switch to PSpice.

In the Probe window, the gain and phase plots both appear on the same graph with the same scale.

- 5 Click the trace name VP(Out) to select the trace.
- 6 From the Edit menu, choose Cut.
- 7 From the Plot menu, choose Add Y Axis.
- 8 From the Edit menu, choose Paste.

The Bode plot appears, as shown in Figure 21.

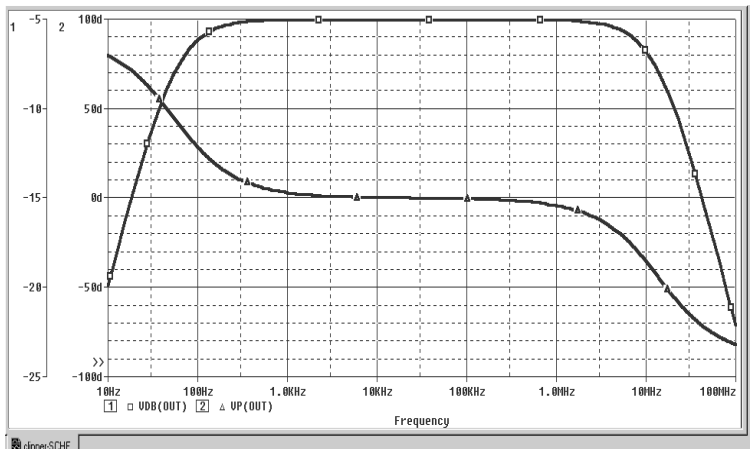
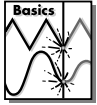


Figure 21 Bode plot of clipper’s frequency response.

Finding out more about AC sweep and noise analysis

Table 2-2

To find out more about this...	See this...
AC sweep analysis	AC sweep analysis on page 10-324
noise analysis based on an AC sweep analysis	Noise analysis on page 10-333



Note Parametric Analysis is not supported in PSpice A/D Basics.

Parametric analysis

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

- Changing the value of R1 to the expression {Rval}.
- Placing a PARAM part to declare the parameter Rval.
- Setting up and running a parametric analysis to step the value of R1 using Rval.

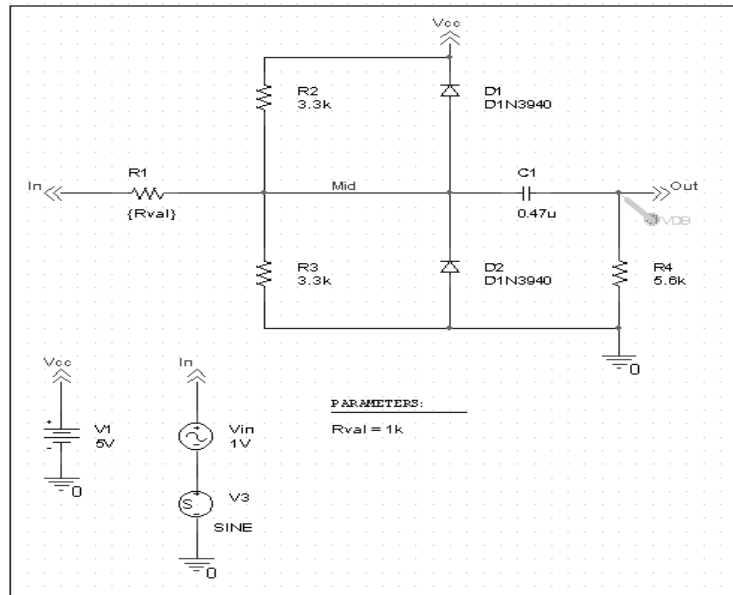


Figure 22 Clipper circuit with global parameter Rval.

This example produces multiple analysis runs, each with a different value of R1. After the analysis is complete, you can analyze curve families for the analysis runs using PSpice A/D.

Setting up and running the parametric analysis

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

- 1 In Capture, open CLIPPER.OPJ.
- 2 Double-click the value (1k) of part R1 to display the Display Properties dialog box.
- 3 In the Value text box, replace 1k with {Rval}.
- 4 Click OK.

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

To add a PARAM part to declare the parameter Rval

- 1 From Capture's Place menu, choose Part.
- 2 In the Part text box, type PARAM (from the PSpice library SPECIAL.OLB) , then click OK.
- 3 Place one PARAM part in any open area on the schematic page.
- 4 Double-click the PARAM part to display the Parts spreadsheet, then click New.
- 5 In the Property Name text box, enter Rval (no curly braces), then click OK.

This creates a new property for the PARAM part, as shown by the new column labeled Rval in the spreadsheet.

- 6 Click in the cell below the Rval column and enter 1k as the initial value of the parametric sweep.
- 7 While this cell is still selected, click Display.
- 8 In the Display Format frame, select Name and Value, then click OK.
- 9 Click Apply to update all the changes to the PARAM part.
- 10 Close the Parts spreadsheet.
- 11 Select the VP marker and press Delete to remove the marker from the schematic page.
- 12 From the File menu, choose Save to save the design.

Note For more information about using the Parts spreadsheet, see the OrCAD Capture User's Guide.

This example is only interested in the magnitude of the response.

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

- 1 From Capture’s PSpice menu, choose New Simulation Profile.

The New Simulation dialog box appears.

- 2 In the Name text box, type Parametric.
- 3 From the Inherit From list, select AC Sweep, then click Create.

The Simulation Settings dialog box appears.

- 4 Click the Analysis tab.
- 5 Under Options, select Parametric Sweep and enter the settings as shown below.

The root schematic listed is the schematic page associated with the simulation profile you are creating.

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

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

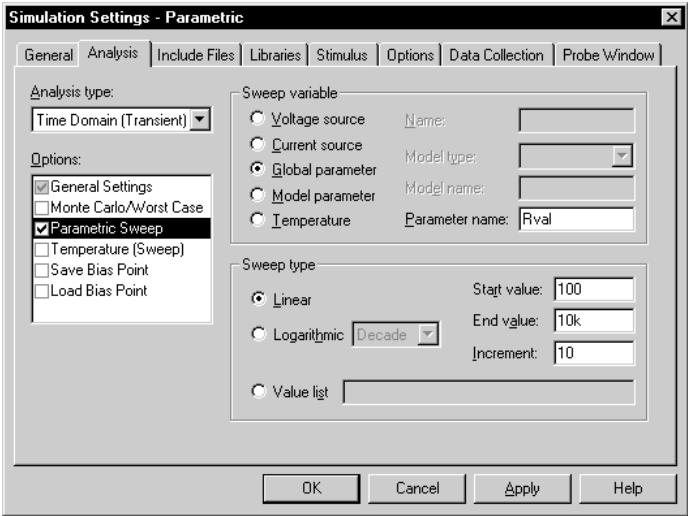


Figure 23 Parametric simulation settings.

- 6 Click OK.
- 7 From the PSpice menu, choose Run to start the analysis.



Analyzing waveform families

Continuing from the example above, there are 21 analysis runs, each with a different value of R1. After PSpice A/D completes the simulation, the Available Sections dialog box appears, listing all 21 runs and the Rval parameter value for each. You can select one or more runs to display.

To display all 21 traces

- 1 In the Available Sections dialog box, click OK.

All 21 traces (the entire family of curves) for VDB(Out) appear in the Probe window as shown in Figure 24.

To select individual runs, click each one separately.

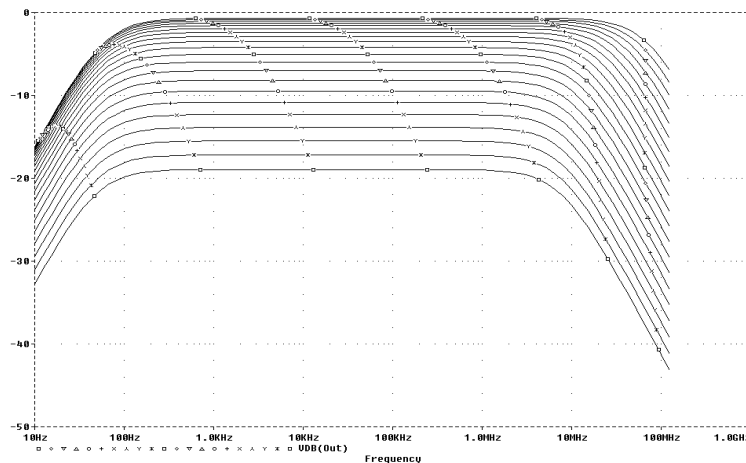


Figure 24 *Small signal response as R1 is varied from 100Ω to 10 kΩ*

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

- 2 Click the trace name to select it, then press **Delete** to remove the traces shown.

You can also remove the traces by removing the VDB marker from your schematic page in Capture.

press **Insert**

You can avoid some of the typing for the Trace Expression text box by selecting V(OUT) twice in the trace list and inserting text where appropriate in the resulting Trace Expression.

press **Insert**

The search command tells PSpice to search for the point on the trace where the x-axis value is 100.

To compare the last run to the first run

- 1 From the Trace menu, choose Add Trace to display the Add Traces dialog box.
- 2 In the Trace Expression text box, type the following:
`Vdb(Out)@1 Vdb(Out)@21`
- 3 Click OK.

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

- 4 Plot the new trace by specifying a waveform expression:
 - a From the Trace menu, choose Add Trace.
 - b In the Trace Expression text box, type the following waveform expression:
`Vdb(Out)@1 - Vdb(OUT)@21`
 - c Click OK.
- 5 Use the search commands to find the value of the difference trace at its maximum and at a specific frequency:
 - a From the Tools menu, point to Cursor and choose Display.
 - b Right-click then left-click the trace part (triangle) for `Vdb(Out)@1 - Vdb(Out)@21`. Make sure that you left-click last to make cursor 1 the active cursor.
 - c From the Trace menu, point to Cursor and choose Max.
 - d From the Trace menu, point to Cursor and choose Search Commands.
 - e In the Search Command text box, type the following:
`search forward x value (100)`
 - f Select 2 as the Cursor to Move option.

g Click OK.

Figure 25 shows the Probe window with cursors placed.

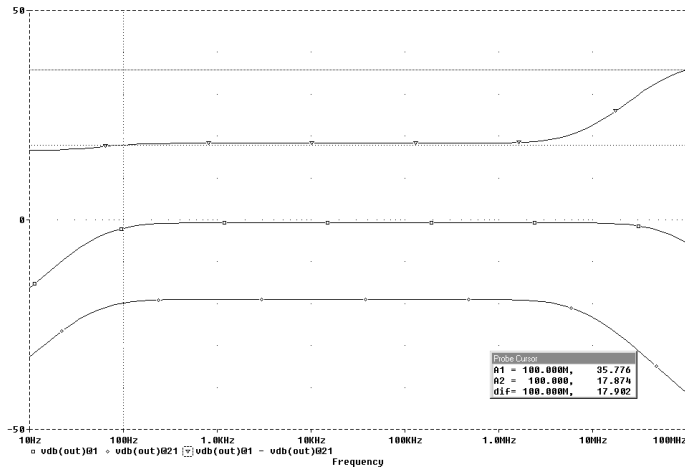


Figure 25 Small signal frequency response at 100 and 10 k Ω input resistance.

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

- 6 From the Trace menu, point to Cursor and choose Display to turn off the display of the cursors.
- 7 Delete the trace.



Finding out more about parametric analysis

Table 2-3

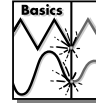
To find out more about this...	See this...
parametric analysis	<u>Parametric analysis on page 12-364</u>
using global parameters	<u>Using global parameters and expressions for values on page 3-107</u>

Performance analysis

Performance analysis is an advanced feature in PSpice A/D that you can use to compare the characteristics of a family of waveforms. Performance analysis uses the principle of search commands introduced earlier in this chapter to define functions that detect points on each curve in the family.

After you define these functions, you can apply them to a family of waveforms and produce traces that are a function of the variable that changed within the family.

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



Note Performance Analysis is not supported in PSpice A/D Basics.

To plot bandwidth vs. Rval using the performance analysis wizard

- 1 In Capture, open CLIPPER.OPJ.
- 2 From PSpice's Trace menu, choose Performance Analysis.

The Performance Analysis dialog box appears with information about the currently loaded data and performance analysis in general.
- 3 Click the Wizard button.
- 4 Click the Next> button.
- 5 In the Choose a Goal Function list, click Bandwidth, then click the Next> button.
- 6 Click in the Name of Trace to search text box and type V(Out).
- 7 Click in the db level down for bandwidth calc text box and type 3.
- 8 Click the Next> button.

At each step, the wizard provides information and guidelines.

Click , then double-click V(Out).

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

Double-click the x-axis.

- 9 Click the Next> button or the Finish button.
A plot of the 3dB bandwidth vs. Rval appears.
- 10 Change the x-axis to log scale:
 - a From the Plot menu, choose Axis Settings.
 - b Click the X Axis tab.
 - c Under Scale, choose Log.
 - d Click OK.

To plot gain vs. Rval manually



or press **Insert**

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

- 1 From the Plot menu, choose Add Y Axis.
- 2 From the Trace menu, choose Add to display the Add Traces dialog box.
- 3 In the Functions or Macros frame, select the Goal Functions list, and then click the Max(1) goal function.
- 4 In the Simulation Output Variables list, click V(out).
- 5 In the Trace Expression text box, edit the text to be `Max(Vdb(out))`, then click OK.

PSpice displays gain on the second y-axis vs. Rval.

Figure 26 shows the final performance analysis plot of 3dB bandwidth and gain in dB vs. the swept input resistance value.

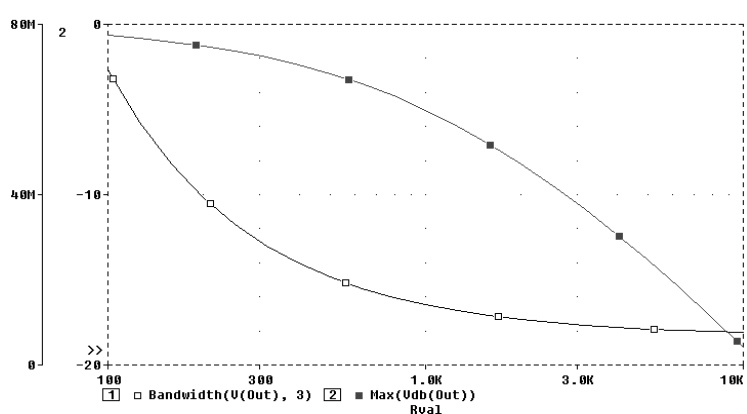


Figure 26 Performance analysis plots of bandwidth and gain vs. Rval.

Finding out more about performance analysis

Table 2-4

To find out more about this...	See this...
how to use performance analysis	RLC filter example on page 12-366 Example: Monte Carlo analysis of a pressure sensor on page 13-385
how to use search commands and create goal functions	PSpice A / D online Help

Part two

Design entry

Part two provides information about how to enter circuit designs in OrCAD® Capture that you want to simulate.

- [Chapter 3, Preparing a design for simulation](#), outlines the things you need to do to successfully simulate your schematic including troubleshooting tips for the most frequently asked questions.
- [Chapter 4, Creating and editing models](#), describes how to use the tools to create and edit model definitions, and how to configure the models for use.
- [Chapter 5, Creating parts for models](#), explains how to create symbols for existing or new model definitions so you can use the models when simulating from your schematic.
- [Chapter 6, Analog behavioral modeling](#), describes how to model analog behavior mathematically or using table lookups.
- [Chapter 7, Digital device modeling](#), explains the structure of digital subcircuits and how to create your own from primitives.

Preparing a design for simulation

3

Chapter overview

This chapter provides introductory information to help you enter circuit designs that simulate properly. If you want an overview, use the checklist on page [3-96](#) to guide you to specific topics.

Refer to your *OrCAD Capture User's Guide* for general schematic entry information.

Topics include:

- [Checklist for simulation setup on page 3-96](#)
- [Using parts that you can simulate on page 3-100](#)
- [Using global parameters and expressions for values on page 3-107](#)
- [Defining power supplies on page 3-114](#)
- [Defining stimuli on page 3-116](#)
- [Things to watch for on page 3-120](#)

Checklist for simulation setup

This section describes what you need to do to set up your circuit for simulation.

- 1 Find the topic that is of interest in the first column of any of these tables.
- 2 Go to the referenced section. For those sections that provide overviews, you will find references to more detailed discussions.

Typical simulation setup steps

For more information on this step...	See this...	To find out this...
✓ Set component values and other properties.	Using parts that you can simulate on page 3-100 Using global parameters and expressions for values on page 3-107	An overview of vendor, passive, breakout, and behavioral parts. How to define values using variable parameters, functional calls, and mathematical expressions.
✓ Define power supplies.	Defining power supplies on page 3-114	An overview of DC power for analog circuits and digital power for mixed-signal circuits.
✓ Define input waveforms.	Defining stimuli on page 3-116	An overview of DC, AC, and time-based stimulus parts.
✓ Set up one or more analyses.	Chapter 8, Setting up analyses and starting simulation Chapter 9 through Chapter 14 (see the table of contents)	Procedures, general to all analysis types, to set up and start the simulation. Detailed information about DC, AC, transient, parametric, temperature, Monte Carlo, sensitivity / worst-case, and digital analyses.

For more information on this step...	See this...	See this...
✓ Place markers.	Using schematic page markers to add traces on page 17-487	How to display results in PSpice by picking design nets.
✓	Limiting waveform data file size on page 17-490	How to limit the data file size.

Advanced design entry and simulation setup steps

For more information on this step...	See this...	To find out how to...
✓ Create new models.	Chapter 4, Creating and editing models	Define models using the Model Editor or Create Subcircuit command.
	Chapter 6, Analog behavioral modeling	Define the behavior of a block of analog circuitry as a mathematical function or lookup table.
	Chapter 7, Digital device modeling	Define the functional, timing, and I/O characteristics of a digital part.
✓ Create new parts.	Chapter 5, Creating parts for models	Create parts either automatically for models using the part wizard or the Parts utility, or by manually defining AKO parts; define simulation-specific properties.
	<i>The OrCAD Capture User's Guide</i>	Create and edit part graphics, pins, and properties in general.

When netlisting fails or the simulation does not start

If you have problems starting the simulation, there may be problems with the design or with system resources. If there are problems with the design, PSpice A/D displays errors and warnings in the Simulation Output window. You can use the Simulation Output window to get more information quickly about the specific problem.

To get online information about an error or warning shown in the Simulation Output window

- 1 Select the error or warning message.
- 2 Press **F1**.

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

Things to check in your design

Table 5

Make sure that...	To find out more, see this...
✓ The model libraries, stimulus files, and include files are configured.	Configuring model libraries on page 4-162
✓ The parts you are using have models.	Unmodeled parts on page 3-120 and Defining part properties needed for simulation on page 5-181
✓ You are not using unmodeled pins.	Unmodeled pins on page 3-123
✓ You have defined the grounds.	Missing ground on page 3-124
✓ Every analog net has a DC path to ground.	Missing DC path to ground on page 3-125
✓ The part template is correct.	Defining part properties needed for simulation on page 5-181
✓ Hierarchical parts, if used, are properly defined.	The <i>OrCAD Capture User's Guide</i>
✓ Ports that connect to the same net have the same name.	The <i>OrCAD Capture User's Guide</i>

Things to check in your system configuration

Table 6

Make sure that...	To find out more, see this...
✓ Path to the PSpice A / D programs is correct.	
✓ Directory containing your design has write permission.	Your operating system manual
✓ Your system has sufficient free memory and disk space.	Your operating system manual

Using parts that you can simulate

The OrCAD part libraries also include special parts that you can use for simulation only. These include:

- **stimulus parts** to generate input signals to the circuit (see [Defining stimuli on page 3-116](#))
- **ground parts** required by all analog and mixed-signal circuits, which need reference to ground
- **simulation control parts** to do things like set bias values (see [Appendix A, Setting initial state](#))
- **output control parts** to do things like generate tables and line-printer plots to the PSpice output file (see [Chapter 18, Other output options](#))

The OrCAD part libraries supply numerous parts designed for simulation. These include:

- vendor-supplied parts
- passive parts
- breakout parts
- behavioral parts

At minimum, a part that you can simulate has these properties:

- A simulation model to describe the part's electrical behavior; the model can be:
 - explicitly defined in a model library,
 - built into PSpice A/D, or
 - built into the part (for some kinds of analog behavioral parts).
- A part with modeled pins to form electrical connections in your design.
- A translation from design part to netlist statement so that PSpice A/D can read it in.

Note *Not all parts in the libraries are set up for simulation. For example, connectors are parts destined for board layout only and do not have these simulation properties.*

Vendor-supplied parts

The OrCAD libraries provide an extensive selection of manufacturers' analog and digital parts. Typically, the library name reflects the kind of parts contained in the library and the vendor that provided the models.

Example: MOTOR_RF.OLB and MOTOR_RF.LIB contain parts and models, respectively, for Motorola-made RF bipolar transistors.

For a listing of vendor-supplied parts contained in the OrCAD libraries, refer to the online *Library List*.

To find out more about each model library, read the comments in the .LIB file header.

Part naming conventions

The part names in the OrCAD libraries usually reflect the manufacturers' part names. If multiple vendors supply the same part, each part name includes a suffix that indicates the vendor that supplied the model.

Example: The OrCAD libraries include several models for the OP-27 opamp as shown by these entries in the online *Library List*.

Device Type	Generic Name	Mfg. Name	Symbol Name	Library	Tech Type	Model	Package	New for 8.0
Operational Amplifier	OP-249	Analog Devices Inc.	OP-249G/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-260	Analog Devices Inc.	OP-260/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Analog Devices Inc.	OP-27/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Analog Devices Inc.	OP-27A/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Analog Devices Inc.	OP-27B/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Analog Devices Inc.	OP-27C/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Analog Devices Inc.	OP-27E/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Analog Devices Inc.	OP-27F/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Analog Devices Inc.	OP-27G/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Linear Technology Corp.	OP-27/LT	LIN_TECH.SLB		*	*	
Operational Amplifier	OP-27		OP-27	OPAMP.SLB		*	*	
Operational Amplifier	OP-275	Analog Devices Inc.	OP-275/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-275	Analog Devices Inc.	OP-275G/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27A	Linear Technology Corp.	OP-27A/LT	LIN_TECH.SLB		*	*	
Operational Amplifier	OP-27C	Linear Technology Corp.	OP-27C/LT	LIN_TECH.SLB		*	*	

Notice the following:

- There is a generic OP-27 part provided by OrCAD, the OP-27/AD from Analog Devices, Inc., and the OP-27/LT from Linear Technology Corporation.
- The Model column for all of these parts contains an asterisk. This indicates that this part is modeled and that you can simulate it.

Finding the part that you want

If you are having trouble finding a part, you can search the libraries for parts with similar names by using either:

- the parts browser in Capture and restricting the parts list to those names that match a specified wildcard text string, or
- the online *Library List* and searching for the generic part name using capabilities of the Adobe Acrobat Reader.

To find parts using the parts browser

- 1 In Capture, from the Place menu, choose Part.
- 2 In the Part Name text box, type a text string with wildcards that approximates the part name that you want to find. Use this syntax:

`<wildcard><part_name_fragment><wildcard>`

where `<wildcard>` is one of the following:

- * to match zero or more characters
- ? to match exactly one character

The parts browser displays only the matching part names.

Note This method finds any part contained in the current part libraries configuration, including parts for user-defined models.

If you want to find out more about a part supplied in the OrCAD libraries, such as manufacturer or whether you can simulate it, then search the online *Library List* (see page [3-103](#)).

To find parts using the online OrCAD Library List

- 1 In Windows Explorer, double-click LIBLIST.PDF, located in the directory where PSpice A / D is installed. Acrobat Reader starts and displays the OrCAD Library List.
- 2 From the Tools menu, choose Find.
- 3 In the Find What text box, type the generic part name.
- 4 Enter any other search criteria, and then click Find.
The Acrobat Reader displays the first page where it finds a match. Each page maps the generic part name to the parts (and corresponding vendor and part library name) in the OrCAD libraries.
- 5 If you want to repeat the search, from the Tools menu, choose Find Again.

Note *If you are unsure of the device type, you can scan all of the device type lists using the Acrobat search capability. The first time you do this, you need to set up the across-list index. To find out more, refer to the online Adobe Acrobat manuals.*

Note *This method finds only parts that OrCAD supplies that have models.*

If you want to include user-defined parts in the search, use the parts browser in Capture (see page [3-102](#)).



or press **Ctrl** + **F**

Instead of the generic part name, you can enter other kinds of search information, such as device type or manufacturer.

press **Ctrl** + **G**

Passive parts

The OrCAD libraries supply several basic parts based on the passive device models built into PSpice A/D. These are summarized in the following table.

Table 7 *Passive parts*

To find out more about how to use these parts and define their properties, look up the corresponding PSpice device letter in the *Analog Devices* chapter in the online *OrCAD PSpice A/D Reference Manual*, and then see the *Capture Parts* sections.

These parts are available...	For this device type...	Which is this PSpice device letter...
C C_VAR	capacitor	C
L	inductor	L
R R_VAR	resistor	R
XFRM_LINEAR K_LINEAR	transformer	K and L
T	ideal transmission line	T
TLOSSY*	Lossy transmission line	T
TnCOUPLED** TnCOUPLEDX** KCOUPLEN**	coupled transmission line	T and K

* TLOSSY is not available in Basics+ packages.

** For these device types, the OrCAD libraries supply several parts. Refer to the online *OrCAD PSpice A/D Reference Manual* for the available parts.

Breakout parts

The OrCAD libraries supply passive and semiconductor parts with default model definitions that define a basic set of model parameters. This way, you can easily:

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

These are called breakout parts and are summarized in the following table.

Table 8 Breakout parts

Use this breakout part...	For this device type...	Which is this PSpice device letter...
BBREAK	GaAsFET	B
CBREAK	capacitor	C
DBREAKx*	diode	D
JBREAKx*	JFET	J
KBREAK	inductor coupling	K
LBREAK	inductor	L
MBREAKx*	MOSFET	M
QBREAKx*	bipolar transistor	Q
RBREAK	resistor	R
SBREAK	voltage-controlled switch	S
TBREAK	transmission line	T
WBREAK	current-controlled switch	W
XFRM_NONLINEAR	transformer	K and L
ZBREAKN	IGBT	Z

* For this device type, the OrCAD libraries supply several breakout parts. Refer to the online *OrCAD PSpice A/D Reference Manual* for the available parts.

To find out more about models, see [What are models? on page 4-129](#).

To find out more about Monte Carlo and sensitivity/worst-case analyses, see [Chapter 13, Monte Carlo and sensitivity/worst-case analyses](#).

To find out more about setting temperature parameters, see the *Analog Devices* chapter in the online *OrCAD PSpice A/D Reference Manual* and find the device type that you are interested in.

To find out more about how to use these parts and define their properties, look up the corresponding PSpice device letter in the *Analog Devices* chapter of the online *OrCAD PSpice A/D Reference Manual*, and then look in the *Capture Parts* section.

Behavioral parts

Behavioral parts allow you to define how a block of circuitry should work without having to define each discrete component.

For more information, see [Chapter 6, Analog behavioral modeling](#).

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

For more information, see:

- [Chapter 7, Digital device modeling](#)
- the *Digital Devices* chapter in the online *OrCAD PSpice A/D Reference Manual*

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

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

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

Using global parameters and expressions for values

In addition to literal values, you can use global parameters and expressions to represent numeric values in your circuit design.

Global parameters

A global parameter is like a programming variable that represents a numeric value by name.

Once you have defined a parameter (declared its name and given it a value), you can use it to represent circuit values anywhere in the design; this applies to any hierarchical level.

Some ways that you can use parameters are as follows:

- Apply the same value to multiple part instances.
- Set up an analysis that sweeps a variable through a range of values (for example, DC sweep or parametric analysis).

When multiple parts are set to the same value, global parameters provide a convenient way to change all of their values for “what-if” analyses.

Example: If two independent sources have a value defined by the parameter VSUPPLY, then you can change both sources to 10 volts by assigning the value *once* to VSUPPLY.

Declaring and using a global parameter

To use a global parameter in your design, you need to:

- define the parameter using a PARAM part, and
- use the parameter in place of a literal value somewhere in your design.

Note For more information about using the Parts spreadsheet, see the OrCAD Capture User's Guide.

Example: To declare the global parameter VSUPPLY that will set the value of an independent voltage source to 14 volts, place the PARAM part, and then create a new property named VSUPPLY with a value of 14v.

To declare a global parameter

- 1 Place a PARAM part in your design.
- 2 Double-click the PARAM part to display the Parts spreadsheet, then click New.
- 3 Declare up to three global parameters by doing the following for each global parameter:
 - a Click New.
 - b In the Property Name text box, enter NAME n , then click OK.
This creates a new property for the PARAM part, NAME n in the spreadsheet.
 - c Click in the cell below the NAME n column and enter a default value for the parameter.
 - d While this cell is still selected, click Display.
 - e In the Display Format frame, select Name and Value, then click OK.
- Note** The system variables in [Table 11 on page 3-113](#) have reserved parameter names. Do not use these parameter names when defining your own parameters.
- 4 Click Apply to update all the changes to the PARAM part.
- 5 Close the Parts spreadsheet.

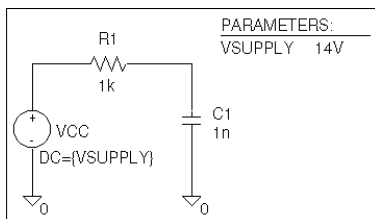
To use the global parameter in your circuit

- 1 Find the numeric value that you want to replace: a component value, model parameter value, or other property value.
- 2 Replace the value with the name of the global parameter using the following syntax:

{ global_parameter_name }

The curly braces tell PSpice A/D to evaluate the parameter and use its value.

Example: To set the independent voltage source, VCC, to the value of the VSUPPLY parameter, set its DC property to {VSUPPLY}.



Expressions

An expression is a mathematical relationship that you can use to define a numeric or boolean (TRUE / FALSE) value.

PSpice A/D evaluates the expression to a single value every time:

- it reads in a new circuit, and
- a parameter value used within an expression changes during an analysis.

Example: A parameter that changes with each step of a DC sweep or parametric analysis.

Specifying expressions

To use an expression in your circuit

- 1 Find the numeric or boolean value you want to replace: a component value, model parameter value, other property value, or logic in an IF function test (see page [3-112](#) for a description of the IF function).
- 2 Replace the value with an expression using the following syntax:

{ *expression* }

where *expression* can contain any of the following:

- standard operators listed in [Table 9](#)
- built-in functions listed in [Table 10](#)
- user-defined functions
- system variables listed in [Table 11](#)
- user-defined global parameters
- literal operands

The curly braces tell PSpice A/D to *evaluate* the expression and use its value.

Example: Suppose you have declared a parameter named FACTOR (with a value of 1.2) and want to scale a -10 V independent voltage source, VEE, by the value of FACTOR. To do this, set the DC property of VEE to:

{-10*FACTOR}

PSpice A/D evaluates this expression to:

(-10 * 1.2) or -12 volts

For more information on user-defined functions, see the .FUNC command in the *Commands* chapter in the online *OrCAD PSpice A/D Reference Manual*.

For more information on user-defined parameters, see [Using global parameters and expressions for values on page 3-107](#).

Table 9 *Operators in expressions*

This operator class...	Includes this operator...	Which means...
arithmetic	+	addition or string concatenation
	-	subtraction
	*	multiplication
	/	division
	**	exponentiation
logical*	~	unary NOT
		boolean OR
	^	boolean XOR
	&	boolean AND
relational*	==	equality test
	!=	non-equality test
	>	greater than test
	>=	greater than or equal to test
	<	less than test
	<=	less than or equal to test

* Logical and relational operators are used within the IF() function; for digital parts, logical operators are used in Boolean expressions.

Table 10 Functions in arithmetic expressions

This function...	Means this...	
ABS(x)	$ x $	
SQRT(x)	$x^{1/2}$	
EXP(x)	e^x	
LOG(x)	$\ln(x)$	which is log base e
LOG10(x)	$\log(x)$	which is log base 10
PWR(x,y)	$ x ^y$	
PWRS(x,y)	$+ x ^y$ (if $x > 0$) $- x ^y$ (if $x < 0$)	
SIN(x)	$\sin(x)$	where x is in radians
ASIN(x)	$\sin^{-1}(x)$	where the result is in radians
SINH(x)	$\sinh(x)$	where x is in radians
COS(x)	$\cos(x)$	where x is in radians
ACOS(x)	$\cos^{-1}(x)$	where the result is in radians
COSH(x)	$\cosh(x)$	where x is in radians
TAN(x)	$\tan(x)$	where x is in radians
ATAN(x)	$\tan^{-1}(x)$	where the result is in radians
ARCTAN(x)		
ATAN2(y,x)	$\tan^{-1}(y/x)$	where the result is in radians
TANH(x)	$\tanh(x)$	where x is in radians
M(x)	magnitude of x^*	which is the same as ABS(x)
P(x)	phase of x^*	in degrees; returns 0.0 for real numbers
R(x)	real part of x^*	
IMG(x)	imaginary part of x^*	which is applicable to AC analysis only

Table 10 *Functions in arithmetic expressions (continued)*

	This function...	Means this...	
<i>Note</i> In waveform analysis, this function is $D(x)$.	DDT(x)	time derivative of x	which is applicable to transient analysis only
<i>Note</i> In waveform analysis, this function is $S(x)$.	SDT(x)	time integral of x	which is applicable to transient analysis only
	TABLE(x,x ₁ ,y ₁ ,...)	y value as a function of x	where x _n ,y _n point pairs are plotted and connected by straight lines
	MIN(x,y)	minimum of x and y	
	MAX(x,y)	maximum of x and y	
	LIMIT(x,min,max)	min if x < min max if x > max else x	
	SGN(x)	+1 if x > 0 0 if x = 0 -1 if x < 0	
Example: {v(1)*STP(TIME-10ns)} gives a value of 0.0 until 10 nsec has elapsed, then gives v(1).	STP(x)	1 if x > 0 0 otherwise	which is used to suppress a value until a given amount of time has passed
	IF(t,x,y)	x if t is true y otherwise	where t is a relational expression using the relational operators shown in Table 9

* M(x), P(x), R(x), and IMG(x) apply to Laplace expressions only.

Table 11 *System variables*

This variable...	Evaluates to this...
TEMP	Temperature values resulting from a temperature, parametric temperature, or DC temperature sweep analysis. The default temperature, TNOM, is set in the Options dialog box (from the Simulation Settings dialog box, choose the Options tab). TNOM defaults to 27°C. Note <i>TEMP can only be used in expressions pertaining to analog behavioral modeling and the propagation delay of digital models.</i>
TIME	Time values resulting from a transient analysis. If no transient analysis is run, this variable is undefined. Note <i>TIME can only be used in analog behavioral modeling expressions.</i>

Note *If a passive or semiconductor device has an independent temperature assignment, then TEMP does not represent that device's temperature.*

To find out more about customizing temperatures for passive or semiconductor devices, refer to the .MODEL command in the *Commands* chapter in the online *OrCAD PSpice A/D Reference Manual*.

Defining power supplies

For the analog portion of your circuit

If the analog portion of your circuit requires DC power, then you need to include a DC source in your design. To specify a DC source, use one of the following parts.

To find out how to use these parts and specify their properties, see the following:

- [Setting up a DC stimulus on page 9-310](#)
- [Using VSRC or ISRC parts on page 3-119](#)

Table 12

For this source type...	Use this part...
voltage	VDC or VSRC
current	IDC or ISRC

For A/D interfaces in mixed-signal circuits

Default digital power supplies

Every digital part supplied in the OrCAD libraries has a default digital power supply defined for its A-to-D or D-to-A interface subcircuit. This means that if you are designing a mixed-signal circuit, then you have a default 5 volt digital power supply built-in to the circuit at every interface.

Custom digital power supplies

If needed, you can customize the power supply for different logic families.

Table 13

For this logic family...	Use this part...
CD4000	CD4000_PWR
TTL	DIGIFPWR
ECL 10K	ECL_10K_PWR
ECL 100K	ECL_100K_PWR

To find out how to use these parts and specify their digital power and ground pins, see [Specifying digital power supplies on page 15-449](#).

Defining stimuli

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

The OrCAD libraries supply several source parts that are described in the tables that follow. These parts depend on:

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

Analog stimuli

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

Table 14

If you want this kind of input...	Use this part for voltage...
For DC analyses	
DC bias	VDC or VSRC
For AC analyses	
AC magnitude and phase	VAC or VSRC
For transient analyses	
exponential	VEXP or VSTIM*
periodic pulse	VPULSE or VSTIM*
piecewise-linear	VPWL or VSTIM*
piecewise-linear that repeats forever	VPWL_RE_FOREVER or VPWL_F_RE_FOREVER**

See [Setting up a DC stimulus on page 9-310](#) for more details.

See [Setting up an AC stimulus on page 10-325](#) for more details.

See [Defining a time-based stimulus on page 11-344](#) for more details.

Table 14

If you want this kind of input...	Use this part for voltage...
piecewise-linear that repeats n times	VPWL_N_TIMES or VPWL_F_N_TIMES**
frequency-modulated sine wave	VSFFM or VSTIM*
sine wave	VSIN or VSTIM*

* VSTIM and ISTIM parts require the Stimulus Editor to define the input signal; these parts are not available in Basics+.

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

To determine the part name for an equivalent current source

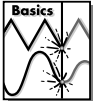
- 1 In the table of voltage source parts, replace the first V in the part name with I.

Example: The current source equivalent to VDC is IDC, to VAC is IAC, to VEXP is IEXP, and so on.

Using VSTIM and ISTIM

You can use VSTIM and ISTIM parts to define any kind of time-based input signal. To specify the input signal itself, you need to use the Stimulus Editor. See [The Stimulus Editor utility on page 11-346](#).

Note *The Stimulus Editor is not included in PSpice A/D Basics.*



If you want to specify multiple stimulus types

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

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

Using time-based stimulus parts with AC and DC properties

The time-based stimulus parts that you can use to define a transient, DC, and /or AC input signal are listed below.

VEXP	IEXP
VPULSE	IPULSE
VPWL	IPWL
VPWL_F_RE_FOREVER	IPWL_F_RE_FOREVER
VPWL_F_N_TIMES	IPWL_F_N_TIMES
VPWL_RE_FOREVER	IPWL_RE_FOREVER
VPWL_RE_N_TIMES	IPWL_RE_N_TIMES
VSFFM	ISFFM
VSIN	ISIN

For the meaning of transient source properties, refer to the I/V (independent current and voltage source) device type syntax in the *Analog Devices* chapter in the online *OrCAD PSpice A/D Reference Manual*.

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

Table 15

This property...	Has this syntax...
DC	<i>DC_value</i> [units]
AC	<i>magnitude_value</i> [units] [<i>phase_value</i>]

Using VSRC or ISRC parts

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

Table 16

This property...	Has this syntax...
DC	<i>DC_value[units]</i>
AC	<i>magnitude_value[units] [phase_value]</i>
TRAN	<i>time-based_type (parameters)</i> where <i>time-based_type</i> is EXP, PULSE, PWL, SFFM, or SIN, and the <i>parameters</i> depend on the <i>time-based_type</i> .

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

For the syntax and meaning of transient source specifications, refer to the I/V (independent current and voltage source) device type in the *Analog Devices* chapter in the online *OrCAD PSpice A/D Reference Manual*.

Digital stimuli

Table 17

If you want this kind of input...	Use this part....
For transient analyses	
signal or bus (any width)	DIGSTIMn*
clock signal	DIGCLOCK
1-bit signal	STIM1
4-bit bus	STIM4
8-bit bus	STIM8
16-bit bus	STIM16
file-based signal or bus (any width)	FILESTIMn

* The DIGSTIM part requires the Stimulus Editor to define the input signal; these parts are not available in Basics+.

You can use the DIGSTIM part to define both 1-bit signal or bus (any width) input signals using the Stimulus Editor.

See [Defining a digital stimulus on page 14-413](#) to find out more about:

- all of these source parts, and
- how to use the Stimulus Editor to specify DIGSTIMn (DIGSTIM1, DIGSTIM4, etc.) part.

Things to watch for

For a roadmap to other commonly encountered problems and solutions, see [When netlisting fails or the simulation does not start on page 3-98.](#)

This section includes troubleshooting tips for some of the most common reasons your circuit design may not netlist or simulate.

Unmodeled parts

If you see messages like this in the PSpice Simulation Output window,

```
Warning: Part part_name has no simulation model.
```

then you may have done one of the following things:

- Placed a part from the OrCAD libraries that is not available for simulation (used only for board layout).
- Placed a custom part that has been incompletely defined for simulation.

Do this if the part in question is from the OrCAD libraries

- Replace the part with an equivalent part from one of the libraries listed in the tables below.
- Make sure that you can simulate the part by checking the following:
 - That it has a PSPICETEMPLATE property and that its value is non-blank.
 - That it has an Implementation Type = PSpice MODEL property and that its Implementation property is non-blank.

The libraries listed in the tables that follow all contain parts that you can simulate. Some files also contain parts that you can only use for board layout. That's why you need to check the Pspice TEMPLATE property if you are unsure or still getting warnings when you try to simulate your circuit.

Table 18

Analog libraries with modeled parts (installed in Capture\Library\PSpice)		
1_SHOT	EPWRBJT	MOTOR_RF
ABM	FILTSUB	NAT_SEMI
ADV_LIN	FWBELL	OPAMP
AMP	HARRIS	OPTO
ANALOG	IGBT*	PHIL_BJT
ANA_SWIT	JBIPOLAR	PHIL_FET
ANLG_DEV	JDIODE	PHIL_RF
ANL_MISC	JFET	POLYFET
APEX	JJFET	PWRBJT
BIPOLAR	JOPAMP	PWRMOS
BREAKOUT	JPWRBJT	SIEMENS
BUFFER	JPWRMOS	SWIT_RAV
BURR_BRN	LIN_TECH	SWIT_REG
CD4000	MAGNETIC*	TEX_INST
COMLINR	MAXIM	THYRISTR*
DIODE	MIX_MISC**	TLINE*
EBIPOLAR	MOTORAMP	XTAL
EDIODE	MOTORMOS	ZETEX
ELANTEC	MOTORSEN	

* Not included in Basics+.

** Contains mixed-signal parts.

To find out more about a particular library, refer to the online *Library List* or read the header of the model library file itself.

Digital libraries with modeled parts		
7400	74H	DIG_ECL
74AC	74HC	DIG_GAL
74ACT	74HCT	DIG_MISC
74ALS	74L	DIG_PAL
74AS	74LS	DIG_PRIM
74F	74S	

To find out more about setting the simulation properties for parts, see [Defining part properties needed for simulation on page 5-181](#).

To find out more about using the part editor, refer to your *OrCAD Capture User's Guide*.

Check for this if the part in question is custom-built

Are there blank (or inappropriate) values for the part's Implementation and PSPICETEMPLATE properties?

If so, load this part into the part editor and set these properties appropriately. One way to approach this is to edit the part that appears in your design.

To edit the properties for the part in question

- 1 In the schematic page editor, select the part.
- 2 From the Edit menu, choose Part.

The part editor window appears with the part already loaded.
- 3 From the Edit menu, choose Properties and proceed to change the property values.

Unconfigured model, stimulus, or include files

If you see messages like these in the PSpice Simulation Output window,

(*design_name*) Floating pin: *refdes* pin
pin_name

Floating pin: *pin_id*

File not found

Can't open stimulus file

or messages like these in the PSpice output file,

Model *model_name* used by *device_name* is
undefined.

Subcircuit *subckt_name* used by *device_name*
is undefined.

Can't find .STIMULUS "*refdes*" definition

then you may be missing a model library, stimulus file, or include file from the configuration list, or the configured file is not on the library path.

Check for this

- Does the relevant model library, stimulus file, or include file appear in the configuration list?
- If the file is configured, does the default library search path include the directory path where the file resides, or explicitly define the directory path in the configuration list?

If the file is not configured, add it to the list and make sure that it appears before any other library or file that has an identically-named definition.

To view the configuration list

- 1 In the Simulation Settings dialog box, click the Include Files tab.

If the directory path is not specified, update the default library search path or change the file entry in the configuration list to include the full path specification.

To view the default library search path

- 1 In the Simulation Settings dialog box, click the Libraries tab.

To find out more about how to configure these files and about search order, see [Configuring model libraries on page 4-162](#).

To find out more about the default configuration, see [How are models organized? on page 4-130](#).

To find out more about the library search path, see [Changing the library search path on page 4-167](#).

Unmodeled pins

If you see messages like these in the PSpice Simulation Output window,

Warning: Part *part_name* pin *pin_name* is unmodeled.

Warning: Less than 2 connections at node *node_name*.

or messages like this in the PSpice output file,

Floating/unmodeled pin fixups

then you may have drawn a wire to an unmodeled pin.

The OrCAD libraries include parts that are suitable for both simulation and board layout. The unmodeled pins map into packages but have no electrical significance; PSpice A/D ignores unmodeled pins during simulation.

Check for this

Are there connections to unmodeled pins?

If so, do one of the following:

- Remove wires connected to unmodeled pins.
- If you expect the connection to affect simulation results, find an equivalent part that models the pins in question and draw the connections.

To find out more about searching for parts, see [Finding the part that you want on page 3-102](#).

This applies to analog-only and mixed-signal circuits.

Missing ground

If for *every net* in your circuit you see this message in the PSpice output file,

```
ERROR -- Node node_name is floating.
```

then your circuit may not be tied to ground.

Check for this

Are there ground parts named 0 (zero) connected appropriately in your design?

If not, place and connect one (or more, as needed) in your design. You can use the 0 (zero) ground part in SOURCE.OLB or any other ground part as long as you change its name to 0.

Missing DC path to ground

If for *selected nets* in your circuit you see this message in the PSpice output file,

```
ERROR -- Node node_name is floating.
```

then you may be missing a DC path to ground.

Check for this

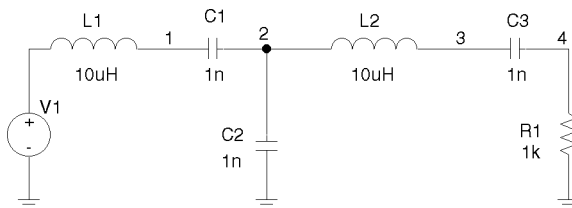
Are there any nets that are isolated from ground by either open circuits or capacitors?

If so, then add a very large (for example, 1 Gohm) resistor either:

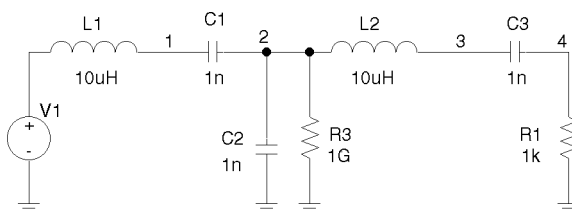
- in parallel with the capacitor or open circuit, or
- from the isolated net to ground.

Example: The circuit shown below connects capacitors (DC open circuits) such that both ends of inductor L2 are isolated from ground.

Note When calculating the bias point solution, PSpice A/D treats capacitors as open circuits and inductors as short circuits.



When simulated, PSpice A/D flags nets 2 and 3 as floating. The following topology solves this problem.



Creating and editing models

4

Chapter overview

This chapter provides information about creating and editing models for parts that you want to simulate.

Topics are grouped into four areas introduced later in this overview. If you want to find out quickly which tools to use to complete a given task and how to start, then:

- 1 Go to the roadmap in [Ways to create and edit models on page 4-134](#).
- 2 Find the task you want to complete.
- 3 Go to the sections referenced for that task for more information about how to proceed.

Background information These sections present model library concepts and an overview of the tools that you can use to create and edit models:

- [What are models? on page 4-129](#)
- [How are models organized? on page 4-130](#)

- [Tools to create and edit models on page 4-133](#)

Task roadmap This section helps you find other sections in this chapter that are relevant to the model editing task that you want to complete:

- [Ways to create and edit models on page 4-134](#)

How to use the tools These sections explain how to use different tools to create and edit models on their own and when editing schematic pages or parts:

- [Using the Model Editor to edit models on page 4-135](#)
- [Editing model text on page 4-152](#)
- [Using the Create Subcircuit command on page 4-157](#)

Other useful information These sections explain how to configure and reuse models after you have created or edited them:

- [Changing the model reference to an existing model definition on page 4-159](#)
- [Reusing instance models on page 4-160](#)
- [Configuring model libraries on page 4-162](#)

What are models?

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

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

- a model parameter set
- a subcircuit netlist

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

Models defined as model parameter sets

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

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

```
.MODEL MLOAD NMOS
+ (LEVEL=1 VT0=0.7 CJ=0.02pF)
```

In addition to the analog models built in to PSpice A/D, the .MODEL syntax applies to the timing and I/O characteristics of digital parts.

Models defined as subcircuit netlists

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

Subcircuit syntax includes:

- *Netlists* to describe the structure and function of the part.
- *Variable input parameters* to fine-tune the model.

For example:

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

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

How are models organized?

The key concepts behind model organization are as follows:

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

Model libraries

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

For example: MOTOR_RF.LIB contains models for Motorola-made RF bipolar transistors.

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

Most model libraries contain models of similar type. For vendor-supplied models, libraries are also partitioned by manufacturer. To find out more about the models contained in a model library, read the comments in the file header.

Model library configuration

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

For PSpice A/D to know where to look for these model definitions, you must configure the libraries. This means:

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

To optimize the search, PSpice A/D uses indexes. To find out more about this and how to add, delete, and rearrange configured libraries, see [Configuring model libraries on page 4-162](#).

Global vs. design models and libraries

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

To find out how to change the design and global configuration of model libraries, see [Changing design and global scope on page 4-165](#).

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

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

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

PSpice A/D searches design libraries before global libraries. To find out more, see [Changing model library search order on page 4-166](#).

Nested model libraries

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

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

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

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

For a list of device models provided by OrCAD, refer to the online *Library List*.

OrCAD-provided models

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

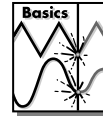
If you click the Libraries tab in the Simulation Settings dialog box immediately after installation, you see the NOM.LIB* entry in the Library Files list. The asterisk means that this model library, and any of the model libraries it references, contain global model definitions.

Tools to create and edit models

There are three tools that you can use to create and edit model definitions. Use the:

- **Model Editor** when you want to:
 - derive models from data sheet curves provided by manufacturers, or
 - modify the behavior of a Model Editor-supported model.
 - edit the PSpice command syntax (text) for .MODEL and .SUBCKT definitions.
- **Create Subcircuit command in the schematic page editor** when you have a hierarchical level in your design that you want to set up as an equivalent part with behavior described as a subcircuit netlist (.SUBCKT syntax).

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



Note *A limited version of the Model Editor is supplied with PSpice A/D Basics.*

For a description of models supported by the Model Editor, see [Model Editor-supported device types on page 4-137](#).

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

Ways to create and edit models

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

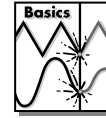
If you want to...	Then do this...	To find out more, see this...
➔ Create or edit the model for an existing part and have it affect all designs that use that part.	Create or load the part first in the part editor, then edit the model using the Model Editor *.	Running the Model Editor from the schematic page editor on page 4-143..
➔ Create a model from scratch and automatically create a part for it to use in any design.	Start the Model Editor * and enable/disable automatic part creation as needed; then create or view the model.	Running the Model Editor alone on page 4-141.
➔ Create a model from scratch without a part and have the model definition available to any design.		
➔ View model characteristics for a part.		
➔ Define tolerances on model parameters for statistical analyses.	Select the part instance on your schematic, then edit the model using the Model Editor.	Starting the Model Editor from the schematic page editor in Capture on page 4-153.
➔ Test behavior variations on a part.	Select the part instance on your schematic page, then edit the model using the Model Editor *.	Running the Model Editor from the schematic page editor on page 4-143
➔ Refine a model before making it available to all designs.		Starting the Model Editor from the schematic page editor in Capture on page 4-153.
➔ Derive subcircuit definitions from a hierarchical design.	Use the Create Subcircuit command in the schematic page editor.	Using the Create Subcircuit command on page 4-157.

* For a list of device types that the Model Editor supports, see [Model Editor-supported device types on page 4-137](#). If the Model Editor does not support the device type for the model definition that you want to create, then you can edit the text using the Model Editor to create a model definition using the PSpice .MODEL and .SUBCKT command syntax. Remember to configure the new model library.

Using the Model Editor to edit models

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

- model parameter sets using PSpice .MODEL syntax, or
- subcircuit netlists using PSpice .SUBCKT syntax, and saves these definitions to model libraries that PSpice A/D can search when looking for simulation models.



Note A limited version of the Model Editor is not supplied with PSpice A/D Basics.

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

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

To refine the subcircuit definition for these constructs, use the Model Text view in Model Editor, described in [Editing model text on page 4-152](#).

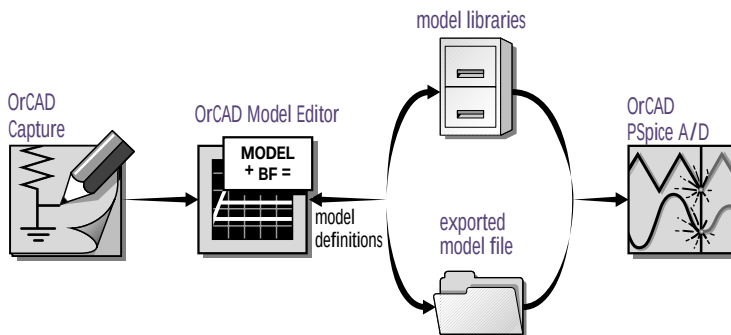


Figure 27 Relationship of the Model Editor to Capture and PSpice A/D.

Note By default, the Model Editor creates or updates model libraries. To create an exported model file, choose the Export command from the Model menu and configure it as an include file. For more information, see [How PSpice A/D uses model libraries](#) and the companion sidebar on page [4-163](#).

Ways to use the Model Editor

You can use the Model Editor five ways:

To find out more, see [Running the Model Editor alone on page 4-141](#).

To find out more, see [Running the Model Editor alone on page 4-141](#).

To find out more, see [Running the Model Editor from the schematic page editor on page 4-143](#).

To find out more, see [Running the Model Editor alone on page 4-141](#).

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

Model Editor-supported device types

Table 19 summarizes the device types supported in the Model Editor.

Table 19 Models supported in the Model Editor

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

* This is the standard PSpice A/D device letter notation. Refer to the online *OrCAD PSpice A/D Reference Manual*.

** The Model Editor only supports .SUBCKT models that were generated by the Model Editor. However, you can edit the text of a .SUBCKT model created manually, or by another tool, using the Model Editor. When you load a .SUBCKT model that the Model Editor did not create, the Model Editor displays the text of the model for editing.

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

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

Testing and verifying models created with the Model Editor

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

However, PSpice uses the *full* equation for the diode model, which includes a term involving the reverse characteristic parameters (such as I_{SR} , N_R). These parameters could have a significant effect at low current.

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

Ways To Characterize Models

Figure 28 shows two ways to characterize models using the Model Editor.

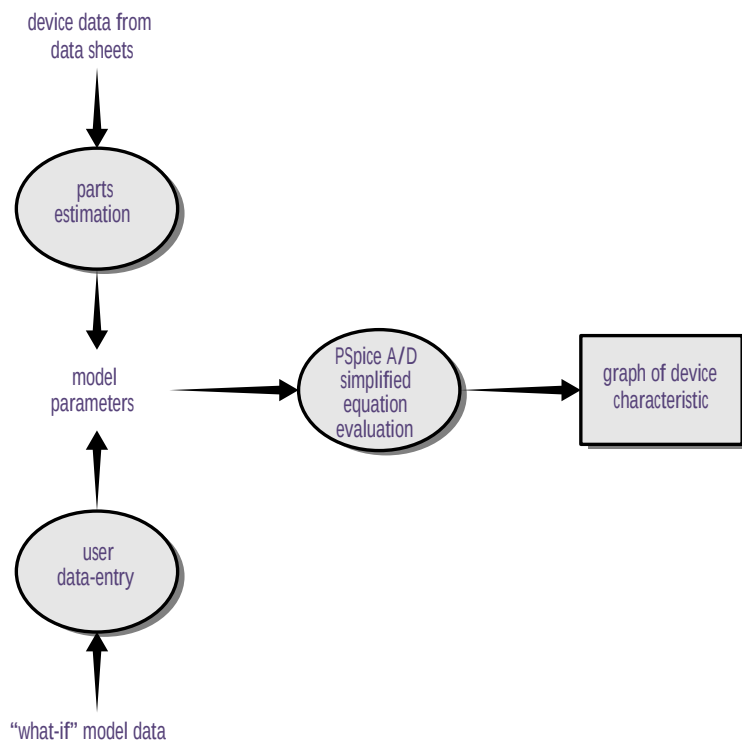


Figure 28 Process and data flow for the Model Editor.

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

Creating models from data sheet information

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

You can repeat this process, and when you are satisfied with the results, save them; the Model Editor creates

model libraries containing appropriate model and subcircuit definitions.

Analyzing the effect of model parameters on device characteristics

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

How to fit models

For a given model, the Model Editor displays a list of the device characteristics and a list of all model parameters and performance curves (see Figure 29).

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

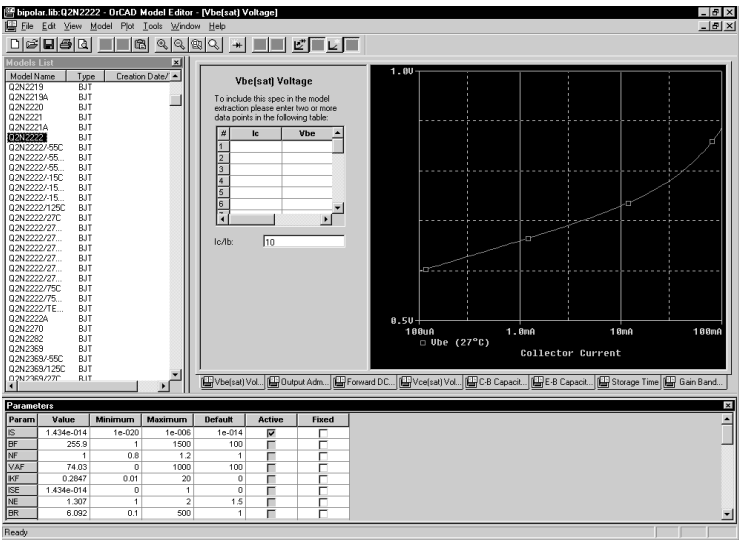


Figure 29 Model Editor workspace with data for a bipolar transistor.

To fit the model

- 1 For each device characteristic that you want to set up:
 - a In the Spec Entry frame, click the tab of the device characteristic.
 - b Enter the device information from the data sheet.



- 2 From the Tools menu, choose Extract Parameters to extract all relevant model parameters for the current specification.

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

- 3 Repeat steps [1-2](#) until the model meets target behaviors.

To view updated performance curves



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

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

Running the Model Editor alone

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

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

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

Note *You can only edit models for device types that the Model Editor supports. See [Model Editor-supported device types on page 4-137](#) for details.*

After you have selected the part that you want to model, you can proceed with entering data sheet information and model fitting as described in [How to fit models on page 4-139](#).

Starting the Model Editor

To start the Model Editor alone

- 1 From the Start menu, point to the OrCAD program folder, then choose Model Editor.
- 2 From the File menu, choose New or Open, and enter an existing or new model library name.
- 3 From the Part menu, choose New, Copy From, or Import to load a model.

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

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

Instead of using the OrCAD default part set for new models, you can have the Model Editor use your own set of standard parts. To find out more, see [Basing new parts on a custom set of parts on page 5-175](#).

Example: If the model library is MYPARTS.LIB, then the Model Editor creates the part library MYPARTS.OLB.

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

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

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

- 1 From the Part menu, select Export.
- 2 Enter the name of the new file.
- 3 If you want PSpice A/D to search this file automatically, configure it in Capture (using the Libraries tab on the Simulation Settings dialog box).

Enabling and disabling automatic part creation

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

To automatically create parts for new models

- 1 From the Tools menu, choose Options.
- 2 If not already checked, select Always Create Part to enable automatic part creation.
- 3 Under Save Part To, enter the name of the part library for the new part. Choose either:
 - Part Library Path Same As Model Library to create or open the *.OLB file that has the same name prefix as the currently open model library (*.LIB).
 - User-Defined Part Library, and then enter a file name in the Part Library Name text box.

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

Saving global models (and parts)

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

- Saves the model definition to the model library that you originally opened.
- If you had the automatic part creation option enabled, saves the part definition to MODEL_LIBRARY_NAME.OLB.

To save the new model (and part)

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

Running the Model Editor from the schematic page editor

If you want to:

- test behavior variations on a part, or
- refine a model before making it available to all designs,

then run the Model Editor from the schematic page editor in Capture.

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

Note You can only edit models for device types that the Model Editor supports. See [Model Editor-supported device types on page 4-137](#) for details.

What is an instance model?

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

When the schematic editor creates the copy, it assigns a unique name that is by default:

original_model_name-Xn

where *n* is <blank 1 | 2 | ... > depending on the number of different instance models derived from the original model for the current design.

Once you have started the Model Editor, you can proceed with entering data sheet information and model fitting as described in [How to fit models on page 4-139](#).

For more information on instance models, see [Reusing instance models on page 4-160](#).

Starting the Model Editor

To start editing an instance model

- 1 In Capture, select one part on your schematic page.
- 2 From the Edit menu, choose PSpice Model.

The schematic page editor searches the model libraries for the instance model.

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

To find out how Capture searches the library, see [Changing model library search order on page 4-166](#).

Saving design models

When you save your edits, the Model Editor saves the model definition to *DESIGN_NAME.LIB*, which is already configured for local use (see [What happens if you don't save the instance model on page 4-145](#)).

To save instance models

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

What happens if you don't save the instance model

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

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

This means that if you:

- quit the Model Editor, or
- return to Capture to simulate the design

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

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

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

To find out how to change model references, see [Changing the model reference to an existing model definition on page 4-159](#).

The Model Editor tutorial

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

- Create the schematic for a simple half-wave rectifier.
- Run the Model Editor from the schematic editor to create an instance model for the diode in your schematic.

Creating the half-wave rectifier design

To draw the design

- 1 From the Project Manager, from the File menu point to New, then choose Project.
- 2 Enter the name of the new project (RECTFR) and click Create.
- 3 From Capture's Place menu, choose Part.
- 4 Place one each of the following parts (reference designator shown in parentheses) as shown in Figure 30:
 - Dbreak (D1 diode)
 - C (C1 capacitor)
 - R (R1 resistor)
 - VSIN (V1 sine wave source)
- 5 Click the Ground button and place the analog ground.
- 6 From the Place menu, choose Wire, and draw the connections between parts as shown in Figure 30.
- 7 From the File menu, choose Save.

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

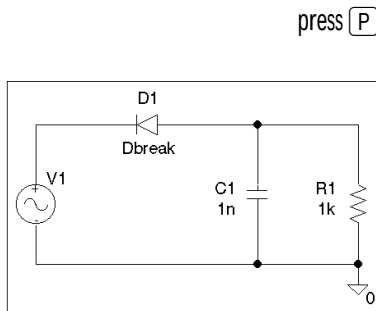


Figure 30 Design for a half-wave rectifier.

Using the Model Editor to edit the D1 diode model

To create a new model and model library

- 1 In the Model Editor, from the Model menu, choose New.
- 2 In the New dialog box, do the following:
 - a In the Model text box, type DbreakX.
 - b From the From Model list, select Diode.
 - c Click OK.
- 3 From the File menu, choose Save As.
- 4 In the File name text box, type rectfr . lib to save the library as RECTFR.LIB.



Entering data sheet information

As shown in Figure 31, the Model Editor initially displays:

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

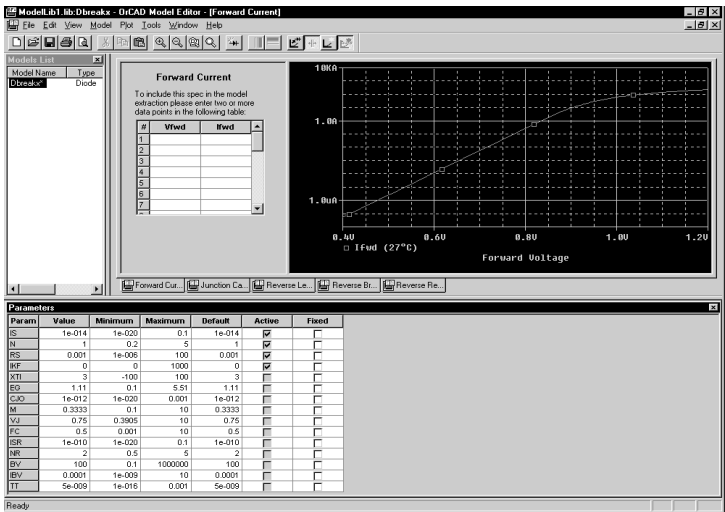


Figure 31 Model characteristics and parameter values for DbreakX.

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

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

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

depending on the device characteristic.

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

Table 1 lists the data sheet information for the Dbreak-X model.

Table 1 Sample diode data sheet values

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

To change the Forward Current characteristic

- 1 In the Spec Entry frame, click the Forward Current tab.
This tab requires curve data.
- 2 In the Vfwd text box, type 1.3.
- 3 Press **Tab** to move to the Ifwd text box, and then type 0.2.

To change the values for Junction Capacitance and Reverse Leakage

- 1 Follow the same steps as for Forward Current, entering the data sheet information listed in [Table 1](#) that corresponds to the current model characteristic.

To change the Reverse Breakdown characteristic

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

This tab requires single-valued data.

- 2 In the Vz text box, type 7.5.
- 3 Press **[Tab]** to move to the Iz text box, and then type 20m.
- 4 Press **[Tab]** to move to the Zz text box, and then type 5.

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

Extracting model parameters

To generate new model parameter values

- 1 From the Tools menu, choose Extract Parameters.
A check mark appears in the Active column of the Parameters frame for each extracted model parameter.



To display the curves for the five diode characteristics

- 1 From the Window menu, choose Tile.
Some of the plots are shown in Figure 32 below.

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

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

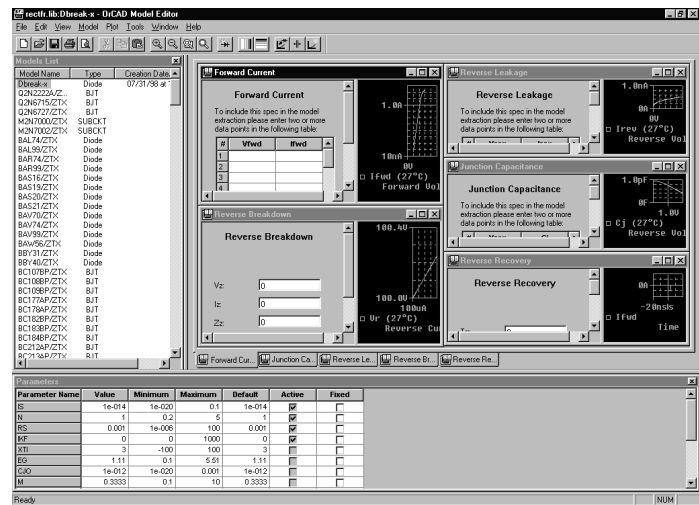


Figure 32 Assorted device characteristic curves for a diode.

Adding curves for more than one temperature

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

To add curves for Forward Current at a different temperature

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

The Forward Current plot should appear as shown in Figure 33 below.

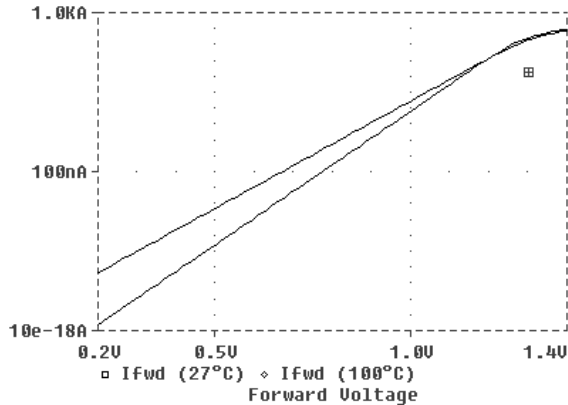


Figure 33 Forward Current device curve at two temperatures.

Completing the model definition

You can refine the model definition by:

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

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

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

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

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

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

Editing model text

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

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

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

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

- change definitions, and
- create new definitions

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

To display the model text

- 1 From the View menu, choose Model Text.

The Model Editor displays the PSpice syntax for model definitions:

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

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

Editing .MODEL definitions

For definitions implemented as model parameter sets using PSpice .MODEL syntax, the Model Editor lists one parameter per line. This makes it easier to add DEV / LOT

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

tolerances to model parameters for Monte Carlo or sensitivity / worst-case analysis.

Editing .SUBCKT definitions

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

Changing the model name

You can change the model name directly in the PSpice .MODEL or .SUBCKT syntax, but double-check that the new name does not conflict with models already contained in the libraries.

Note *If you do create a model with the same name as another model and want PSpice A/D to always use your model, make sure the configured model libraries are ordered so your definition precedes any other definitions.*

To find out more about instance model naming conventions, see [What is an instance model? on page 4-154](#).

To find out more about search order in the model library, see [Changing model library search order on page 4-166](#).

Starting the Model Editor from the schematic page editor in Capture

Start the model editor from the schematic page editor in Capture when you want to:

- define tolerances on model parameters for statistical analyses,
- test behavior variations on a part, or
- refine a model before making it available to all designs.

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

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

For more information on instance models, see [Reusing instance models on page 4-160](#).

After you start the Model Editor, you can proceed to change the text as described in [To display the model text on page 4-152](#).

To find out how Capture searches the library, see [Changing model library search order on page 4-166](#).

What is an instance model?

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

When the schematic page editor creates the copy, it assigns a unique name that is by default:

original_model_name-Xn

where *n* is *<blank 1 | 2 | ... >* depending on the number of different instance models derived from the original model for the current design.

Starting the Model Editor

To start editing an instance model

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

The schematic page editor searches the configured libraries for the instance model:

- If found, the schematic page editor starts the Model Editor, which opens the library containing the instance model and displays the model for editing.
- If not found, the schematic page editor assumes that this is a new instance model and starts the Model Editor, which does the following: makes a copy of the original model definition, names it *original_model_name-Xn*, and displays the new model text for editing.

Saving design models

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

- The Model Editor saves the model definition to *DESIGN_NAME.LIB*.
- If the library is new, the Model Editor configures *DESIGN_NAME.LIB* for local use.
- The schematic page editor assigns the new model name to the Implementation property for each of the selected part instances.

To save instance models

- 1 In the Model Editor, from the File menu, choose Save.
- 2 From the File menu, choose Exit to quit the Model Editor.

Actions that automatically configure the instance model library for global use instead

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

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

Example: editing a Q2N2222 instance model

Suppose you have a design named MY.OPJ that contains several instances of a Q2N2222 bipolar transistor.

Suppose also that you are interested in the effect of base resistance variation on one specific device: Q6. To do this, you need to do the following:

- Define a tolerance (in this example, 5%) on the Rb model parameter.
- Set up and run a Monte Carlo analysis.

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

Starting the Model Editor

To start the Model Editor, you need to:

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

The Model Editor automatically creates a copy of the Q2N2222 base model definition.

- 3 In the Model Editor, from the View menu, choose Model Text.

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

Editing the Q2N2222-X model instance

Text edits appropriate to this example are as follows:

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

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

Saving the edits and updating the schematic

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

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

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

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

If you verify the model library configuration (in the Simulation Settings dialog box, click the Libraries tab), you see entries for NOM.LIB* (for global use, as denoted by the asterisk) and MY.LIB (for design use, no asterisk) in the Library files list.

You can change the model reference for this part back to the original Q2N2222 by following the procedure [To change model references for part instances on your design on page 4-159](#).

Using the Create Subcircuit command

The Create Subcircuit command creates a subcircuit netlist definition for the displayed level of hierarchy and all lower levels in your design.

The schematic page editor does the following things for you:

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

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

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

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

To create a subcircuit definition for a portion of your design

To create a part for the subcircuit

- 1 In the schematic page editor, move to the level of hierarchy for which you want to create a subcircuit (.SUBCKT) definition.
- 2 From the Place menu, choose Hierarchical Port.
- 3 From the File menu, choose Save.
- 4 In the Project Manager, from the Tools menu, choose Create Netlist.
- 5 Select the PSpice tab.
- 6 In the Options frame, select Create SubCircuit Format Netlist.
- 7 Click OK to generate the subcircuit definition and save it to *DESIGN_NAME.SUB*.

To configure the subcircuit file

- 1 In the schematic page editor, from the PSpice menu, choose Edit Simulation Settings to display the Simulation Settings dialog box.
- 2 Click either the Libraries tab or the Include Files tab, then configure *DESIGN_NAME.SUB* as either a model library or an include file (see [Configuring model libraries on page 4-162](#)).
- 3 If necessary, refine the subcircuit definition for the new part or for a part instance on your schematic page using the Model Editor (see [Editing model text on page 4-152](#)).
- 4 From Capture's Edit menu, choose Part to start the part editor.

Refinements can include extending the subcircuit definition using the optional nodes construct, OPTIONAL:, the variable parameters construct, PARAMS:, and the .FUNC and local .PARAM commands.

- 5 Create a new part for the subcircuit definition.

One way to do this is to use the part wizard. See [Chapter 5, Creating parts for models](#) for a complete discussion.

Changing the model reference to an existing model definition

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

You can do this for:

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

To change model references for part instances on your design

- 1 Find the name of the model that you want to use.
- 2 In the schematic page editor, select one or more parts on your schematic page.
- 3 From the Edit menu, choose Properties.
The Parts spreadsheet appears.
- 4 Click the cell under the column Implementation Type.
- 5 From the Implementation list, select PSpice Model.
- 6 In the Implementation column, type the name of the existing model that you want to use if it is not already listed.
- 7 Click Apply to update the changes, then close the spreadsheet.

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

- 1 Find the name of the model that you want to use.
- 2 In the schematic page editor, select the part you want to change.
- 3 From the Edit menu, choose Part to start the part editor with that part loaded for editing.
- 4 From the Options menu, choose Part Properties to display the User Properties dialog box.
- 5 Select Implementation Type.
- 6 From the Implementation list, select PSpice Model.
- 7 In the Implementation text box, type the name of the existing model that you want to use if it is not already listed.
- 8 Click OK to close the Edit Part dialog box.

Reusing instance models

For information on how to create instance models, see:

- [Running the Model Editor from the schematic page editor on page 4-143.](#)
- [Starting the Model Editor from the schematic page editor in Capture on page 4-153.](#)

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

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

Reusing instance models in the same schematic

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

To use the instance model elsewhere in your design

- 1 Do one of the following:

- Change the model reference for other part instances to the name of the new model instance.
- From the Edit menu, use the Copy and Paste commands to place more part instances.

See [Changing the model reference to an existing model definition on page 4-159](#).

Making instance models available to all designs

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

To make your instance model available to any design

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

See [Chapter 5, Creating parts for models](#) for more information.

See [Configuring model libraries on page 4-162](#) for more information.

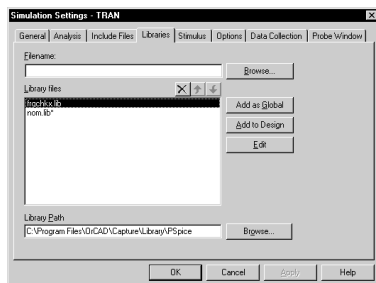
Note *If you use the part wizard to create the part automatically from the model definition, then this step is completed for you.*

Configuring model libraries

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

These are:

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



The Libraries and Include Files tabs

The Libraries and Include Files tabs of the Simulation Settings dialog box are where you can add, change, and remove model libraries and include files from the configuration or resequence the search order.

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

To display the Libraries tab

- 1 In PSpice A/D, from the Simulation menu, choose Edit Simulation Settings.
- 2 Click the Libraries tab.

The Library Files list shows the model libraries that PSpice A/D searches for definitions matching the parts in your design. Files showing an asterisk (*) after their name have global scope; files with names left unmarked have design scope.

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

- Add to Design for design models.

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

The Stimulus tab contains stimulus files. See [Configuring stimulus files on page 11-347](#) for more information.

- Add as Global for global models.

How PSpice A/D uses model libraries

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

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

Search order

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

- design model libraries before global model libraries
- model library sequence as listed in the Libraries tab of the Simulation Settings dialog box
- local directory (where the current design resides) first, then the list of directories specified in the library search path in the order given (see [Changing the library search path on page 4-167](#))

Caution—When you use include files instead

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

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

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

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

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

To reconfigure an include file as a library file:

- 1 From the Simulation menu, choose Edit Simulation Settings, then click the Include Files tab.
- 2 Select the include file that you want to change.
- 3 Click either the Add as Global or the Add to Design button.
- 4 Click Remove to remove the include file entry.

Handling duplicate model names

If your model libraries contain duplicate model names, PSpice A/D always uses the first model it finds. This means you might need to resequence the search order to make sure PSpice A/D uses the model that you want. See [Changing model library search order on page 4-166](#).

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

Adding model libraries to the configuration

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

To add model libraries to the configuration

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

Note *If the model libraries reside in a directory that is not on the library search path, and you use the Browse button in step 3 to select the libraries you want to add, then the schematic editor automatically updates the library search path. Otherwise, you need to add the directory path yourself. See [Changing the library search path on page 4-167](#).*

Changing design and global scope

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

To change the scope of a design model to global

- 1 From the Simulation menu, choose Edit Simulation Settings, then click the Libraries tab.
- 2 Select the model library that you want to change.
- 3 Do one of the following:
 - Click the Add as Global button to add a global entry.
 - Click the Add to Design button to add a design entry.
- 4 Click the Delete toolbar button to remove the local entry.

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

For more information, see [Global vs. design models and libraries on page 4-131](#).

Changing model library search order

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

- reduce the search time
- avoid using the wrong model when there are model names duplicated across libraries; PSpice A / D always uses the first instance

See [Handling duplicate model names on page 4-164](#) for more information.

To change the order of libraries

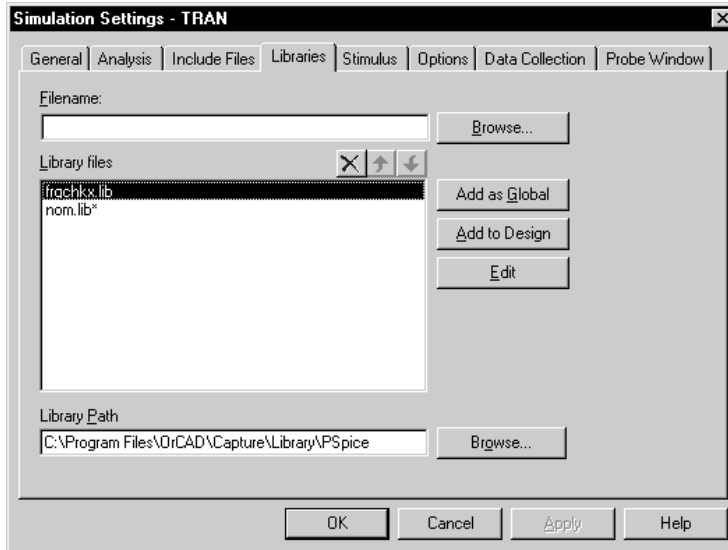
- 1 On the Libraries tab of the Simulation Settings dialog box:
 - a Select the library name you wish to move.
 - b Use either the Up Arrow or Down Arrow toolbar button to move the library name to a different place in the list.
- 2 If you have listed multiple *.LIB commands within a single library (like NOM.LIB), then edit the library using a text editor to change the order.

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

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

Changing the library search path

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



To change the library search path

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

Example: To search first C:\ORCAD\LIB, then C:\MYLIBS, for model libraries, type "C:\ORCAD\LIB"; "C:\MYLIBS" in the Library Path text box.

Creating parts for models

5

Chapter overview

This chapter provides information about creating parts for model definitions, so you can simulate the model from your design using OrCAD Capture.

For general information about creating parts, refer to the *OrCAD Capture User's Guide*.

Topics are grouped into four areas introduced later in this overview. If you want to find out quickly which tools to use to complete a given task and how to start, then:

- 1 Go to the roadmap in [Ways to create parts for models on page 5-171](#).
- 2 Find the task you want to complete.
- 3 Go to the sections referenced for that task for more information about how to proceed.

Background information These sections provide background on the things you need to know and do to prepare for creating parts:

- [What's different about parts used for simulation? on page 5-171](#)
- [Preparing your models for part creation on page 5-172](#)

Task roadmap This section helps you find the sections in this chapter that are relevant to the part creation task that you want to complete:

- [Ways to create parts for models on page 5-171](#)

How to use the tools These sections explain how to use different tools to create parts for model definitions:

- [Using the Model Editor to create parts on page 5-173](#)
- [Using the Model Editor to create parts on page 5-173](#)
- [Basing new parts on a custom set of parts on page 5-175](#)

Other useful information These sections explain how to refine part graphics and properties:

- [Editing part graphics on page 5-177](#)
- [Defining part properties needed for simulation on page 5-181](#)

What's different about parts used for simulation?

A part used for simulation has these special characteristics:

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

For information on adding simulation models to a model library, see [Chapter 4, Creating and editing models](#).

Ways to create parts for models

If you want to...	Then do this...	To find out more, see this...
➔ Create parts for a set of vendor or user-defined models saved in a model library. ➔ Change the graphic standard for an existing model library.	Use the Model Editor to create parts from a model library.	Basing new parts on a custom set of parts on page 5-175
➔ Automatically create one part each time you extract a new model.	Use the Model Editor* and enable automatic creation of parts.	Using the Model Editor to create parts on page 5-173 Using the Model Editor to edit models on page 4-135 Basing new parts on a custom set of parts on page 5-175

* For a list of device types that the Model Editor supports, see [Model Editor-supported device types on page 4-137](#).

Preparing your models for part creation

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

To set up a model library for part creation

- 1 If all of your models are in one file and you wish to keep them that way, rename the file to:
 - Reflect the kinds of models contained in the file.
 - Have the .LIB extension.
- 2 If each model is in its own file, and you want to concatenate them into one file, use the DOS copy command.

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

```
copy *.MOD MYLIB.LIB
```

- 3 Make sure the model names in your new library do not conflict with model names in any other model library.

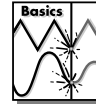
Model libraries typically have a .LIB extension. However, you can use a different file extension as long as the file format conforms to the standard model library file format.

For information on managing model libraries, including the search order PSpice A/D uses, see [Configuring model libraries on page 4-162](#).

Using the Model Editor to create parts

If you want to run the Model Editor and enable automatic creation of parts for any model that you create or change, then run the Model Editor alone. This means any models you create are not tied to the current design or to a part editing session.

Note *If you open an existing model library, the Model Editor creates parts for only the models that you change or add to it.*



Note *The Model Editor is not included in PSpice A/D Basics.*

To find out how to use the Model Editor to create models, see [Using the Model Editor to edit models on page 4-135](#).

To find out which device types the Model Editor supports, see [Model Editor-supported device types on page 4-137](#).

Starting the Model Editor

To start the Model Editor alone

- 1 From the Windows Start menu, point to the OrCAD Release 9 program folder, then choose PSpice Model Editor.
- 2 From the File menu, choose Open or New, and enter an existing or new model library name.
- 3 In the Models List frame, select the name of a model to display it for editing in the Spec Entry frame.

To start the Model Editor from within Capture

- 1 In the schematic page editor, select the part whose model you want to edit.
- 2 From the Edit menu, choose PSpice Model.
The Model Editor starts with the model loaded for editing.

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

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

Setting up automatic part creation

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

Instead of using the OrCAD default part set, you can use your own set of standard parts. To find out more, see [Basing new parts on a custom set of parts on page 5-175](#).

For example, if the model library is named MYPARTS.LIB, then the Model Editor creates the part library named MYPARTS.OLB.

To automatically create parts for new models

- 1 In the Model Editor, from the Tools menu, choose Options.
- 2 In the Part Creation Setup frame, select Create Parts for Models if it is not already enabled.
- 3 In the Save Part To frame, define the name of the part library for the new part. Choose one of the following:
 - Part library path same as model library to create or open the *.OLB file that has the same filename as the open model library (*.LIB).
 - User-defined part library, and then enter a library name in the part Library Name text box.

Basing new parts on a custom set of parts

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

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

To create a custom set of parts for automatic part generation

- 1 Create a part library with the custom parts.

Be sure to name these parts by their device type as shown in Table 2; this is how the Model Editor determines which part to use for a model definition.

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

Table 2 Part names for custom part generation.

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

* Does not apply to the Model Editor.

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

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

To base new parts on custom parts using the Model Editor

- 1 In the Model Editor, from the Options menu, choose Part Creation Setup, and enable automatic part creation as described in [To automatically create parts for new models on page 5-174](#).
- 2 In the Base Parts On frame, enter the name of the existing part library (*.OLB) that contains your custom parts.
- 3 Click OK.

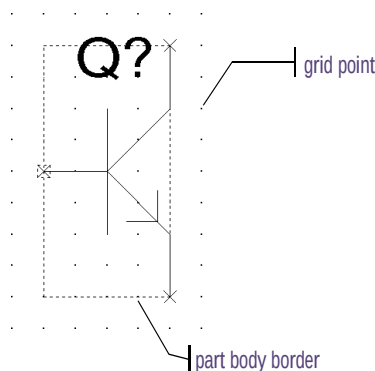
Editing part graphics

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

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

How Capture places parts

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



To edit a part in a library

- 1 From Capture's File menu, point to Open, then choose Library.
- 2 Select the library that has the part you want to edit.
The library opens and displays all its parts.
- 3 Double-click the part you want to edit.
The part appears in the part editor.
- 4 Edit the part.
You can resize it, add or delete graphics, and add or delete pins.

For more information about specific part editing tasks, refer to the *OrCAD Capture User's Guide*.

- 5 After you have finished editing the part, from the File menu, choose Save to save the part to its library.

Defining grid spacing

Grid spacing for graphics

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

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

To edit the part graphics

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

The part body border automatically changes to fit the size of the part graphic.

- 3 After you have finished editing the part, from the File menu, choose Save to save the part to its library.

Note Pin changes that alter the part template can occur if you either:

- change pin names

or

- delete pins

In these cases you must adjust the value of the part's PSPICETEMPLATE property to reflect these changes. To find out how, see

[Pin callout in subcircuit templates on page 5-187.](#)

Grid spacing for pins

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

Pins *must* be placed on the grid at integer multiples of the grid spacing. Because the default grid spacing for the Schematic Page Grid is set at 0.10", OrCAD recommends setting pin spacing in the Part and Symbol Grid at 0.10" intervals from the origin of the part and at least 0.10" from any adjacent pins.

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

Here are two guidelines:

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

For more information about grid spacing and pin placement, refer to the *OrCAD Capture User's Guide*.

Attaching models to parts

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

MODEL

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

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

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

Note *Make sure that the model library containing the definition for the attached model is configured in the list of libraries for your project. See [Configuring model libraries on page 4-162](#) for more information.*

For more information on model editing in general, see [Chapter 4, Creating and editing models](#). For specific information on changing model references, see [Changing the model reference to an existing model definition on page 4-159](#).

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

To attach a model implementation

- 1 In the schematic page editor, double-click a part to display the Parts spreadsheet of the Property Editor.
- 2 From the Implementation list, select PSpice Model.
- 3 In the Implementation column, type the name of the model to attach to the part.
- 4 Click Apply to update the design, then close the Parts spreadsheet.

Defining part properties needed for simulation

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

- PSpice PSPICETEMPLATE for simulation
- PART and REFDES for identification

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

For example, if you create a part that has electrical behavior described by the subcircuit definition that starts with:

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

then the appropriate part properties are:

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

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

Table 3

To find out more about this property...	See this...
PSPICETEMPLATE	page 5-182
IO_LEVEL	page 5-189
MNTYMXDLY	page 5-190
PSPICEDEFAULTNET	page 5-191

Here are the things to check when editing part properties:

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

To edit a property needed for simulation:

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

Caution—Creating parts not intended for simulation

Some part libraries contain parts designed only for board layout; PSpice A/D cannot simulate these parts. This means they do not have PSPICETEMPLATE properties or that the PSPICETEMPLATE property value is blank.

PSPICETEMPLATE

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

Any part that you want to simulate must have a defined PSPICETEMPLATE property. These rules apply:

- The pin names specified in the PSPICETEMPLATE property must match the pin names on the part.
- The number and order of the pins listed in the PSPICETEMPLATE property must match those for the associated .MODEL or .SUBCKT definition referenced for simulation.
- The first character in a PSPICETEMPLATE must be a PSpice A / D device letter appropriate for the part (such as Q for a bipolar transistor).

PSPICETEMPLATE syntax

The PSPICETEMPLATE contains:

- *regular characters* that the schematic page editor interprets verbatim
- *property names* and *control characters* that the schematic page editor translates

Regular characters in templates

Regular characters include the following:

- alphanumerics
- any keyboard part *except* the special syntactical parts used with properties (@ & ? ~ #).
- white space

An *identifier* is a collection of regular characters of the form:

alphabetic character [any other regular character].*

Property names in templates

Property names are preceded by a special character as follows:

[@ | ? | ~ | # | &]<identifier>

The schematic page editor processes the property according to the special character as shown in the following table.

Table 4

This syntax...*	Is replaced with this...
@<id>	Value of <id>. Error if no <id> attribute or if no value assigned.
&<id>	Value of <id> if <id> is defined.
?<id>s...s	Text between s...s separators if <id> is defined.
?<id>s...ss...s	Text between the first s...s separators if <id> is defined, else the second s...s clause.
~<id>s...s	Text between s...s separators if <id> is undefined.
~<id> s...ss...s	Text between the first s...s separators if <id> is undefined, else the second s...s clause.
#<id>s...s	Text between s...s separators if <id> is defined, but delete rest of template if <id> is undefined.

* s is a separator character

Separator characters include commas (,), periods (.), semi-colons (;), forward slashes (/), and vertical bars (|). You must always use the same character to specify an opening-closing pair of separators.

Note *You can use different separator characters to nest conditional property clauses.*

Example: The template fragment
 ?G|G=@G||G=1000| uses the vertical bar as the separator between the if-then-else parts of this conditional clause. If G has a value, then this fragment translates to G=<G property value>. Otherwise, this fragment translates to G=1000.

Caution—Recommended scheme for netlist templates

Templates for devices in the part library start with a PSpice A/D device letter, followed by the hierarchical path, and then the reference designator (REFDES) property.

OrCAD recommends that you adopt this scheme when defining your own netlist templates.

Example: R^@REFDES ... for a resistor

The ^ character in templates

The schematic page editor replaces the ^ character with the complete hierarchical path to the device being netlisted.

The \n character sequence in templates

The part editor replaces the character sequence \n with a new line. Using \n, you can specify a multi-line netlist entry from a one-line template.

The % character and pin names in templates

Pin names are denoted as follows:

`%<pin name>`

where *pin name* is one or more regular characters.

The schematic page editor replaces the `%<pin name>` clause in the template with the name of the net connected to that pin.

The end of the pin name is marked with a separator (see [Property names in templates on page 5-183](#)). To avoid name conflicts in PSpice, the schematic page editor translates the following characters contained in pin names.

Table 5

This pin name character...	Is replaced with this...
<	l (L)
>	g
=	e
\XXX\	XXXbar

Note *To include a literal % character in the netlist, type %% in the template.*

PSPICETEMPLATE examples

Simple resistor (R) template

The R part has:

- two pins: 1 and 2
- two required properties: REFDES and VALUE

Template

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

Sample translation

```
R_R23 abc def 1k
```

where REFDES equals R23, VALUE equals 1k, and R is connected to nets abc and def.

Voltage source with optional AC and DC specifications (VAC) template

The VAC part has:

- two properties: AC and DC
- two pins: + and -

Template

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

Sample translation

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

where REFDES equals V6, VSRC is connected to nodes vp and vm, DC is set to 5v, and AC is undefined.

Sample translation

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

where, in addition to the settings for the previous translation, AC is set to 1v.

Parameterized subcircuit call (X) template

Suppose you have a subcircuit Z that has:

- two pins: a and b
- a subcircuit parameter: G, where G defaults to 1000 when no value is supplied

To allow the parameter to be changed on the schematic page, treat G as an property in the template.

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

Template

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

Equivalent template (using the if...else form)

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

Sample translation

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

where REFDES equals U33, G is set to 1024, and the subcircuit connects to nets 101 and 102.

Sample translation

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

where the settings of the previous translation apply except that G is undefined.

Digital stimulus parts with variable width pins template

For a digital stimulus device template (such as that for a DIGSTIM part), a pin name can be preceded by a * character. This signifies that the pin can be connected to a bus and the width of the pin is set to be equal to the width of the bus.

Template

```
U^@REFDES STIM(%#PIN, 0) %*PIN
  \n+ STIMULUS=@STIMULUS
```

where #PIN refers to a variable width pin.

Sample translation

```
U_U1 STIM(4,0) 5PIN1 %PIN2 %PIN3 %PIN4
+ STIMULUS=mystim
```

where the stimulus is connected to a four-input bus, a[0-3].

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

Pin callout in subcircuit templates

The number and sequence of pins named in a template for a subcircuit must agree with the definition of the subcircuit itself—that is, the node names listed in the .SUBCKT statement, which heads the definition of a subcircuit. These are the pinouts of the subcircuit.

Example: Consider the following first line of a (hypothetical) subcircuit definition:

```
.SUBCKT SAMPLE 10 3 27 2
```

The four numbers following the name SAMPLE—10, 3, 27, and 2—are the node names for this subcircuit's pinouts.

Now suppose that the part definition shows four pins:

```
IN+          OUT+          IN-          OU
T-
```

The number of pins on the part equals the number of nodes in the subcircuit definition.

To find out how to define subcircuits, refer to the .SUBCKT command in the online *OrCAD PSpice A/D Reference Manual*.

If the correspondence between pin names and nodes is as follows:

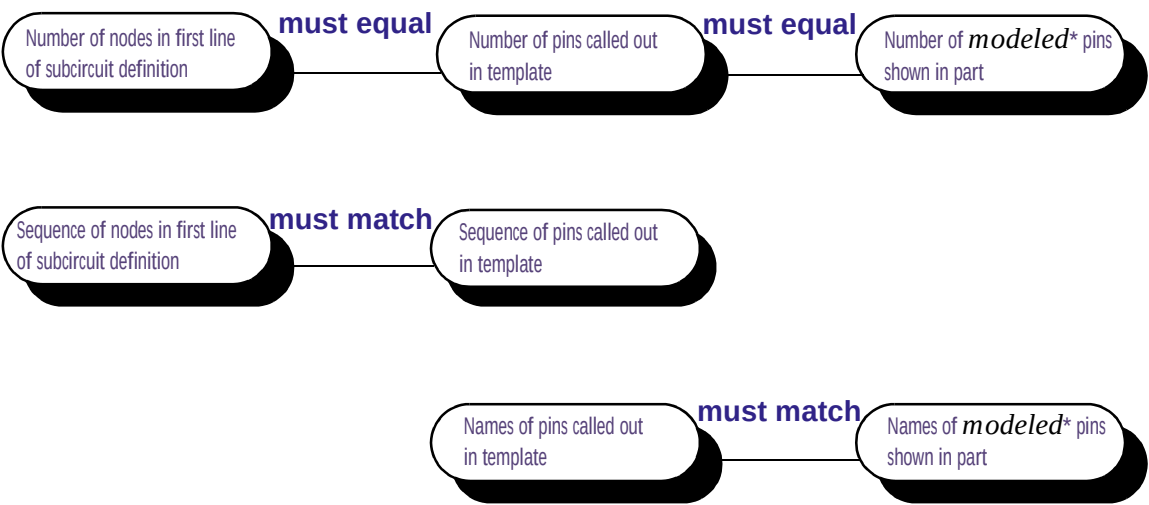
Table 6

This node name...	Corresponds to this pin name...
10	IN+
3	IN-
27	OUT+
2	OUT-

then the template looks like this:

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

The *rules of agreement* are outlined in Figure 34.



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

Figure 34 *Rules for pin callout in subcircuit templates.*

IO_LEVEL

The IO_LEVEL property defines what level of interface subcircuit model PSpice A/D must use for a digital part that is connected to an analog part.

To use the IO_LEVEL property with a digital part

- 1 Add the IO_LEVEL property to the part and assign a value shown in the table below.

Table 7

Assign this value...	To use this interface subcircuit (level)...
0	circuit-wide default
1	AtoD1 and DtoA1
2	AtoD2 and DtoA2
3	AtoD3 and DtoA3
4	AtoD4 and DtoA4

- 2 Use this property in the PSPICETEMPLATE property definition (IO_LEVEL is also a subcircuit parameter used in calls for digital subcircuits).

Example:

```
PSPICETEMPLATE=X^@REFDES %A %B %C %D %PWR
%GND
@MODEL PARAMS:\n+
IO_LEVEL=@IO_LEVEL
MNTYMXDLY=@MNTYMXDLY
```

All digital parts provided in the OrCAD libraries have an IO_LEVEL property.

To find out more about interface subcircuits, see [Interface subcircuit selection by PSpice A/D on page 15-445](#).

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

All digital parts provided in the OrCAD libraries have a MNTYMXDLY property. To find out more about propagation delays, see [Timing characteristics on page 7-251](#) and [Selecting propagation delays on page 14-428](#).

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

MNTYMXDLY

The MNTYMXDLY property defines the digital propagation delay level that PSpice A/D must use for a digital part.

To use the MNTYMXDLY property with a digital part

- 1 Add the MNTYMXDLY property to the part and assign a value shown in the table below.

Table 8

Assign this value...	To use this propagation delay...
0	circuit-wide default
1	minimum
2	typical
3	maximum
4	worst-case (min / max)

- 2 Use this property in the PSPICETEMPLATE property definition (MNTYMXDLY is also a subcircuit parameter used in calls for digital subcircuits).

Example:

```
PSPICETEMPLATE=X^@REFDES %A %B %C %D %PWR
%GND
@MODEL PARAMS:\n+
IO_LEVEL=@IO_LEVEL
MNTYMXDLY=@MNTYMXDLY
```

PSPICEDEFAULTNET

The PSPICEDEFAULTNET pin property defines the net name to which a hidden (invisible) pin is connected.

Hidden pins are typically used for power and ground on digital parts.

To use the PSPICEDEFAULTNET property with a digital part

- 1 For each PSPICEDEFAULTNET property, assign the name of the digital net to which the pin is connected.

Example: If power (PWR) and ground (GND) pins of a digital part connect to the digital nets \$G_DPWR and \$G_DGND, respectively, then the PSPICEDEFAULTNET properties for these pins are:

```
PSPICEDEFAULTNET=$G_DPWR
```

```
PSPICEDEFAULTNET=$G_DGND
```

- 2 Use the appropriate hidden pin name in the PSPICETEMPLATE property definition.

Example: If the name of the hidden power pin is PWR and the name of the hidden ground pin is GND, then the template might look like this:

```
PSPICETEMPLATE=X^@REFDES %A %B %C %D %PWR
%GND
@MODEL PARAMS:\n+
IO_LEVEL=@IO_LEVEL
MNTYMXDLY=@MNTYMXDLY
```

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

Analog behavioral modeling

6

Chapter overview

This chapter describes how to use the Analog Behavioral Modeling (ABM) feature of PSpice A / D. This chapter includes the following sections:

- [Overview of analog behavioral modeling on page 6-194](#)
- [The ABM.OLB part library file on page 6-195](#)
- [Placing and specifying ABM parts on page 6-196](#)
- [ABM part templates on page 6-198](#)
- [Control system parts on page 6-199](#)
- [PSpice A/D-equivalent parts on page 6-220](#)
- [Cautions and recommendations for simulation and analysis on page 6-232](#)
- [Basic controlled sources on page 6-239](#)

Overview of analog behavioral modeling

You can use the Analog Behavioral Modeling (ABM) feature of PSpice A / D to make flexible descriptions of electronic components in terms of a transfer function or lookup table. In other words, a mathematical relationship is used to model a circuit segment, so you do not need to design the segment component by component.

The part library contains several ABM parts that are classified as either control system parts or as PSpice A / D-equivalent parts. See [Basic controlled sources on page 6-239](#) for an introduction to these parts, how to use them, and the difference between parts with general-purpose application and parts with special-purpose application.

Control system parts are defined with the reference voltage preset to ground so that each controlling input and output are represented by a single pin in the part. These are described in [Control system parts on page 6-199](#).

PSpice A / D-equivalent parts reflect the structure of the PSpice A / D E and G device types, which respond to a differential input and have double-ended output. These are described in [PSpice A/D-equivalent parts on page 6-220](#).

You can also use the Device Equations option (described in the online *OrCAD PSpice A/D Reference Manual*) for modeling of this type, but OrCAD recommends using the ABM feature wherever possible. With Device Equations, the PSpice A / D source code is actually modified. While this is more flexible and produces faster results, it is also much more difficult to use and to troubleshoot. Also, any changes you make using Device Equations must be made to all new PSpice A / D updates you install.

Device models made with ABM can be used for most cases, are much easier to create, and are compatible with PSpice A / D updates.

The ABM.OLB part library file

The part library ABM.OLB contains the ABM components. This library contains two sections.

The first section has parts that you can quickly connect to form control system types of circuits. These components have names like SUM, GAIN, LAPLACE, and HIPASS.

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

Implement ABM components by using PSpice A/D primitives; there is no corresponding `abm.lib` model library. A few components generate multi-line netlist entries, but most are implemented as single PSpice A/D E or G device declarations. See [ABM part templates on page 6-198](#) for a description of PSPICETEMPLATE properties and their role in generating netlist declarations. See [Implementation of PSpice A/D-equivalent parts on page 6-221](#) for more information about PSpice A/D E and G syntax.

Placing and specifying ABM parts

Place and connect ABM parts the same way you place other parts. After you place an ABM part, you can edit the instance properties to customize the operational behavior of the part. This is equivalent to defining an ABM expression describing how inputs are transformed into outputs. The following sections describe the rules for specifying ABM expressions.

Net names and device names in ABM expressions

In ABM expressions, refer to signals by name. This is also considerably more convenient than having to connect a wire from a pin on an ABM component to a point carrying the voltage of interest.

The name of an interface port does not extend to any connected nets. To refer to a signal originating at an interface port, connect the port to an offpage connector of the desired name.

If you used an expression such as `V(2)`, then the referenced net (2 in this case) is interpreted as the name of a local or global net. A local net is a labeled wire or bus segment in a hierarchical schematic, or a labeled offpage connector. A global net is a labeled wire or bus segment at the top level, or a global connector.

OrCAD Capture recognizes these constructs in ABM expressions:

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

When one of these is recognized, Capture searches for `<net name>` or `<vdevice>` in the net name space or the device name space, respectively. Names are searched for first at the hierarchical level of the part being netlisted. If not found there, then the set of global names is searched. If the fragment is not found, then a warning is issued but Capture still outputs the resulting netlist. When a match is

found, the original fragment is replaced by the fully qualified name of the net or device.

For example, suppose we have a hierarchical part U1. Inside the schematic representing U1 we have an ABM expression including the term V(Reference). If “Reference” is the name of a local net, then the fragment written to the netlist will be translated to V(U1_Reference). If “Reference” is the name of a global net, the corresponding netlist fragment will be V(Reference).

Names of voltage sources are treated similarly. For example, an expression including the term I(Vsense) will be output as I(V_U1_Vsense) if the voltage source exists locally, and as I(V_Vsense) if the voltage source exists at the top level.

Forcing the use of a global definition

If a net name exists both at the local hierarchical level and at the top level, the search mechanism used by Capture will find the local definition. You can override this, and force Capture to use the global definition, by prefixing the name with a single quote (') character.

For example, suppose there is a net called Reference both inside hierarchical part U1 and at the top level. Then, the ABM fragment V(Reference) will result in V(U1_Reference) in the netlist, while the fragment V('Reference) will produce V(Reference).

ABM part templates

For most ABM parts, a single PSpice A/D “E” or “G” device declaration is output to the netlist per part instance. The PSPICETEMPLATE property in these cases is straightforward. For example the LOG part defines an expression variant of the E device with its output being the natural logarithm of the voltage between the input pin and ground:

```
E^@REFDES %out 0 VALUE { LOG(V(%in)) }
```

The fragment E^@REFDES is standard. The “E” specifies a PSpice A/D controlled voltage source (E device); %in and %out are the input and output pins, respectively; VALUE is the keyword specifying the type of ABM device; and the expression inside the curly braces defines the logarithm of the input voltage.

Several ABM parts produce more than one primitive PSpice A/D device per part instance. In this case, the PSPICETEMPLATE property may be quite complicated. An example is the DIFFER (differentiator) part. This is implemented as a capacitor in series with a current sensor together with an E device which outputs a voltage proportional to the current through the capacitor.

The template has several unusual features: it gives rise to three primitives in the PSpice A/D netlist, and it creates a local node for the connection of the capacitor and its current-sensing V device.

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

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

The fragments C^@REFDES, V^@REFDES, and E^@REFDES create a uniquely named capacitor, current sensing V device, and E device, respectively. The fragment \$\$U^@REFDES creates a name suitable for use as a local node. The E device generates an output proportional to the current through the local V device.

Control system parts

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

Table 9 lists the control system parts, grouped by function. Also listed are characteristic properties that may be set. In the sections that follow, each part and its properties are described in more detail.

Table 9 *Control system parts*

Category	Part	Description	Properties
Basic components	CONST	constant	VALUE
	SUM	adder	
	MULT	multiplier	
	GAIN	gain block	GAIN
	DIFF	subtractor	
Limiters	LIMIT	hard limiter	LO, HI
	GLIMIT	limiter with gain	LO, HI, GAIN
	SOFTLIM	soft (tanh) limiter	LO, HI, GAIN
Chebyshev filters	LOPASS	lowpass filter	FP, FS, RIPPLE, STOP
	HIPASS	highpass filter	FP, FS, RIPPLE, STOP
	BANDPASS	bandpass filter	F0, F1, F2, F3, RIPPLE, STOP
	BANDREJ	band reject (notch) filter	F0, F1, F2, F3, RIPPLE, STOP
Integrator and differentiator	INTEG	integrator	GAIN, IC
	DIFFER	differentiator	GAIN
Table look-ups	TABLE	lookup table	ROW1...ROW5
	FTABLE	frequency lookup table	ROW1...ROW5

Table 9 Control system parts (continued)

Category	Part	Description	Properties
Laplace transform	LAPLACE	Laplace expression	NUM, DENOM
Math functions (where 'x' is the input)	ABS	$ x $	
	SQRT	$x^{1/2}$	
	PWR	$ x ^{\text{EXP}}$	EXP
	PWRS	x^{EXP}	EXP
	LOG	$\ln(x)$	
	LOG10	$\log(x)$	
	EXP	e^x	
	SIN	$\sin(x)$	
	COS	$\cos(x)$	
	TAN	$\tan(x)$	
	ATAN	$\tan^{-1}(x)$	
	ARCTAN	$\tan^{-1}(x)$	
Expression functions	ABM	no inputs, V out	EXP1...EXP4
	ABM1	1 input, V out	EXP1...EXP4
	ABM2	2 inputs, V out	EXP1...EXP4
	ABM3	3 inputs, V out	EXP1...EXP4
	ABM/I	no input, I out	EXP1...EXP4
	ABM1/I	1 input, I out	EXP1...EXP4
	ABM2/I	2 inputs, I out	EXP1...EXP4
	ABM3/I	3 inputs, I out	EXP1...EXP4

Basic components

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

CONST

VALUE constant value

The CONST part outputs the voltage specified by the VALUE property. This part provides no inputs and one output.

SUM

The SUM part evaluates the voltages of the two input sources, adds the two inputs together, then outputs the sum. This part provides two inputs and one output.

MULT

The MULT part evaluates the voltages of the two input sources, multiplies the two together, then outputs the product. This part provides two inputs and one output.

GAIN

GAIN constant gain value

The GAIN part multiplies the input by the constant specified by the GAIN property, then outputs the result. This part provides one input and one output.

DIFF

The DIFF part evaluates the voltage difference between two inputs, then outputs the result. This part provides two inputs and one output.

Limiters

The Limiters can be used to restrict an output to values between a set of specified ranges. These parts are described below.

LIMIT

HI	upper limit value
LO	lower limit value

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

GLIMIT

HI	upper limit value
LO	lower limit value
GAIN	constant gain value

The GLIMIT part functions as a one-line opamp. The gain is applied to the input voltage, then the output is constrained to the limits set by the LO and HI properties. This part takes one input and provides one output.

SOFTLIMIT

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

The SOFTLIMIT part provides a limiting function much like the LIMIT device, except that it uses a continuous curve limiting function, rather than a discontinuous limiting function. This part takes one input and provides one output.

Chebyshev filters

The Chebyshev filters allow filtering of the signal based on a set of frequency characteristics. The output of a Chebyshev filter depends upon the analysis being performed.

Note *PSpice A/D computes the impulse response of each Chebyshev filter used in a transient analysis during circuit read-in. This may require considerable computing time. A message is displayed on your screen indicating that the computation is in progress.*

For DC and bias point, the output is simply the DC response of the filter. For AC analysis, the output for each frequency is the filter response at that frequency. For transient analysis, the output is then the convolution of the past values of the input with the impulse response of the filter. These rules follow the standard method of using Fourier transforms.

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

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

LOPASS

FS	stop band frequency
FP	pass band frequency
RIPPLE	pass band ripple in dB
STOP	stop band attenuation in dB

The LOPASS part is characterized by two cutoff frequencies that delineate the boundaries of the filter pass band and stop band. The attenuation values, RIPPLE and STOP, define the maximum allowable attenuation in the pass band, and the minimum required attenuation in the stop band, respectively. The LOPASS part provides one input and one output.

Figure 35 shows an example of a LOPASS filter device. The filter provides a pass band cutoff of 800 Hz and a stop

OrCAD Capture recommends looking at one or more of the references cited in [Frequency-domain device models on page 6-227](#), as well as some of the following references on analog filter design:

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

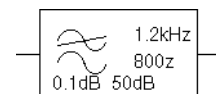


Figure 35 LOPASS filter example.

band cutoff of 1.2 kHz. The pass band ripple is 0.1 dB and the minimum stop band attenuation is 50 dB. Assuming that the input to the filter is the voltage at net 10 and output is a voltage between nets 5 and 0, this will produce a PSpice A/D netlist declaration like this:

```
ELOWPASS 5 0 CHEBYSHEV {V(10)} = LP 800 1.2K .1dB 50dB
```

HIPASS

FS	stop band frequency
FP	pass band frequency
RIPPLE	pass band ripple in dB
STOP	stop band attenuation in dB

The HIPASS part is characterized by two cutoff frequencies that delineate the boundaries of the filter pass band and stop band. The attenuation values, RIPPLE and STOP, define the maximum allowable attenuation in the pass band, and the minimum required attenuation in the stop band, respectively. The HIPASS part provides one input and one output.

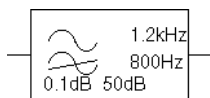


Figure 36 HIPASS filter part example.

Figure 36 shows an example of a HIPASS filter device. This is a high pass filter with the pass band above 1.2 kHz and the stop band below 800 Hz. Again, the pass band ripple is 0.1 dB and the minimum stop band attenuation is 50 dB. This will produce a PSpice A/D netlist declaration like this:

```
EHIGHPASS 5 0 CHEBYSHEV {V(10)} = HP 1.2K 800 .1dB 50dB
```

BANDPASS

RIPPLE	pass band ripple in dB
STOP	stop band attenuation in dB
F0, F1, F2, F3	cutoff frequencies

The BANDPASS part is characterized by four cutoff frequencies. The attenuation values, RIPPLE and STOP, define the maximum allowable attenuation in the pass band, and the minimum required attenuation in the stop

band, respectively. The BANDPASS part provides one input and one output.

Figure 37 shows an example of a BANDPASS filter device. This is a band pass filter with the pass band between 1.2 kHz and 2 kHz, and stop bands below 800 Hz and above 3 kHz. The pass band ripple is 0.1 dB and the minimum stop band attenuation is 50 dB. This will produce a PSpice A/D netlist declaration like this:

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

BANDREJ

RIPPLE is the pass band ripple in dB
 STOP is the stop band attenuation in dB
 F0, F1, are the cutoff frequencies
 F2, F3

The BANDREJ part is characterized by four cutoff frequencies. The attenuation values, RIPPLE and STOP, define the maximum allowable attenuation in the pass band, and the minimum required attenuation in the stop band, respectively. The BANDREJ part provides one input and one output.

Figure 38 shows an example of a BANDREJ filter device. This is a band reject (or “notch”) filter with the stop band between 1.2 kHz and 2 kHz, and pass bands below 800 Hz and above 3 kHz. The pass band ripple is 0.1 dB and the minimum stop band attenuation is 50 dB. This will produce a PSpice A/D netlist declaration like this:

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

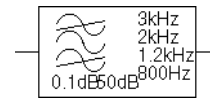


Figure 37 BANDPASS filter part example.

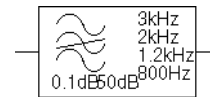


Figure 38 BANDREJ filter part example.

Integrator and differentiator

The integrator and differentiator parts are described below.

INTEG

IC	initial condition of the integrator output
GAIN	gain value

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

DIFFER

GAIN	gain value
------	------------

The DIFFER part implements a simple differentiator. A voltage source / capacitor implementation is used. The DIFFER part provides one input and one output.

Table look-up parts

TABLE and FTABLE parts provide a lookup table that is used to correlate an input and an output based on a set of data points. These parts are described below and on the following pages.

TABLE

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

ROW <i>n</i>	is an (input, output) pair; by default, up to five triplets are allowed where $n=1, 2, 3, 4,$ or 5
--------------	--

The TABLE part allows the response to be defined by a table of one to five values. Each row contains an input and a corresponding output value. Linear interpolation is performed between entries.

For values outside the table's range, the device's output is a constant with a value equal to the entry with the smallest (or largest) input. This characteristic can be used to

impose an upper and lower limit on the output. The TABLE part provides one input and one output.

FTABLE

ROW n	either an (input frequency, magnitude, phase) triplet, or an (input frequency, real part, imaginary part) triplet describing a complex value; by default, up to five triplets are allowed where $n=1, 2, 3, 4$, or 5
DELAY	group delay increment; defaults to 0 if left blank
R_I	table type; if left blank, the frequency table is interpreted in the (input frequency, magnitude, phase) format; if defined with any value (such as YES), the table is interpreted in the (input frequency, real part, imaginary part) format
MAGUNITS	units for magnitude where the value can be DB (decibels) or MAG (raw magnitude); defaults to DB if left blank
PHASEUNITS	units for phase where the value can be DEG (degrees) or RAD (radians); defaults to DEG if left blank

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

The FTABLE part is described by a table of frequency responses in either the magnitude / phase domain (R_I=) or complex number domain (R_I=YES). The entire table is read in and converted to magnitude in dB and phase in degrees.

Interpolation is performed between entries. Magnitude is interpolated logarithmically; phase is interpolated linearly. For frequencies outside the table's range, 0 (zero) magnitude is used. This characteristic can be used to impose an upper and lower limit on the output.

The DELAY property increases the group delay of the frequency table by the specified amount. The delay term is particularly useful when a frequency table device generates a non-causality warning message during a transient analysis. The warning message issues a delay

value that can be assigned to the part’s DELAY property for subsequent runs, without otherwise altering the table.

The output of the part depends on the analysis being done. For DC and bias point, the output is the zero frequency magnitude times the input voltage. For AC analysis, the input voltage is linearized around the bias point (similar to EVALUE and GVALUE parts, [Modeling mathematical or instantaneous relationships on page 6-222](#)). The output for each frequency is then the input times the gain, times the value of the table at that frequency.

For transient analysis, the voltage is evaluated at each time point. The output is then the convolution of the past values with the impulse response of the frequency response. These rules follow the standard method of using Fourier transforms. We recommend looking at one or more of the references cited in [Frequency-domain device models on page 6-227](#) for more information.

Note *The table’s frequencies must be in order from lowest to highest. The TABLE part provides one input and one output.*

Example

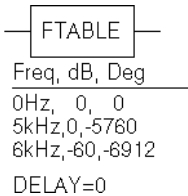


Figure 39 FTABLE part example.

A device, ELOFILT, is used as a frequency filter. The input to the frequency response is the voltage at net 10. The output is a voltage across nets 5 and 0. The table describes a low pass filter with a response of 1 (0 dB) for frequencies below 5 kilohertz and a response of 0.001 (-60 dB) for frequencies above 6 kilohertz. The phase lags linearly with frequency. This is the same as a constant time delay. The delay is necessary so that the impulse response is causal. That is, so that the impulse response does not have any significant components before time zero. The FTABLE part in Figure 39 could be used.

This part is characterized by the following properties:

```

ROW1 = 0Hz      0      0
ROW2 = 5kHz     0     -5760
ROW3 = 6kHz     -60    -6912
DELAY =
R_I =
MAGUNITS =
PHASEUNITS =

```

Since `R_I`, `MAGUNITS`, and `PHASEUNITS` are undefined, each table entry is interpreted as containing frequency, magnitude value in dB, and phase values in degrees. Delay defaults to 0.

This produces a PSpice A/D netlist declaration like this:

```

ELOFILT 5 0 FREQ {V(10)} = (0,0,0) (5kHz,0,-5760)
+ (6kHz,-60,-6912)

```

Since constant group delay is calculated from the values for a given table entry as:

$$\text{group delay} = \text{phase} / 360 / \text{frequency}$$

An equivalent `FTABLE` instance could be defined using the `DELAY` property. For this example, the group delay is 3.2 msec ($6912 / 360 / 6k = 5760 / 360 / 6k = 3.2m$).

Equivalent property assignments are:

```

ROW1 = 0Hz      0      0
ROW2 = 5kHz     0      0
ROW3 = 6kHz     -60    0
DELAY = 3.2ms
R_I =
MAGUNITS =
PHASEUNITS =

```

This produces a PSpice A/D netlist declaration like this:

```

ELOFILT 5 0 FREQ {V(10)} = (0,0,0) (5kHz,0,0) (6kHz,-60,0)
+ DELAY=3.2ms

```

Laplace transform part

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

LAPLACE

NUM	numerator of the Laplace expression
DENOM	denominator of the Laplace expression

The LAPLACE part uses a Laplace transform description. The input to the transform is a voltage. The numerator and denominator of the Laplace transform function are specified as properties for the part.

Note *Voltages, currents, and TIME may not appear in a Laplace transform specification.*

The output of the part depends on the type of analysis being done. For DC and bias point, the output is the zero frequency gain times the value of the input. The zero frequency gain is the value of the Laplace transform with $s=0$. For AC analysis, the output is then the input times the gain times the value of the Laplace transform. The value of the Laplace transform at a frequency is calculated by substituting $j\omega$ for s , where ω is 2π -frequency. For transient analysis, the output is the convolution of the input waveform with the impulse response of the transform. These rules follow the standard method of using Laplace transforms.

Example one

The input to the Laplace transform is the voltage at net 10. The output is a voltage and is applied between nets 5 and 0. For DC, the output is simply equal to the input, since the gain at $s = 0$ is 1. The transform, $1/(1+.001 \cdot s)$, describes a simple, lossy integrator with a time constant of 1 millisecond. This can be implemented with an RC pair that has a time constant of 1 millisecond.

For AC analysis, the gain is found by substituting $j\omega$ for s . This gives a flat response out to a corner frequency of $1000/(2\pi) = 159$ hertz and a roll-off of 6 dB per octave after

159 Hz. There is also a phase shift centered around 159 Hz. In other words, the gain has both a real and an imaginary component. For transient analysis, the output is the convolution of the input waveform with the impulse response of $1/(1+.001 \cdot s)$. The impulse response is a decaying exponential with a time constant of 1 millisecond. This means that the output is the “lossy integral” of the input, where the loss has a time constant of 1 millisecond. The LAPLACE part shown in Figure 40 could be used for this purpose.

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

$$\begin{aligned} \text{NUM} &= 1 \\ \text{DENOM} &= 1 + .001 \cdot s \end{aligned}$$

The gain and phase characteristics are shown in Figure 41.

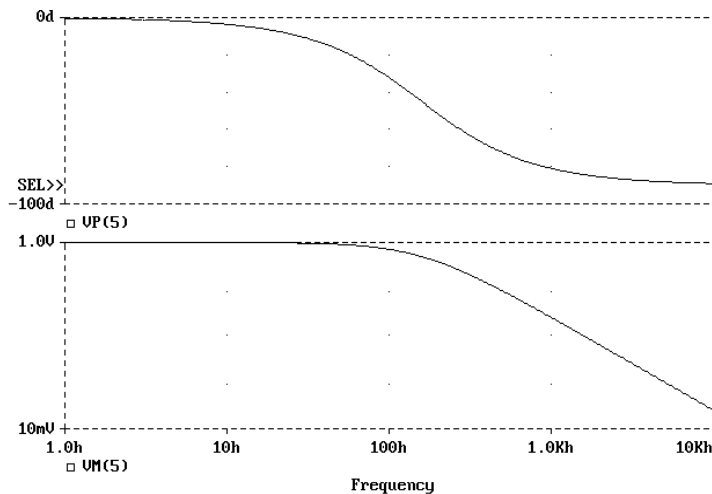


Figure 41 Viewing gain and phase characteristics of a lossy integrator.

This produces a PSpice A/D netlist declaration like this:

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

Example two

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

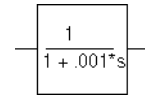


Figure 40 LAPLACE part example one.

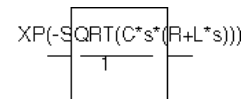


Figure 42 LAPLACE part example two.

lossy transmission line. R , L , and C are the resistance, inductance, and capacitance of the line per unit length.

If R is small, the characteristic impedance of such a line is $Z = ((R + j\omega L)/(j\omega C))^{1/2}$, the delay per unit length is $(L/C)^{1/2}$, and the loss in dB per unit length is $23 \cdot R/Z$. This could be represented by the device in Figure 42.

The parameters R , L , and C can be defined in a .PARAM statement contained in a model file. (Refer to the online *OrCAD PSpice A/D Reference Manual* for more information about using .PARAM statements.) More useful, however, is for R , L , and C to be arguments passed into a subcircuit. This part has the following characteristics:

$$\begin{aligned}\text{NUM} &= \text{EXP}(-\text{SQRT}(C*s*(R+L*s))) \\ \text{DENOM} &= 1\end{aligned}$$

This produces a PSpice A/D netlist declaration like this:

```
GLOSSY 5 0 LAPLACE {V(10)} = {exp(-sqrt(C*s*(R + L*s)))}
```

The Laplace transform parts are, however, an inefficient way, in both computer time and memory, to implement a delay. For ideal delays we recommend using the transmission line part instead.

Math functions

The ABM math function parts are shown in [Table 1](#). For each device, the corresponding template is shown, indicating the order in which the inputs are processed, if applicable.

Table 1 ABM math function parts

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

Math function parts are based on the PSpice A/D “E” device type. Each provides one or more inputs, and a mathematical function which is applied to the input. The result is output on the output net.

ABM expression parts

The expression parts are shown in [Table 2](#). These parts can be customized to perform a variety of functions depending on your requirements. Each of these parts has a set of four expression *building block* properties of the form:

$$\text{EXP}n$$

where $n = 1, 2, 3,$ or 4 .

During netlist generation, the complete expression is formed by concatenating the building block expressions in numeric order, thus defining the transfer function. Hence, the first expression fragment should be assigned to the EXP1 property, the second fragment to EXP2, and so on.

Expression properties can be defined using a combination of arithmetic operators and input designators. You may use any of the standard PSpice A/D arithmetic operators (see [Table 9 on page 3-110](#)) within an expression statement. You may also use the EXP n properties as variables to represent nets or constants.

Table 2 *ABM expression parts*

Part	Inputs	Output
ABM	none	V
ABM1	1	V
ABM2	2	V
ABM3	3	V
ABM/I	none	I
ABM1/I	1	I
ABM2/I	2	I
ABM3/I	3	I

The following examples illustrate a variety of ABM expression part applications.

Example one

Suppose you want to set an output voltage on net 4 to 5 volts times the square root of the voltage between nets 3 and 2. You could use an ABM2 part (which takes two inputs and provides a voltage output) to define a part like the one shown in Figure 43.

In this example of an ABM device, the output voltage is set to 5 volts times the square root of the voltage between net 3 and net 2. The property settings for this part are as follows:

```
EXP1 = 5V *
EXP2 = Sqrt(V(%IN2,%IN1))
```

This will produce a PSpice A/D netlist declaration like this:

```
ESQROOT 4 0 VALUE = {5V*Sqrt(V(3,2))}
```

Example two

GPSK is an oscillator for a PSK (Phase Shift Keyed) modulator. Current is pumped from net 11 through the source to net 6. Its value is a sine wave with an amplitude of 15 mA and a frequency of 10 kHz. The voltage at net 3 can shift the phase of GPSK by 1 radian / volt. Note the use of the TIME parameter in the EXP2 expression. This is the PSpice A/D internal sweep variable used in transient analyses. For any analysis other than transient, TIME = 0. This could be represented with an ABM1/I part (single input, current output) like the one shown in Figure 44.

This part is characterized by the following properties:

```
EXP1 = 15ma * SIN(
EXP2 = 6.28*10kHz*TIME
EXP3 = + V(%IN))
```

This produces a PSpice A/D netlist declaration like this:

```
GPSK 11 6 VALUE = {15MA*SIN(6.28*10kHz*TIME+V(3))}
```

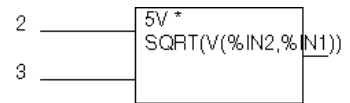


Figure 43 ABM expression part example one.

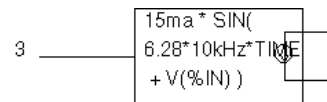


Figure 44 ABM expression part example two.

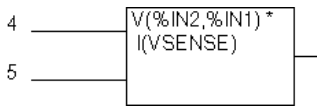


Figure 45 ABM expression part example three.

Example three

A device, EPWR, computes the instantaneous power by multiplying the voltage across nets 5 and 4 by the current through VSENSE. Sources are controlled by expressions which may contain voltages or currents or both. The ABM2 part (two inputs, current output) in Figure 45 could represent this.

This part is characterized by the following properties:

$$\begin{aligned} \text{EXP1} &= V(\%IN2, \%IN1) * \\ \text{EXP2} &= I(VSENSE) \end{aligned}$$

This produces a PSpice A/D netlist declaration like this:

```
EPWR 3 0 VALUE = {V(5,4)*I(VSENSE)}
```

Example four

The output of a component, GRATIO, is a current whose value (in amps) is equal to the ratio of the voltages at nets 13 and 2. If $V(2) = 0$, the output depends upon $V(13)$ as follows:

$$\begin{aligned} \text{if } V(13) &= 0, \text{ output} = 0 \\ \text{if } V(13) &> 0, \text{ output} = \text{MAXREAL} \\ \text{if } V(13) &< 0, \text{ output} = -\text{MAXREAL} \end{aligned}$$

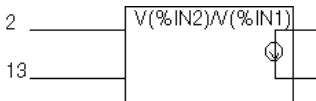


Figure 46 ABM expression part example four.

where MAXREAL is a PSpice A/D internal constant representing a very large number (on the order of $1e30$). In general, the result of evaluating an expression is limited to MAXREAL. This is modeled with an ABM2/I (two input, current output) part like this one in Figure 46.

This part is characterized by the following properties:

$$\text{EXP1} = V(\%IN2)/V(\%IN1)$$

Note that output of GRATIO can be used as part of the controlling function. This produces a PSpice A/D netlist declaration like this:

```
GRATIO 2 3 VALUE = {V(13)/V(2)}
```

Note *Letting a current approach $\pm 1e30$ will almost certainly cause convergence problems. To avoid this, use the limit function on the ratio to keep the current within reasonable limits.*

An instantaneous device example: modeling a triode

This section provides an example of using various ABM parts to model a triode vacuum tube. The schematic of the triode subcircuit is shown in Figure 47.

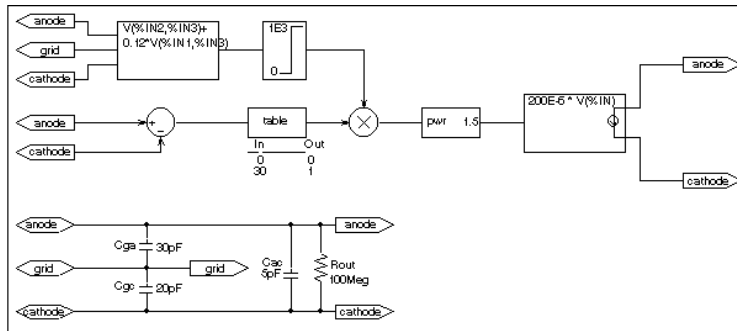


Figure 47 Triode circuit.

Assumptions: In its main operating region, the triode's current is proportional to the 3/2 power of a linear combination of the grid and anode voltages:

$$i_{\text{anode}} = k_0 \cdot (v_g + k_1 \cdot v_a)^{1.5}$$

For a typical triode, $k_0 = 200\text{e-}6$ and $k_1 = 0.12$.

Looking at the upper left-hand portion of the schematic, notice the a general-purpose ABM part used to take the input voltages from anode, grid, and cathode. Assume the following associations:

- $V(\text{anode})$ is associated with $V(\%IN1)$
- $V(\text{grid})$ is associated with $V(\%IN2)$
- $V(\text{cathode})$ is associated with $V(\%IN3)$

The expression property EXP1 then represents $V(\text{grid}, \text{cathode})$ and the expression property EXP2 represents $0.12[V(\text{anode}, \text{cathode})]$. When the template substitution is performed, the resulting VALUE is equivalent to the following:

$$V = V(\text{grid}, \text{cathode}) + 0.12 \cdot V(\text{anode}, \text{cathode})$$

The part would be defined with the following characteristics:

$$\begin{aligned}\text{EXP1} &= V(\%IN2, \%IN3) + \\ \text{EXP2} &= 0.12 * V(\%IN1, \%IN3)\end{aligned}$$

This works for the main operating region but does not model the case in which the current stays 0 when combined grid and anode voltages go negative. We can accommodate that situation as follows by adding the LIMIT part with the following characteristics:

$$\begin{aligned}\text{HI} &= 1\text{E}3 \\ \text{LO} &= 0\end{aligned}$$

This part instance, LIMIT1, converts all negative values of $v_g + .12 * v_a$ to 0 and leaves all positive values (up to 1 kV) alone. For a more realistic model, we could have used TABLE to correctly model how the tube turns off at 0 or at small negative grid voltages.

We also need to make sure that the current becomes zero when the anode alone goes negative. To do this, we can use a DIFF device, (immediately below the ABM3 device) to monitor the difference between V(anode) and V(cathode), and output the difference to the TABLE part. The table translates all values at or below zero to zero, and all values greater than or equal to 30 to one. All values between 0 and 30 are linearly interpolated. The properties for the TABLE part are as follows:

$$\begin{aligned}\text{ROW1} &= 00 \\ \text{ROW2} &= 301\end{aligned}$$

The TABLE part is a simple one, and ensures that only a zero value is output to the multiplier for negative anode voltages.

The output from the TABLE part and the LIMIT part are combined at the MULT multiplier part. The output of the MULT part is the product of the two input voltages. This value is then raised to the 3/2 or 1.5 power using the PWR part. The exponential property of the PWR part is defined as follows:

$$\text{EXP} = 1.5$$

The last major component is an ABM expression component to take an input voltage and convert it into a current. The relevant ABM1/I part property looks like this:

$$\text{EXP1} = 200\text{E-6} * \text{V}(\% \text{IN})$$

A final step in the model is to add device parasitics. For example, a resistor can be used to give a finite output impedance. Capacitances between the grid, cathode, and anode are also needed. The lower part of the schematic in Figure 47 shows a possible method for incorporating these effects. To complete the example, one could add a circuit which produces the family of I-V curves (shown in Figure 48).

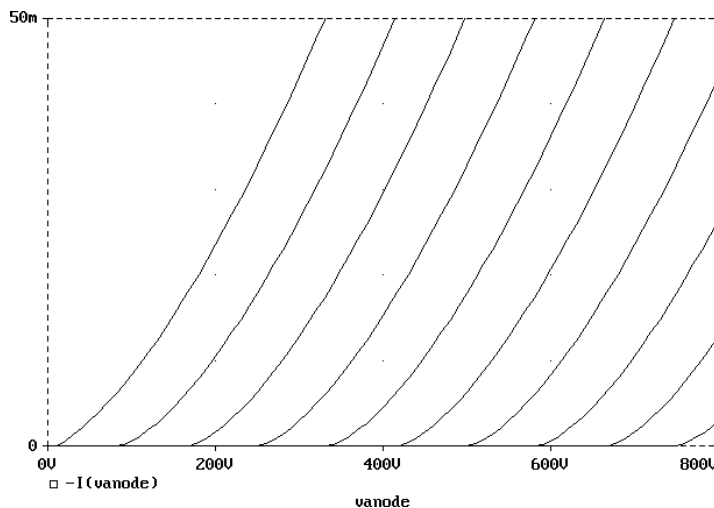


Figure 48 Triode subcircuit producing a family of I-V curves.

PSpice A / D-equivalent parts

PSpice A / D-equivalent parts respond to a differential input and have double-ended output. These parts reflect the structure of PSpice A / D E and G devices, thus having two pins for each controlling input and the output in the part. [Table 1](#) summarizes the PSpice A / D-equivalent parts available in the part library.

Table 1 *PSpice A/D-equivalent parts*

Category	Part	Description	Properties
Mathematical expression	EVALUE	general purpose	EXPR
	GVALUE		
	ESUM	special purpose	(none)
	GSUM		
	EMULT		
	GMULT		
Table look-up	ETABLE	general purpose	EXPR
	GTABLE		TABLE
Frequency table look-up	EFREQ	general purpose	EXPR
	GFREQ		TABLE
Laplace transform	ELAPLACE	general purpose	EXPR
	GLAPLACE		XFORM

There are no equivalent “F” or “H” part types in the part library because PSpice A/D “F” and “H” devices do not support the ABM extensions.

PSpice A / D-equivalent ABM parts can be classified as either E or G device types. The E part type provides a voltage output, and the G device type provides a current output. The device’s transfer function can contain any mixture of voltages and currents as inputs. Hence, there is no longer a division between voltage-controlled and current-controlled parts. Rather the part type is dictated only by the output requirements. If a voltage output is required, use an E part type. If a current output is necessary, use a G part type.

Each E or G part type in the ABM.OLB part file is defined by a template that provides the specifics of the transfer function. Other properties in the model definition can be edited to customize the transfer function. By default, the template cannot be modified directly choosing Properties from the Edit menu in Capture. Rather, the values for other properties (such as the expressions used in the template) are usually edited, then these values are substituted into the template. However, the part editor can be used to modify the template or designate the template as modifiable from within Capture. This way, custom parts can be created for special-purpose application.

Implementation of PSpice A/D-equivalent parts

Although you generally use Capture to place and specify PSpice A/D-equivalent ABM parts, it is useful to know the PSpice A/D command syntax for “E” and “G” devices. This is especially true when creating custom ABM parts since part templates must adhere to PSpice A/D syntax.

The general forms for PSpice A/D “E” and “G” extensions are:

```
E <name> <connecting nodes> <ABM keyword> <ABM function>
G <name> <connecting nodes> <ABM keyword> <ABM function>
```

where

<i><name></i>	is the device name appended to the E or G device type character
<i><connecting nodes></i>	specifies the <i><(+ node name, - node name)></i> pair between which the device is connected

<i><ABM keyword></i>	specifies the form of the transfer function to be used, as one of: VALUE arithmetic expression TABLE lookup table LAPLACE Laplace transform FREQ frequency response table CHEBYSHEV Chebyshev filter characteristics
<i><ABM function></i>	specifies the transfer function as a formula or lookup table as required by the specified <i><ABM keyword></i>

Refer to the online *OrCAD PSpice A/D Reference Manual* for detailed information.

Modeling mathematical or instantaneous relationships

The instantaneous models (using VALUE and TABLE extensions to PSpice A / D “E” and “G” devices in the part templates) enforce a direct response to the input at each moment in time. For example, the output might be equal to the square root of the input at every point in time. Such a device has no memory, or, a flat frequency response. These techniques can be used to model both linear and nonlinear responses.

Note *For AC analysis, a nonlinear device is first linearized around the bias point, and then the linear equivalent is used.*

EVALUE and GVALUE parts

The EVALUE and GVALUE parts allow an instantaneous transfer function to be written as a mathematical expression in standard notation. These parts take the input signal, perform the function specified by the EXPR property on the signal, and output the result on the output pins.

In controlled sources, EXPR may contain constants and parameters as well as voltages, currents, or time. Voltages may be either the voltage at a net, such as V(5), or the voltage across two nets, such as V(4,5). Currents must be the current through a voltage source (V device), for example, I(VSENSE). Voltage sources with a value of 0 are handy for sensing current for use in these expressions.

Functions may be used in expressions, along with arithmetic operators (+, -, *, and /) and parentheses. Available built-in functions are summarized in [Table 10 on page 3-111](#).

The EVALUE and GVALUE parts are defined, in part, by the following properties (default values are shown):

EVALUE	
EXPR	V(%IN+, %IN-)
GVALUE	
EXPR	V(%IN+, %IN-)

Sources are controlled by expressions which may contain voltages, currents, or both. The following examples illustrate customized EVALUE and GVALUE parts.

Example 1

In the example of an EVALUE device shown in Figure 49, the output voltage is set to 5 volts times the square root of the voltage between pins %IN+ and %IN-.

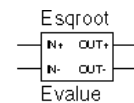
The property settings for this device are as follows:

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

Example 2

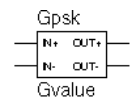
Consider the device in Figure 50. This device could be used as an oscillator for a PSK (Phase Shift Keyed) modulator.

A current through a source is a sine wave with an amplitude of 15 mA and a frequency of 10 kHz. The voltage at the input pin can shift the phase by 1 radian/volt. Note the use of the TIME parameter in this



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

Figure 49 EVALUE part example.



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

Figure 50 GVALUE part example.

expression. This is the PSpice A/D internal sweep variable used in transient analyses. For any analysis other than transient, $TIME = 0$. The relevant property settings for this device are shown below:

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

EMULT, GMULT, ESUM, and GSUM

The EMULT and GMULT parts provide output which is based on the product of two input sources. The ESUM and GSUM parts provide output which is based on the sum of two input sources. The complete transfer function may also include other mathematical expressions.

Example 1

Consider the device in Figure 51. This device computes the instantaneous power by multiplying the voltage across pins %IN+ and %IN- by the current through VSENSE. This device's behavior is built-in to the PSPICETEMPLATE property as follows (appears on one line):

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

You can use the part editor to change the characteristics of the template to accommodate additional mathematical functions, or to change the nature of the transfer function itself. For example, you may want to create a voltage divider, rather than a multiplier. This is illustrated in the following example.

Example 2

Consider the device in Figure 52.

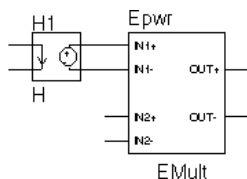
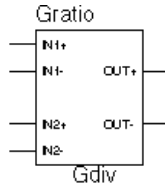


Figure 51 EMULT part example.



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

Figure 52 *GMULT part example.*

With this device, the output is a current is equal to the ratio of the voltages at input pins 1 and input pins 2. If $V(\%IN2+, \%IN2-) = 0$, the output depends upon $V(\%IN1+, \%IN1-)$ as follows:

- if $V(\%IN1+, \%IN1-) = 0$, output = 0
- if $V(\%IN1+, \%IN1-) > 0$, output = MAXREAL
- if $V(\%IN1+, \%IN1-) < 0$, output = -MAXREAL

where MAXREAL is a PSpice A/D internal constant representing a very large number (on the order of $1e30$). In general, the result of evaluating an expression is limited to MAXREAL. Note that the output of the part can also be used as part of the controlling function.

To create this device, you would first make a new part, GDIV, based on the GMULT part. Edit the GDIV template to divide the two input values rather than multiply them.

Lookup tables (ETABLE and GTABLE)

The ETABLE and GTABLE parts use a transfer function described by a table. These device models are well suited for use with measured data.

The ETABLE and GTABLE parts are defined in part by the following properties (default values are shown):

ETABLE

TABLE	(-15, -15), (15,15)
EXPR	$V(\%IN+, \%IN-)$

GTABLE

TABLE	(-15, -15), (15,15)
EXPR	V(%IN+, %IN-)

First, EXPR is evaluated, and that value is used to look up an entry in the table. EXPR is a function of the input (current or voltage) and follows the same rules as for VALUE expressions.

The table consists of pairs of values, the first of which is an input, and the second of which is the corresponding output. Linear interpolation is performed between entries. For values of EXPR outside the table's range, the device's output is a constant with a value equal to the entry with the smallest (or largest) input. This characteristic can be used to impose an upper and lower limit on the output.

An example of a table declaration (using the TABLE property) would be the following:

```
TABLE =  
+ (0, 0) (.02, 2.690E-03) (.04, 4.102E-03) (.06, 4.621E-03)  
+ (.08, 4.460E-03) (.10, 3.860E-03) (.12, 3.079E-03) (.14,  
+ 2.327E-03)  
+ (.16, 1.726E-03) (.18, 1.308E-03) (.20, 1.042E-03) (.22,  
+ 8.734E-04)  
+ (.24, 7.544E-04) (.26, 6.566E-04) (.28, 5.718E-04) (.30,  
+ 5.013E-04)  
+ (.32, 4.464E-04) (.34, 4.053E-04) (.36, 3.781E-04) (.38,  
+ 3.744E-04)  
+ (.40, 4.127E-04) (.42, 5.053E-04) (.44, 6.380E-04) (.46,  
+ 7.935E-04)  
+ (.48, 1.139E-03) (.50, 2.605E-03) (.52, 8.259E-03) (.54,  
+ 2.609E-02)  
+ (.56, 7.418E-02) (.58, 1.895E-01) (.60, 4.426E-01)
```

Frequency-domain device models

Frequency-domain models (ELAPLACE, GLAPLACE, EFREQ, and GFREQ) are characterized by output that depends on the current input as well as the input history. The relationship is therefore non-instantaneous. For example, the output may be equal to the integral of the input over time. In other words, the response depends upon frequency.

During AC analysis, the frequency response determines the complex gain at each frequency. During DC analysis and bias point calculation, the gain is the zero-frequency response. During transient analysis, the output of the device is the convolution of the input and the impulse response of the device.

Laplace transforms (LAPLACE)

The ELAPLACE and GLAPLACE parts allow a transfer function to be described by a Laplace transform function. The ELAPLACE and GLAPLACE parts are defined, in part, by the following properties (default values are shown):

ELAPLACE

EXPR	V(%IN+, %IN-)
XFORM	1 / s

GLAPLACE

EXPR	V(%IN+, %IN-)
XFORM	1 / s

The LAPLACE parts use a Laplace transform description. The input to the transform is the value of EXPR, where EXPR follows the same rules as for VALUE expressions (see [EVALUE and GVALUE parts on page 6-222](#)). XFORM is an expression in the Laplace variable, s. It follows the rules for standard expressions as described for VALUE expressions with the addition of the s variable.

Moving back and forth between the time and frequency-domains can cause surprising results. Often the results are quite different than what one would intuitively expect. For this reason, we strongly recommend familiarity with a reference on Fourier and Laplace transforms. A good one is:

- 1 R. Bracewell, *The Fourier Transform and Its Applications*, McGraw-Hill, Revised Second Edition (1986)

We also recommend familiarity with the use of transforms in analyzing linear systems. Some references on this subject:

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

Voltages, currents, and TIME cannot appear in a Laplace transform.

The output of the device depends on the type of analysis being done. For DC and bias point, the output is simply the zero frequency gain times the value of EXPR. The zero frequency gain is the value of XFORM with $s = 0$. For AC analysis, EXPR is linearized around the bias point (similar to the VALUE parts). The output is then the input times the gain of EXPR times the value of XFORM. The value of XFORM at a frequency is calculated by substituting $j \cdot \omega$ for s , where ω is $2\pi \cdot \text{frequency}$. For transient analysis, the value of EXPR is evaluated at each time point. The output is then the convolution of the past values of EXPR with the impulse response of XFORM. These rules follow the standard method of using Laplace transforms. We recommend looking at one or more of the references cited in [Frequency-domain device models on page 6-227](#) for more information.

Example

The input to the Laplace transform is the voltage across the input pins, or $V(\%IN+, \%IN-)$. The EXPR property may be edited to include constants or functions, as with other parts. The transform, $1/(1+.001 \cdot s)$, describes a simple, lossy integrator with a time constant of 1 millisecond. This can be implemented with an RC pair that has a time constant of 1 millisecond.

Using the part editor, you would define the XFORM and EXPR properties as follows:

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

The default template remains (appears on one line):

```
TEMPLATE= E^@REFDES %OUT+ %OUT- LAPLACE
{@EXPR}= (@XFORM)
```

After netlist substitution of the template, the resulting transfer function would become:

```
V(%OUT+, %OUT-) = LAPLACE {V(%IN+, %IN-)} = (1/(1+.001*s))
```

The output is a voltage and is applied between pins %OUT+ and %OUT-. For DC, the output is simply equal to the input, since the gain at $s = 0$ is 1.

For AC analysis, the gain is found by substituting $j\omega$ for s . This gives a flat response out to a corner frequency of $1000/(2\pi) = 159$ Hz and a roll-off of 6 dB per octave after 159 Hz. There is also a phase shift centered around 159 Hz. In other words, the gain has both a real and an imaginary component. The gain and phase characteristic is the same as that shown for the equivalent control system part example using the LAPLACE part (see Figure 41 on page 6-211).

For transient analysis, the output is the convolution of the input waveform with the impulse response of $1/(1+.001s)$. The impulse response is a decaying exponential with a time constant of 1 millisecond. This means that the output is the “lossy integral” of the input, where the loss has a time constant of 1 millisecond.

This will produce a PSpice A/D netlist declaration similar to:

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

Frequency response tables (EFREQ and GFREQ)

The EFREQ and GFREQ parts are described by a table of frequency responses in either the magnitude/phase domain or complex number domain. The entire table is read in and converted to magnitude in dB and phase in degrees. Interpolation is performed between entries. Phase is interpolated linearly; magnitude is interpolated logarithmically. For frequencies outside the table's range, 0 (zero) magnitude is used.

EFREQ and GFREQ properties are defined as follows:

EXPR	value used for table lookup; defaults to V(%IN+, %IN-) if left blank.
TABLE	series of either (input frequency, magnitude, phase) triplets, or (input frequency, real part, imaginary part) triplets describing a complex value; defaults to (0,0,0) (1Meg,-10,90) if left blank.
DELAY	group delay increment; defaults to 0 if left blank.
R_I	table type; if left blank, the frequency table is interpreted in the (input frequency, magnitude, phase) format; if defined with any value (such as YES), the table is interpreted in the (input frequency, real part, imaginary part) format.
MAGUNITS	units for magnitude where the value can be DB (decibels) or MAG (raw magnitude); defaults to DB if left blank.
PHASEUNITS	units for phase where the value can be DEG (degrees) or RAD (radians); defaults to DEG if left blank.

The DELAY property increases the group delay of the frequency table by the specified amount. The delay term is particularly useful when an EFREQ or GFREQ device generates a non-causality warning message during a transient analysis. The warning message issues a delay value that can be assigned to the part's DELAY property for subsequent runs, without otherwise altering the table.

The output of the device depends on the analysis being done. For DC and bias point, the output is simply the zero frequency magnitude times the value of EXPR. For AC analysis, EXPR is linearized around the bias point (similar to EVALUE and GVALUE parts). The output for each frequency is then the input times the gain of EXPR times the value of the table at that frequency. For transient

analysis, the value of EXPR is evaluated at each time point. The output is then the convolution of the past values of EXPR with the impulse response of the frequency response. These rules follow the standard method of using Fourier transforms. We recommend looking at one or more of the references cited in [Frequency-domain device models on page 6-227](#) for more information.

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

Figure 53 shows an EFREQ device used as a low pass filter. The input to the frequency response is the voltage across the input pins. The table describes a low pass filter with a response of 1 (0 dB) for frequencies below 5 kilohertz and a response of .001 (-60 dB) for frequencies above 6 kilohertz. The output is a voltage across the output pins.

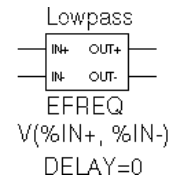


Figure 53 EFREQ part example.

This part is defined by the following properties:

```
TABLE = (0, 0, 0) (5kHz, 0, -5760) (6kHz, -60, -6912)
DELAY =
R_I =
MAGUNITS =
PHASEUNITS =
```

Since R_I, MAGUNITS, and PHASEUNITS are undefined, each table entry is interpreted as containing frequency, magnitude value in dB, and phase values in degrees. Delay defaults to 0.

The phase lags linearly with frequency meaning that this table exhibits a constant time (group) delay. The delay is necessary so that the impulse response is causal. That is, so that the impulse response does not have any significant components before time zero.

The constant group delay is calculated from the values for a given table entry as follows:

$$\text{group delay} = \text{phase} / 360 / \text{frequency}$$

For this example, the group delay is 3.2 msec ($6912 / 360 / 6k = 5760 / 360 / 6k = 3.2m$). An alternative specification for this table could be:

```
TABLE = (0, 0, 0) (5kHz, 0, 0) (6kHz, -60, 0)
DELAY = 3.2ms
```

```
R_I =
MAGUNITS =
PHASEUNITS =
```

This produces a PSpice A/D netlist declaration like this:

```
ELOWPASS 5 0 FREQ {V(10)} = (0,0,0) (5kHz,0,0) (6kHz-60,0)
+ DELAY = 3.2ms
```

Cautions and recommendations for simulation and analysis

Instantaneous device modeling

During AC analysis, nonlinear transfer functions are handled the same way as other nonlinear parts: each function is linearized around the bias point and the resulting small-signal equivalent is used.

Consider the voltage multiplier (mixer) shown in Figure 54. This circuit has the following characteristics:

```
Vin1:    DC=0v  AC=1v
Vin2:    DC=0v  AC=1v
```

where the output on net 3 is $V(1)*V(2)$.

During AC analysis, $V(3) = 0$ due to the 0 volts bias point voltage on nets 1, 2, and 3. The small-signal equivalent therefore has 0 gain (the derivative of $V(1)*V(2)$ with respect to both $V(1)$ and $V(2)$ is 0 when $V(1)=V(2)=0$). So, the output of the mixer during AC analysis will be 0 regardless of the AC values of $V(1)$ and $V(2)$.

Another way of looking at this is that a mixer is a nonlinear device and AC analysis is a linear analysis. The output of the mixer has 0 amplitude at the fundamental. (Output is nonzero at DC and twice the input frequency, but these are not included in a linear analysis.)

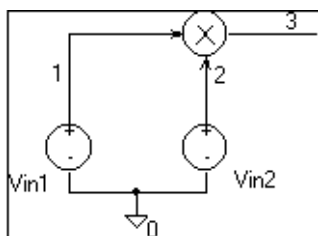


Figure 54 Voltage multiplier circuit (mixer).

If you need to analyze nonlinear functions, such as a mixer, use transient analysis. Transient analysis solves the full, nonlinear circuit equations. It also allows you to use input waveforms with different frequencies (for example, VIN1 could be 90 MHz and VIN2 could be 89.8 MHz). AC analysis does not have this flexibility, but in return it uses much less computer time.

Frequency-domain parts

Some caution is in order when moving between frequency and time domains. This section discusses several points that are involved in the implementation of frequency-domain parts. These discussions all involve the transient analysis, since both the DC and AC analyses are straightforward.

The first point is that there are limits on the maximum values and on the resolution of both time and frequency. These are related: the frequency resolution is the inverse of the maximum time and vice versa. The maximum time is the length of the transient analysis, TSTOP. Therefore, the frequency resolution is $1/TSTOP$.

Laplace transforms

For Laplace transforms, PSpice A/D starts off with initial bounds on the frequency resolution and the maximum frequency determined by the transient analysis parameters as follows. The frequency resolution is initially set below the theoretical limit to $(.25/TSTOP)$ and is then made as large as possible without inducing sampling errors. The maximum frequency has an initial upper bound of $(1/(REL TOL * TMAX))$, where TMAX is the transient analysis Step Ceiling value, and RELTOL is the relative accuracy of all calculated voltages and currents. If a Step Ceiling value is not specified,

Note $TSTOP$, $TMAX$, and $TSTEP$ values are configured using *Transient* on the *Setup* menu. The $RELTOL$ property is set using *Options* on the *Setup* menu.

PSpice A / D uses the Transient Analysis Print Step, $TSTEP$, instead.

PSpice A / D then attempts to reduce the maximum frequency by searching for the frequency at which the response has fallen to $RELTOL$ times the maximum response. For instance, for the transform:

$$1/(1+s)$$

the maximum response, 1.0, is at $s = j\omega = 0$ (DC). The cutoff frequency used when $RELTOL = .001$, is approximately $1000/(2\pi) = 159$ Hz. At 159 Hz, the response is down to .001 (down by 60 db). Since some transforms do not have such a limit, there is also a limit of $10/RELTOL$ times the frequency resolution, or $10/(RELTOL \cdot TSTOP)$. For example, consider the transform:

$$e^{-0.001 \cdot s}$$

This is an ideal delay of 1 millisecond and has no frequency cutoff. If $TSTOP = 10$ milliseconds and $RELTOL = .001$, then PSpice A / D imposes a frequency cutoff of 10 MHz. Since the time resolution is the inverse of the maximum frequency, this is equivalent to saying that the delay cannot resolve changes in the input at a rate faster than .1 microseconds. In general, the time resolution will be limited to $RELTOL \cdot TSTOP / 10$.

A final computational consideration for Laplace parts is that the impulse response is determined by means of an FFT on the Laplace expression. The FFT is limited to 8192 points to keep it tractable, and this places an additional limit on the maximum frequency, which may not be greater than 8192 times the frequency resolution.

If your circuit contains many Laplace parts which can be combined into a more complex single device, it is generally preferable to do this. This saves computation and memory since there are fewer impulse responses. It also reduces the number of opportunities for numerical artifacts that might reduce the accuracy of your transient analyses.

Laplace transforms can contain poles in the left half-plane. Such poles will cause an impulse response that increases with time instead of decaying. Since the transient analysis is always for a finite time span, PSpice A/D does not have a problem calculating the transient (or DC) response. However, such poles will make the actual device oscillate.

Non-causality and Laplace transforms

PSpice A/D applies an inverse FFT to the Laplace expression to obtain an impulse response, and then convolves the impulse response with the dependent source input to obtain the output. Some common impulse responses are inherently non-causal. This means that the convolution must be applied to both past and future samples of the input in order to properly represent the inverse of the Laplace expression.

For example, the expression $\{S\}$ corresponds to differentiation in the time domain. The impulse response for $\{S\}$ is an impulse pair separated by an infinitesimal distance in time. The impulses have opposite signs, and are situated one in the infinitesimal past, the other in the infinitesimal future. In other words, convolution with this corresponds to applying a finite-divided difference in the time domain.

The problem with this for PSpice A/D is that the simulator only has the present and past values of the simulated input, so it can only apply half of the impulse pair during convolution. This will obviously not result in time-domain differentiation. PSpice A/D can detect, but not fix this condition, and issues a non-causality warning message when it occurs. The message tells what percentage of the impulse response is non-causal, and how much delay would need to be added to slide the non-causal part into a causal region. $\{S\}$ is theoretically 50% non-causal. Non-causality on the order of 1% or less is usually not critical to the simulation results.

You can delay $\{S\}$ to keep it causal, but the separation between the impulses is infinitesimal. This means that a

very small time step is needed. For this reason, it is usually better to use a macromodel to implement differentiation.

Here are some guidelines:

- In the case of a Laplace device (ELAPLACE), multiply the Laplace expression by e to the $(-s * \text{<the suggested delay>})$.
- In the case of a frequency table (EFREQ or GFREQ), do either of the following:
 - Specify the table with `DELAY=<the suggested delay>`.
 - Compute the delay by adding a phase shift.

Chebyshev filters

All of the considerations given above for Laplace parts also apply to Chebyshev filter parts. However, PSpice A/D also attempts to deal directly with inaccuracies due to sampling by applying Nyquist criteria based on the highest filter cutoff frequency. This is done by checking the value of TMAX. If TMAX is not specified it is assigned a value, or if it is specified, it may be reduced.

For low pass and band pass filters, TMAX is set to $(0.5/FS)$, where FS is the stop band cutoff in the case of a low pass filter, or the upper stop band cutoff in the case of a band pass filter.

For high pass and band reject filters, there is no clear way to apply the Nyquist criterion directly, so an additional factor of two is thrown in as a safety margin. Thus, TMAX is set to $(0.25/FP)$, where FP is the pass band cutoff for the high pass case or the upper pass band cutoff for the band reject case. It may be necessary to set TMAX to something smaller if the filter input has significant frequency content above these limits.

Frequency tables

For frequency response tables, the maximum frequency is twice the highest value. It will be reduced to $10/(REL TOL \cdot TSTOP)$ or 8192 times the frequency resolution if either value is smaller.

The frequency resolution for frequency response tables is taken to be either the smallest frequency increment in the table or the fastest rate of phase change, whichever is least. PSpice A/D then checks to see if it can be loosened without inducing sampling errors.

Trading off computer resources for accuracy

There is a significant trade-off between accuracy and computation time for parts modeled in the frequency domain. The amount of computer time and memory scale approximately inversely to RELTOL. Therefore, if you can use RELTOL=.01 instead of the default .001, you will be ahead. However, this will not adversely affect the impulse response. You may also wish to vary TMAX and TSTOP, since these also come into play.

Since the trade-off issues are fairly complex, it is advisable to first simulate a small test circuit containing only the frequency-domain device, and then after proper validation, proceed to incorporate it in your larger design. The PSpice A/D defaults will be appropriate most of the time if accuracy is your main concern, but it is still worth checking.

Note *Do not set RELTOL to a value above 0.01. This can seriously compromise the accuracy of your simulation.*

Basic controlled sources

As with basic SPICE, PSpice A/D has basic controlled sources derived from the standard SPICE E, F, G, and H devices. [Table 1](#) summarizes the linear controlled source types provided in the standard part library.

Table 1 *Basic controlled sources in ANALOG.OLB*

Device type	Part name
Controlled Voltage Source (PSpice A/D E device)	E
Current-Controlled Current Source (PSpice A/D F device)	F
Controlled Current Source (PSpice A/D G device)	G
Current-Controlled Voltage Source (PSpice A/D H device)	H

Creating custom ABM parts

Create a custom part when you need a controlled source that is not provided in the special purpose set or that is more elaborate than you can build with the general purpose parts (with multiple controlling inputs, for example).

The transfer function can be built into the part two different ways:

- directly in the PSPICETEMPLATE definition.
- by defining the part's EXPR and related properties (if any).

The PSpice A/D syntax for declaring E and G devices can help you form a PSPICETEMPLATE definition.

Refer to your *OrCAD Capture User's Guide* for a description of how to create a custom part.

Refer to the online *OrCAD PSpice A/D Reference Manual* for more information about E and G devices.

Digital device modeling

7

Chapter overview

This chapter provides information about digital modeling, and includes the following sections:

- [Introduction on page 7-242](#)
- [Functional behavior on page 7-243](#)
- [Timing characteristics on page 7-251](#)
- [Input/Output characteristics on page 7-257](#)

Introduction

The standard part libraries contain a comprehensive set of digital parts in many different technologies. Each digital part is described electrically by a digital device model in the form of a subcircuit definition stored in a model library. The corresponding subcircuit name is defined by the part's MODEL attribute value. Other attributes—MNTYMXDLY, IO_LEVEL, and the PSPICEDEFAULTNET set—are passed to the subcircuit, thus providing a high-level means for influencing the behavior of the digital device model.

Generally, the digital parts provided in the part libraries are satisfactory for most circuit designs. However, if your design requires digital parts that are not already provided in OrCAD's part and model libraries, you need to define digital device models corresponding to the new digital parts.

A complete digital device model has three main characteristics:

- Functional behavior: described by the gate-level and behavioral digital primitives comprising the subcircuit.
- I/O behavior: described by the I/O model, interface subcircuits, and power supplies related to a logic family.
- Timing behavior: described by one or more timing models, pin-to-pin delay primitives, or constraint checker primitives.

These characteristics are described in this chapter with a running example demonstrating the use of gate-level primitives.

Functional behavior

A digital device model's functional behavior is defined by one or more interconnected digital primitives. Typically, a logic diagram in a data book can be implemented directly using the primitives provided by PSpice A/D. The table below provides a summary of the digital primitives.

Table 2 *Digital primitives summary*

Type	Description
Standard gates	
BUF	buffer
INV	inverter
AND	AND gate
NAND	NAND gate
OR	OR gate
NOR	NOR gate
XOR	exclusive OR gate
NXOR	exclusive NOR gate
BUFA	buffer array
INVA	inverter array
ANDA	AND gate array
NANDA	NAND gate array
ORA	OR gate array
NORA	NOR gate array
XORA	exclusive OR gate array
NXORA	exclusive NOR gate array
AO	AND-OR compound gate
OA	OR-AND compound gate
AOI	AND-NOR compound gate
OA	OR-NAND compound gate

Table 2 *Digital primitives summary (continued)*

Type	Description
Tristate gates	
BUF3	buffer
INV3	inverter
AND3	AND gate
NAND3	NAND gate
OR3	OR gate
NOR3	NOR gate
XOR3	exclusive OR gate
NXOR3	exclusive NOR gate
BUF3A	buffer array
INV3A	inverter array
AND3A	AND gate array
NAND3A	NAND gate array
OR3A	OR gate array
NOR3A	NOR gate array
XOR3A	exclusive OR gate array
NXOR3A	exclusive NOR gate array
Bidirectional transfer gates	
NBTG	N-channel transfer gate
PBTG	P-channel transfer gate
Flip-flops and latches	
JKFF	J-K, negative-edge triggered
DFF	D-type, positive-edge triggered
SRFF	S-R gated latch
DLTCH	D gated latch
Pullup / pulldown resistors	
PULLUP	pullup resistor array
PULLDN	pulldown resistor array
Delay lines	
DLYLINE	delay line

Table 2 *Digital primitives summary (continued)*

Type	Description
Programmable logic arrays	
PLAND	AND array
PLOR	OR array
PLXOR	exclusive OR array
PLNAND	NAND array
PLNOR	NOR array
PLNXOR	exclusive NOR array
PLANDC	AND array, true and complement
PLORC	OR array, true and complement
PLXORC	exclusive OR array, true and complement
PLNANDC	NAND array, true and complement
PLNORC	NOR array, true and complement
PLNXORC	exclusive NOR array, true and complement
Memory	
ROM	read-only memory
RAM	random access read-write memory
Multi-Bit A/D & D/A Converters	
ADC	multi-bit A/D converter
DAC	multi-bit D/A converter
Behavioral	
LOGICEXP	logic expression
PINDLY	pin-to-pin delay
CONSTRAINT	constraint checking

The format for digital primitives is similar to that for analog devices. One difference is that most digital primitives require two models instead of one:

- The *timing model*, which specifies propagation delays and timing constraints such as setup and hold times.

- The *I/O model*, which specifies information specific to the device's input/output characteristics.

The reason for having two models is that, while timing information is specific to a device, the input/output characteristics are specific to a whole logic family. Thus, many devices in the same family reference the same I/O model, but each device has its own timing model.

Figure 55 presents an overview of a digital device definition in terms of its primitives and underlying model attributes. These models are discussed further on [Timing model on page 7-251](#) and [Input/Output model on page 7-257](#).

For specific information on each primitive type see the online *OrCAD PSpice A/D Reference Manual*.

Note that some digital primitives, such as pullups, do not have Timing models. See [Timing model on page 7-251](#) for more information.

Digital primitive syntax

The general digital primitive format is shown below.

```
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>]
```

where

```
<primitive type> [( <parameter value>* )]
```

is the type of digital device, such as NAND, JKFF, or INV. It is followed by zero or more parameters specific to the primitive type, such as number of inputs. The number and meaning of the parameters depends on the primitive type.

```
<digital power node> <digital ground node>
```

are the nodes used by the interface subcircuits which connect analog nodes to digital nodes or vice versa.

```
<node>*
```

is one or more input and output nodes. The number of nodes depends on the primitive type and its parameters. Analog devices, digital devices, or both may be connected to a node. If a node has both analog and digital connections, then PSpice A/D

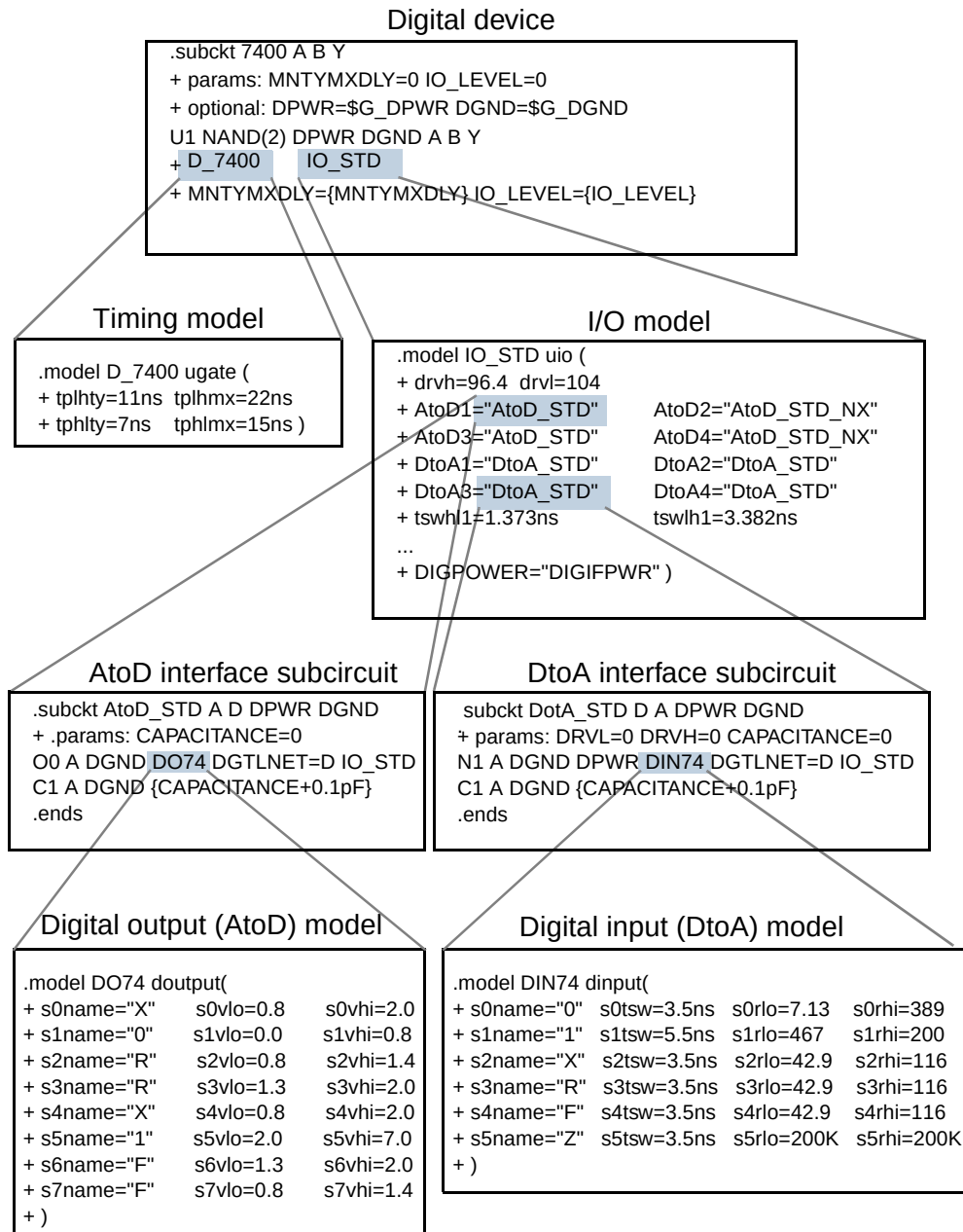


Figure 55 Elements of a digital device definition

This type of Timing model and its parameters are specific to each primitive type and are discussed in the online *OrCAD PSpice A/D Reference Manual*.

See [Input/Output model on page 7-257](#) for more information.

automatically inserts an interface subcircuit to translate between digital output states and voltages.

<Timing model name>

is the name of a timing model that describes the device's timing characteristics, such as propagation delay and setup and hold times. Each timing parameter has a minimum, typical, and maximum value which may be selected during analysis setup.

<I/O model name>

is the name of an I/O model that describes the device's loading and driving characteristics. I/O models also contain the names of up to four DtoA and AtoD interface subcircuits, which are automatically called by PSpice A/D to handle analog/digital interface nodes.

MNTYMXDLY

is an optional device parameter that selects either the minimum, typical, or maximum delay values from the device's timing model. If not specified, MNTYMXDLY defaults to 0. Valid values are:

- 0 = the current value of the circuit-wide DIGMNTYMX option (default=2)
- 1 = minimum
- 2 = typical
- 3 = maximum
- 4 = worst-case timing (min-max)

IO_LEVEL

is an optional device parameter that selects one of the four AtoD or DtoA interface subcircuits from the device's I/O model. PSpice A/D calls the selected subcircuit automatically in the event a node connecting to the primitive also connects to an analog device. If not specified, IO_LEVEL defaults to 0. Valid values are:

- 0 = the current value of the circuit-wide DIGIOLVL option (default=1)
- 1 = AtoD1/DtoA1
- 2 = AtoD2/DtoA2
- 3 = AtoD3/DtoA3
- 4 = AtoD4/DtoA4

Following are some simple examples of “U” device declarations:

```
U1 NAND(2) $G_DPWR $G_DGND 1 2 10 DO_GATE IO_DFT
U2 JKFF(1) $G_DPWR $G_DGND 3 5 200 3 3 10 2 D_293ASTD
+ IO_STD
U3 INV $G_DPWR $G_DGND IN OUT D_INV IO_INV MNTYMXDLY=3
+ IO_LEVEL=2
```

For example, the 74393 part could be defined as a subcircuit composed of “U” devices as shown 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 DO_GATE IO_STD
+ IO_LEVEL={IO_LEVEL}
U1 jkff(1) DPWR DGND
+ $D_HI CLRBAR A $D_HI $D_HI
+ QA_BUF $D_NC D_393_1 IO_STD
+ MNTYMXDLY={MNTYMXDLY}
+ IO_LEVEL={IO_LEVEL}
U2 jkff(1) DPWR DGND
+ $D_HI CLRBAR QA_BUF $D_HI $D_HI
+ QB_BUF $D_NC D_393_2 IO_STD
+ MNTYMXDLY={MNTYMXDLY}
U3 jkff(1) DPWR DGND
+ $D_HI CLRBAR QB_BUF $D_HI $D_HI
+ QC_BUF $D_NC D_393_2 IO_STD
+ MNTYMXDLY={MNTYMXDLY}
U4 jkff(1) DPWR DGND
+ $D_HI CLRBAR QC_BUF $D_HI $D_HI
+ QD_BUF $D_NC D_393_3 IO_STD
+ MNTYMXDLY={MNTYMXDLY}
```

```
UBUFF bufa(4) DPWR DGND
+ QA_BUF QB_BUF QC_BUF QD_BUF
+ QA QB QC QD D_393_4 IO_STD
+ MNTYMXDLY={MNTYMXDLY}
+ IO_LEVEL={IO_LEVEL}
.ends
```

When adding digital parts to the part libraries, you must create corresponding digital device models by connecting U devices in a subcircuit definition similar to the one shown above. OrCAD recommends you save these in your own custom model library, which you can then configure for use with a given design.

Timing characteristics

A digital device model's timing behavior can be defined in one of two ways:

- Most primitives have an associated Timing model, in which propagation delays and timing constraints (such as setup / hold times) are specified. This method is used when it is easy to partition delays among individual primitives; typically when the number of primitives is small.
- Use the PINDLY and CONSTRAINT primitives, which can directly model pin-to-pin delays and timing constraints for the whole device model. With this method, all other functional primitives operate in zero delay.

In addition to explicit propagation delays, other factors, such as output loads, can affect the total propagation delay through a device.

Refer to the online *OrCAD PSpice A/D Reference Manual* for a detailed discussion on these two primitives.

Timing model

With the exception of the PULLUP, PULLDN, and PINDLY devices, all digital primitives have a Timing model which provides timing parameters to the simulator. The Timing model for each primitive type is unique. That is, the model name and the parameters that can be defined for that model vary with the primitive type.

Within a Timing model, there may be one or more types of parameters:

- Propagation delays (TP)
- Setup times (TSU)
- Hold times (TH)
- Pulse widths (TW)
- Switching times (TSW)

Each parameter is further divided into three values: minimum (MN), typical (TY), and maximum (MX). For example, the typical low-to-high propagation delay on a gate is specified as the parameter TPLHTY. The minimum data-to-clock setup time on a flip-flop is specified as the parameter TSUDCLKMN.

Several timing models are used by digital device 74393 from the model libraries. One of them, D_393_1, is shown below for an edge-triggered flip-flop.

```
.model D_393_1 ueff (
+  tppcqh1ty=18ns      tppcqh1mx=33ns
+  tpc1kq1hty=6ns      tpc1kq1hmx=14ns
+  tpc1kqh1ty=7ns      tpc1kqh1mx=14ns
+  twc1khmn=20ns       twc1klmn=20ns
+  twpc1mn=20ns        tsudc1kmn=25ns
+ )
```

For a description of Timing model parameters, see the specific primitive type under U devices in the online *OrCAD PSpice A/D Reference Manual*.

When creating your own digital device models, you can create Timing models like these for the primitives you are using. OrCAD recommends that you save these in your own custom model library, which you can then configure for use with a given design.

One or more parameters may be missing from the Timing model definition. Data books do not always provide all three (minimum, typical, and maximum) timing specifications. The way the simulator handles missing parameters depends on the type of parameter.

Note This discussion applies only to propagation delay parameters (TP). All other timing parameters, such as setup/hold times and pulse widths are handled differently, and are discussed in the following section.

Treatment of unspecified propagation delays

Often, only the typical and maximum delays are specified in data books. If, in this case, the simulator were to assume that the unspecified minimum delay defaults to zero, the logic in certain circuits could break down.

For this reason, the simulator provides two configurable options, DIGMNTYSCALE and DIGTYMXSCALE, which are used to extrapolate unspecified propagation delays in the Timing models.

DIGMNTYSCALE

This option computes the minimum delay when a typical delay is known, using the formula:

$$TP_{xxMN} = DIGMNTYSCALE \cdot TP_{xxTY}$$

DIGMNTYSCALE defaults to the value 0.4, or 40% of the typical delay. Its value must be between 0.0 and 1.0.

DIGTYMXSCALE

This option computes the maximum delay from a typical delay, using the formula

$$TP_{xxMX} = DIGTYMXSCALE \cdot TP_{xxTY}$$

DIGTYMXSCALE defaults to the value 1.6. Its value must be greater than 1.0.

When a typical delay is unspecified, its value is derived from the minimum and / or maximum delays, in one of the following ways. If both the minimum and maximum delays are known, the typical delay is the average of these two values. If only the minimum delay is known, the typical delay is derived using the value of the DIGMNTYSCALE option. Likewise, if only the maximum delay is specified, the typical delay is derived using DIGTYMXSCALE. Obviously, if no values are specified, all three delays will default to zero.

Treatment of unspecified timing constraints

The remaining timing constraint parameters are handled differently than the propagation delays. Often, data books state pulse widths, setup times, and hold times as a minimum value. These parameters do not lend themselves to the extrapolation method used for propagation delays.

Instead, when one or more timing constraints are omitted, the simulator uses the following steps to fill in the missing values:

- If the minimum value is omitted, it defaults to zero.

- If the maximum value is omitted, it takes on the typical value if one was specified, otherwise it takes on the minimum value.
- If the typical value is omitted, it is computed as the average of the minimum and maximum values.

Propagation delay calculation

The timing characteristics of digital primitives are determined by both the timing models and the I/O models. Timing models specify propagation delays and timing constraints such as setup and hold times. I/O models specify input and output loading, driving resistances, and switching times.

When a device's output connects to another digital device, the total propagation delay through a device is determined by adding the loading delay (on the output terminal) to the delay specified in the device's timing model. Loading delay is calculated from the total load on the output and the device's driving resistances. The total load on an output is found by summing the output and input loads (OUTLD and INLD in the I/O model) of all devices connected to that output. This total load, combined with the device's driving resistances (DRVL and DRVH in the I/O model), allows the loading delay to be calculated:

$$\text{Loading delay} = R_{\text{DRIVE}} \cdot C_{\text{TOTAL}} \cdot \ln(2)$$

The loading delay is calculated for each output terminal of every device before the simulation begins. The total propagation delay is easily calculated during the simulation by adding the pre-calculated loading delay to the device's timing delay. However, for any individual timing delay specification (e.g., TPLH) having a value of 0, the loading delay is *not used*.

See [Input/Output characteristics on page 7-257](#) for more information.

When outputs connect to analog devices, the propagation delay is reduced by the switching times specified in the I/O model.

Inertial and transport delay

The simulator uses two different types of internal delay functions when simulating the digital portion of the circuit: *inertial delay* and *transport delay*. The application of these concepts is embodied within the implementation of the digital primitives within the simulator. Therefore, they are not user-selectable.

Inertial delay

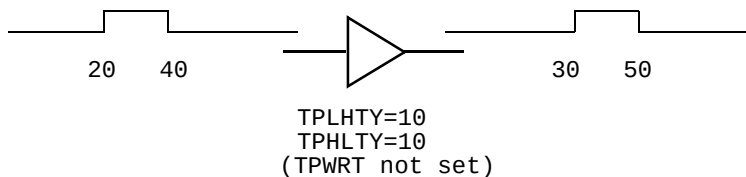
The simulation of a device may be described as the application of some *stimulus* (S) to a *function* (F) and predicting the *response* (R).



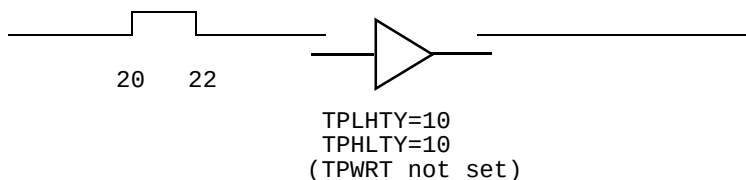
If this device is electrical in nature, application of the stimulus implies that energy will be imparted to the device to cause it to change state. The amount of such energy is a function of the signal's amplitude and duration. If the stimulus is applied to the device for a length of time that is too short, the device will not switch. The minimum duration required for an input change to have an effect on a device's output state is called the *inertial delay* of the device. For digital simulation, all delay parameters specified in timing models are considered inertial, with the exception of the delay line primitive, DLYLINE.

To model the noise immunity behavior of digital devices correctly, the TPWRT (pulse width rejection threshold) parameter can be set in the digital device's I/O model. When *pulse width* $\geq$ TPWRT and *pulse width* $<$ *propagation delay*, then the device generates either a 0-R-0, 1-F-1, or an X pulse.

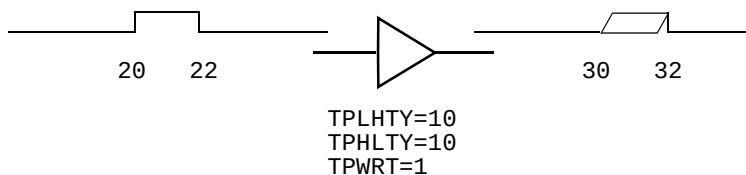
This example shows normal operation in which a pulse of 20 nsec width is applied to a BUF primitive having propagation delays of 10 nsec. TPWRT is not set.



The same device with a short pulse applied produces no output change.



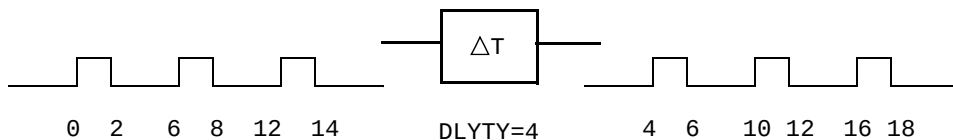
However, if TPWRT is assigned a numerical value (1 or 2 for this example), then the device outputs a glitch.



See the DLYLINE digital primitive in the online *OrCAD PSpice A/D Reference Manual*.

Transport delay

The delay line primitive is the only simulator model that can propagate any width pulse applied to its input. Its function is to skew the applied stimulus by some constant time value. For example:



Input / Output characteristics

A digital device model's input/output characteristics are defined by the I/O model that it references. Some characteristics, such as output drive resistance and loading capacitances, apply to digital simulation. Others, such as the interface subcircuits and the power supplies, apply only to mixed analog/digital simulation.

This section describes in detail:

- the I/O model
- the relationship between drive resistances and output strengths
- charge storage on digital nets
- the format of the interface subcircuits

Input/Output model

I/O models are common to entire logic families. For example, in the model libraries, there are only four I/O models for the entire 74LS family: IO_LS, for standard inputs and outputs; IO_LS_OC, for standard inputs and open-collector outputs; IO_LS_ST, for Schmitt trigger inputs and standard outputs; and IO_LS_OC_ST, for Schmitt trigger inputs and open-collector outputs. In contrast, timing models are unique to each device.

I/O models are specified as

```
.MODEL <I/O model name> UIO [model parameters]*
```

where valid model parameters are described in [Table 3](#).

INLD and OUTLD

These are used in the calculation of loading capacitance, which factors into the propagation delay discussed under timing models on [Timing model on page 7-251](#). Note that INLD does not apply to stimulus generators because they have no input nodes.

DRVH and DRVL

These are used to determine the strength of the output. Strengths are discussed on [Defining Output Strengths on page 7-262](#).

DRVZ, INR, and TSTOREMN

These are used to determine which nets should be simulated as charge storage nets. These are discussed in [Charge storage nets on page 7-264](#).

TPWRT

This is used to specify the pulse width above which the noise immunity behavior of a device is to be considered. See [Inertial delay on page 7-255](#) on inertial delay for detail.

The following UIO model parameters are needed only when creating models for use in mixed-signal simulations, and therefore only apply to PSpice A / D simulations.

AtoD1 through AtoD4, and DtoA1 through DtoA4

These are used to hold the names of interface subcircuits. Note that AtoD1 through AtoD4 do not apply to stimulus generators because digital stimuli have no input nodes.

DIGPOWER

This is used to specify the name of the digital power supply PSpice A / D should call if one of the AtoD or DtoA interface subcircuits is called.

TSWLH_n and TSWHL_n

These switching times are subtracted from a device's propagation delay on the outputs which connect to interface nodes. This compensates for the time it takes the DtoA device to change its output voltage from its current level to that of the switching threshold. By subtracting the switching time from the propagation delay, the analog signal reaches the switching threshold at the correct time (that is, at the exact time of the digital transition). The values for these model parameters should be obtained by measuring the time it takes the analog output of the DtoA (with a nominal analog load attached) to change to the switching threshold after its digital input changes. If the switching time is larger than the propagation delay for an output, no warning is issued, and a delay of zero is used.

When creating your own digital device models, you can create I/O models like these for the primitives you are using. OrCAD recommends that you save these in your own custom model library, which you can then configure for use with a given design.

Note *The switching time parameters are not used when the output drives a digital node.*

See the online *OrCAD PSpice A/D Reference Manual* for more information on units and defaults for these parameters.

Table 3 *Digital I/O model parameters*

UIO model parameter	Description
INLD	input load capacitance
OUTLD	output load capacitance
DRVH	output high level resistance
DRVL	output low level resistance
DRVZ	output Z-state leakage resistance
INR	input leakage resistance
TSTOREMN	minimum storage time for net to be simulated as a charge
TPWRT	pulse width rejection threshold
AtoD1 (Level 1)	name of AtoD interface subcircuit
DtoA1 (Level 1)	name of DtoA interface subcircuit
AtoD2 (Level 2)	name of AtoD interface subcircuit
DtoA2 (Level 2)	name of DtoA interface subcircuit
AtoD3 (Level 3)	name of AtoD interface subcircuit
DtoA3 (Level 3)	name of DtoA interface subcircuit
AtoD4 (Level 4)	name of AtoD interface subcircuit
DtoA4 (Level 4)	name of DtoA interface subcircuit
DIGPOWER	name of power supply subcircuit
TSWLH1	switching time low to high for DtoA1
TSWLH2	switching time low to high for DtoA2
TSWLH3	switching time low to high for DtoA3
TSWLH4	switching time low to high for DtoA4
TSWHL1	switching time high to low for DtoA1

Table 3 *Digital I/O model parameters (continued)*

UIO model parameter	Description
TSWHL2	switching time high to low for DtoA2
TSWHL3	switching time high to low for DtoA3
TSWHL4	switching time high to low for DtoA4

The digital primitives comprising the 74393 part reference the IO_STD I/O model in the model libraries as shown:

```
.model IO_STD uio (
+   drvh=96.4   drvl=104
+   AtoD1="AtoD_STD"   AtoD2="AtoD_STD_NX"
+   AtoD3="AtoD_STD"   AtoD4="AtoD_STD_NX"
+   DtoA1="DtoA_STD"   DtoA2="DtoA_STD"
+   DtoA3="DtoA_STD"   DtoA4="DtoA_STD"
+   tswlh1=1.373ns      tswlh1=3.382ns
+   tswlh2=1.346ns      tswlh2=3.424ns
+   tswlh3=1.511ns      tswlh3=3.517ns
+   tswlh4=1.487ns      tswlh4=3.564ns
+ )
```

Defining Output Strengths

The goal of running simulations is to calculate values for each node in the circuit. For analog nodes, the values are voltages. For digital nodes, these values are *states*. The state of a digital node is calculated from the output *strengths* of the devices driving the node and the logic level of the node.

Node strength calculations are described in [Chapter 14, Digital simulation](#).

The purpose of strengths is to allow the simulator to find the value of a node when more than one output is driving it. A common example is a bus line which is driven by more than one tristate driver. Under normal circumstances, all drivers except one are driving at the Z (high impedance) strength. Thus, the bus line will take on the value of the one gate that is driving at a higher strength (lower impedance).

Another example is a bus line connected to several open collector output devices and a digital pullup resistor. The pullup resistor outputs a 1 level at a weak (but non-Z) strength. If all of the open-collector devices are outputting at Z strength, then the node will have a 1 level because of the pullup resistor. If any of the open collectors output a 0, at a higher strength than the pullup resistor, then the 0 will overpower the weak 1 from the pullup, and the node will be a 0 level.

Configuring the strength scale

The 64 strengths are determined by two configurable options: DIGDRVZ and DIGDRVF. DIGDRVZ defines the impedance of the Z strength, and DIGDRVF defines the impedance of the forcing strength. These two values define a logarithmic scale consisting of 64 *ranges* of impedance values. By default, DIGDRVZ is 20 kohms and DIGDRVF is 2 ohms. The larger the range between DIGDRVZ and DIGDRVF, the larger the range of impedance values in each of the 64 strengths.

You can set these options in the Simulation Settings dialog box in PSpice A/D.

Determining the strength of a device output

The simulator uses the value of the DRVH (high-level driving resistance) or DRVL (low-level driving resistance) parameters from the device's I/O model. If the level of the output is a 1, the simulator obtains the strength by finding the *bin* which contains the value of the DRVH parameter. Likewise, if the level is a 0, the simulator uses the value of the DRVL parameter to obtain the strength.

See [Input/Output model on page 7-257](#) for more information.

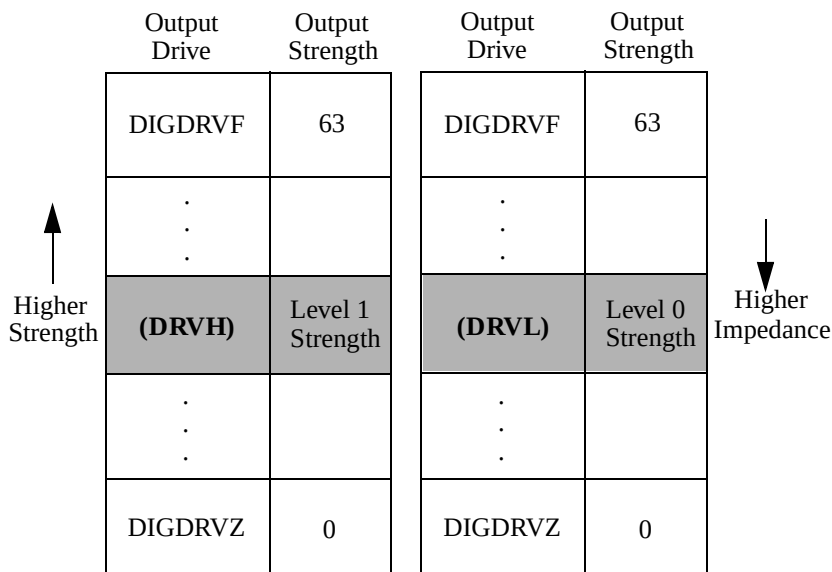


Figure 56 Level 1 and 0 strength determination.

Note that if the values of DRVH and DRVL in the I/O model are different, it is possible for the 1 and 0 levels to have different strengths. This is useful for open-collector devices, where the 0 level is at a higher strength than the 1 level (which drives at the Z strength).

Drive impedances which are higher than the value of DIGDRVZ are assigned the Z strength (0). Likewise, drive impedances lower than the value of DIGDRVF are assigned the forcing strength (63).

Controlling overdrive

During a simulation, the simulator uses only the strength range number (0-63) to compare the driving strength of outputs. The simulator allows you to control how much *stronger* an output must be before it overdrives the other outputs driving the same node. This is controlled with the configurable DIGOVRDRV option. By default, DIGOVRDRV is 3, meaning that the strength value assigned to an output must be at least 3 greater than all other drivers before it determines the level of the node.

The accuracy of the DIGOVRDRV strength comparison is limited by the size of the strength range, DIGDRVZ through DIGDRVF. The default drive range of 2 ohms to 20,000 ohms gives strength ranges of 7.5%. The accuracy of the strength comparison is 15%. In other words, depending on the particular values of DRVH and DRVL, it might take as much as a factor of 3.45 to overdrive a signal, or as little as a factor of 2.55. The accuracy of the comparison increases as the ratio between DIGDRVF and DIGDRVZ decreases.

You can set these options in the Simulation Settings dialog box in PSpice A/D.

Charge storage nets

The ability to model charge storage on digital nets is useful for engineers who are designing dynamic MOS integrated circuits. In such circuits, it is common for the designer to temporarily store a one or zero on a net by driving the net to the appropriate voltage and then

turning off the drive. The charge which is trapped on the net causes the net's voltage to remain unchanged for some time after the net is no longer driven. The technique is not normally used on PCB nets because sub-nanoampere input and output leakage currents would be required, as well as low coupling from adjacent signals.

The simulator models the stored charge nets using a simplified *switch-level* simulation technique. A normalized (with respect to power supply) charge or discharge current is calculated for each output or transfer gate attached to the net. This current, divided by the net's total capacitance, is integrated and recalculated at intervals which are appropriate for the particular net. The net's digital level is determined by the normalized voltage on the net. Only the digital level (1, 0, R, F, X) on the net is used by device inputs attached to the net.

This technique allows accurate simulation of networks of transfer gates and capacitive loads. The sharing of charge among several nets which are connected by transfer gates is handled properly because the simulation method calculates the charge transferred between the nets, and maintains a floating-point value for the charge on the net (not just a one or zero). Because of the increased computation, it takes the simulator longer to simulate charge storage nets than normal digital nets. However, charge storage nets are simulated much faster than analog nets.

The I/O model parameters INR, DRVZ, and TSTOREMN (see [Table 3 on page 7-260](#)) are used by the simulator to determine which nets should be simulated as charge storage nets. The simulator will simulate charge storage only for a net which has some devices attached to it which can be high impedance (Z), and which has a storage time greater than or equal to the smallest TSTOREMN of all

inputs attached to the net. The storage time is calculated as the total capacitance (sum of all INLD and OUTLD values for attached inputs and outputs) multiplied by the total leakage resistance for the net (the parallel combination of all INR and DRVZ values for attached inputs and outputs).

Note *The default values provided by the UIO model will **not** allow the use of charge-storage simulation techniques—even with circuits using non-OrCAD libraries of digital devices. This is appropriate, since these libraries are usually for PCB-based designs.*

Creating your own interface subcircuits for additional technologies

If you are creating custom digital parts for a technology which is not in the model libraries, you may also need to create AtoD and DtoA subcircuits. The new subcircuits need to be referenced by the I/O models for that technology. The AtoD and DtoA interfaces have specific formats, such as node order and parameters, which are expected by PSpice A/D for mixed-signal simulations.

If you are creating parts in one of the logic families *already* in the model libraries, you should reference the existing I/O models appropriate to that family. The I/O models, in turn, automatically reference the correct interface subcircuits for that family. These, too, are already contained in the model libraries.

The AtoD interface subcircuit format is shown here:

```
.SUBCKT ATOD <name suffix>
+ <analog input node>
+ <digital output node>
+ <digital power supply node>
+ <digital ground node>
+ PARAMS: CAPACITANCE=<input load value>
+ {0 device, loading capacitor, and other
+ declarations}
.ENDS
```

It has four nodes as described. The AtoD subcircuit has one parameter, CAPACITANCE, which corresponds to the input load. PSpice A/D passes the value of the I/O model parameter INLD to this parameter when the interface subcircuit is called.

The DtoA interface subcircuit format is shown here:

```
.SUBCKT DTOA <name suffix>
+ <digital input node> <analog output node>
+ <digital power supply node> <digital
+ ground node>
+ PARAMS:  DRVL=<0 level driving resistance>
+ DRVH=<1 level driving resistance>
+ CAPACITANCE=<output load value>
+ {N device, loading capacitor, and other
+ declarations}
.ENDS
```

It also has four nodes. Unlike the AtoD subcircuit, the DtoA subcircuit has three parameters. PSpice A/D will pass the values of the I/O model parameters DRVL, DRVH, and OUTLD to the interface subcircuit's DRVL, DRVH, and CAPACITANCE parameters when it is called.

The library file DIG_IO.LIB contains the I/O models and interface subcircuits for all logic families supported in the model libraries. You should refer to this file for examples of the I/O models, interface subcircuits, and the proper use of N and O devices.

Shown below are the I/O model and AtoD interface subcircuit definition used by the primitives describing the 74393 part.

```
.model IO_STD uio (
+   drvh=96.4   drvl=104
+   AtoD1="AtoD_STD"   AtoD2="AtoD_STD_NX"
+   AtoD3="AtoD_STD"   AtoD4="AtoD_STD_NX"
+   DtoA1="DtoA_STD"   DtoA2="DtoA_STD"
+   DtoA3="DtoA_STD"   DtoA4="DtoA_STD"
+   tswlh1=1.373ns     tswlh1=3.382ns
+   tswlh2=1.346ns     tswlh2=3.424ns
+   tswlh3=1.511ns     tswlh3=3.517ns
+   tswlh4=1.487ns     tswlh4=3.564ns
+ )

.subckt AtoD_STD  A D  DPWR DGND
+   params: CAPACITANCE=0
*
O0  A DGND DO74 DGTNET=D IO_STD
C1  A 0 {CAPACITANCE+0.1pF}
.ends
```

If an instance of the 74393 part is connected to an analog part via node AD_NODE, PSpice A/D generates an interface block using the I/O model specified by the digital primitive actually at the interface. Suppose that U1 is the primitive connected at AD_NODE (see the 74393 subcircuit definition on page 249), and that the IO_LEVEL is set to 1. PSpice A/D determines that IO_STD is the I/O model used by U1. Notice how IO_STD identifies the interface subcircuit names AtoD_STD and DtoA_STD to be used for level 1 subcircuit selection. If the connection with U1 is an input (such as a clock line), PSpice A/D creates an instance of the subcircuit AtoD_STD:

```
X$AD_NODE_AtoD1 AD_NODE AD_NODE$AtoD $G_DPWR
$G_DGND
+ AtoD_STD
+ PARAMS: CAPACITANCE=0
```

The DOUTPUT model parameters are described under O devices in the online *OrCAD PSpice A/D Reference Manual*.

The AtoD_STD interface subcircuit references the DO74 model in its PSpice A/D O device declaration. This model, stated elsewhere in the model libraries, describes how to translate an analog signal on the analog side of an interface node, to a digital state on the digital side of an interface node.


```
.model D074 doutput
+ s0name="X" s0vlo=0.8 s0vhi=2.0
+ s1name="0" s1vlo=-1.5 s1vhi=0.8
+ s2name="R" s2vlo=0.8 s2vhi=1.4
+ s3name="R" s3vlo=1.3 s3vhi=2.0
+ s4name="X" s4vlo=0.8 s4vhi=2.0
+ s5name="1" s5vlo=2.0 s5vhi=7.0
+ s6name="F" s6vlo=1.3 s6vhi=2.0
+ s7name="F" s7vlo=0.8 s7vhi=1.4
+
```

Supposing the output of the 74393 is connected to an analog part via the digital primitive UBUFF. At IO_LEVEL set to 1, PSpice A/D determines that the DtoA_STD interface subcircuit identified in the IO_STD model, should be used.

```
.subckt DtoA_STD D A DPWR DGND
+ params: DRVL=0 DRVH=0 CAPACITANCE=0
*
N1 A DGND DPWR DIN74 DGTNET=D IO_STD
C1 A DGND {CAPACITANCE+0.1pF}
.ends
```

For this subcircuit, the DRVH and DRVL parameters values specified in the IO_STD model would be passed to it. (The interface subcircuits in the model libraries do not currently use these values.)

The DINPUT model parameters are described under PSpice A/D N devices in the online *OrCAD PSpice A/D Reference Manual*.

The DtoA_STD interface subcircuit references the DIN74 model in its PSpice A/D N device declaration. This model, stated elsewhere in the libraries, describes how to translate a digital state into a voltage and impedance.

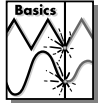
```
.model DIN74 dinput (
+ s0name="0" s0tsw=3.5ns s0rlo=7.13
+ s0rhi=389 ; 7ohm, 0.09v
+ s1name="1" s1tsw=5.5ns s1rlo=467
+ s1rhi=200 ; 140ohm, 3.5v
+ s2name="X" s2tsw=3.5ns s2rlo=42.9
+ s2rhi=116 ; 31.3ohm, 1.35v
+ s3name="R" s3tsw=3.5ns s3rlo=42.9
+ s3rhi=116 ; 31.3ohm, 1.35v
+ s4name="F" s4tsw=3.5ns s4rlo=42.9
+ s4rhi=116 ; 31.3ohm, 1.35v
+ s5name="Z" s5tsw=3.5ns s5rlo=200K
+ s5rhi=200K
+)
```

Each state is turned into a pullup and pulldown resistor pair to provide the correct voltage and impedance. The Z state is accounted for as well as the 0, 1, and X logic levels.

You can create your own interface subcircuits, DINPUT models, DOUTPUT models, and I/O models like these for technologies not currently supported in the model libraries. OrCAD recommends that you save these in your own custom model library, which you can then configure for use with a given design.

Creating a digital model using the PINDLY and LOGICEXP primitives

Note These are not supported in PSpice A/D Basics.



Unlike the majority of analog device types, the bulk of digital devices are not primitives that are compiled into the simulator. Instead, most digital models are macro models or subcircuits that are built from a few primitive devices.

These subcircuits reference interface and timing models to handle the D-to-A and A-to-D interfaces and the overall timing parameters of the physical device. For most families of digital components, the interface models are already defined and available in the DIG_IO.LIB library, which is supplied with all digital and mixed-signal packages. If you are unsure of the exact name of the interface model you need to use, use a text editor to look in DIG_IO.LIB.

For instance, if you are trying to model a 74LS component that is not already in a library, open DIG_IO.LIB with your text editor and search for 74LS to get the interface models for the 74LS family. You can also read the information at the beginning of the file which explains many of the terms and uses for the I/O models.

In the past, the timing model has presented the greatest challenge when trying to model a digital component. This was due to the delays of a component being distributed among the various gates. Recently, the ability to model digital components using logic expressions (LOGICEXP) and pin-to-pin delays (PINDLY) has been added to the simulator. Using the LOGICEXP and PINDLY digital primitives, you can describe the logic of the device with zero delay and then enter the timing parameters for the pin-to-pin delays directly from the manufacturer's data sheet. Digital primitives still must reference a standard timing model, but when the PINDLY device is used, the timing models are simply zero-delay models that are supplied in DIG_IO.LIB. The default timing models can be

found in the same manner as the standard I/O models. The PINDLY primitive also incorporates constraint checking which allows you to enter device data such as pulse width and setup/hold timing from the data sheet. Then the simulator can verify that these conditions are met during the simulation.

Digital primitives

Primitives in the simulator are devices or functions which are compiled directly into the code. The primitives serve as fundamental building blocks for more complex macro models.

There are two types of primitives in the simulator: gate level and behavioral. A gate level primitive normally refers to an actual physical device (such as buffers, AND gates, inverters). A behavioral primitive is not an actual physical device, but rather helps to define parameters of a higher level model. Just like gate level primitives, behavioral primitives are intrinsic functions in the simulator and are treated in much the same manner. They are included in the gate count for circuit size and cannot be described by any lower level model.

In our 74160 example (see *The TTL Data Book* from Texas Instruments for schematic and description), the four J-K flip-flops are the four digital gate level primitives. While flip-flops are physically more complex than gates in terms of modeling, they are defined on the same level as a gate (for example, flip-flops are a basic device in the simulator). Since all four share a common Reset, Clear, and Clock signal, they can be combined into one statement as an array of flip-flops. They could just as easily have been written separately, but the array method is more compact. See the *Digital Devices* chapter in the online *OrCAD PSpice A/D Reference Manual* for more information.

Logic expression (LOGICEXP primitive)

Looking at the listing in [74160 example on page 7-280](#) and at the schematic representation of the 74160 subcircuit, you can see that there are three main parts to the subcircuit. Following the usual header information, .SUBCKT keyword, subcircuit name, interface pin list, and parameter list is the LOGICEXP primitive. It contains everything in the component that can be expressed in terms of simple combinational logic. The logic expression device also serves to buffer other input signals that will go to the PINDLY primitive. In this case, LOGICEXP buffers the ENP_I, ENT_I, CLK_I, CLRBAR_I, LOADBAR_I, and four data signals. See the *Digital Devices* chapter in the online *OrCAD PSpice A/D Reference Manual* for more information.

For our 74160 example, the logic expression (LOGICEXP) has fourteen inputs and twenty outputs. The inputs are the nine interface input pins in the subcircuit plus five feedback signals that come from the flip-flops (QA, QB, QC, QD, and QDBAR). The flip-flops are primitive devices themselves and are not part of the logic expression. The outputs are the eight J-K data inputs to the flip-flops, RCO, the four data lines used internal to the logic expression (A, B, C, D), and the seven control lines: CLK, CLKBAR, EN, ENT, ENP, CLRBAR, and LOADBAR.

The schematic representation of the device shows buffers on every input signal of the model, while the logic diagram of the device in the data book shows buffers or inverters on only the CLRBAR_I, CLK_I, and LOADBAR_I signals. We have added buffers to the inputs to minimize the insertion of A-to-D interfaces when the device is driven by analog circuitry. The best example is the CLK signal. With the buffer in place, if the CLK signal is analog, one A-to-D interface device will be inserted into the circuit by the simulator. If the buffer was not present, then an interface device would be inserted at the CLK pin of each of the flip-flops. The buffers have no delay associated with them, but by minimizing the number of A-to-D interfaces, we speed up the mixed-signal

simulation by reducing the number of necessary calculations. For situations where the device is only connected to other digital nodes, the buffers have no effect on the simulation.

The D0_GATE, shown in the listing, is a zero-delay primitive gate timing model. For most TTL modeling applications, this only serves as a place holder and is not an active part of the model. Its function has been replaced by the PINDLY primitive. The D0_GATE model can be found in the library file DIG_IO.LIB. For a more detailed description of digital primitives, see the *Digital Devices* chapter in the online *OrCAD PSpice A/D Reference Manual*.

IO_STD, shown in the listing, is the standard I/O model. This determines the A-to-D and D-to-A interface characteristics for the subcircuit. The device contains family-specific information, but the models have been created for nearly all of the stock families. The various I/O models can be found in the library file DIG_IO.LIB.

The logic expressions themselves are straightforward. The first nine are buffering the input signals from outside the subcircuit. The rest describe the logic of the actual device up to the flip-flops. By tracing the various paths in the design, you can derive each of the logic equations.

The D0_EFF timing model, shown in the listing, is a zero-delay default model already defined in DIG_IO.LIB for use with flip-flops. All of the delays for the device are defined in the PINDLY section. The I/O model is IO_STD as identified previously. We have not specified a MNTYMXDLY or IO_LEVEL parameter, so the default values are used. For a more detailed description of the general digital primitives MNTYMXDLY and IO_LEVEL, see the *Digital Devices* chapter in the online *OrCAD PSpice A/D Reference Manual*.

The primitive MNTYMXDLY specifies whether to use the minimum, typical, maximum, or digital worst-case timing values from the device's timing model (in this case the PINDLY device). For the 74160, MNTYMXDLY is set to 0. This means that it takes on the current value of the DIGMNTYMX parameter. DIGMNTYMX defaults to 2

(typical timing) unless specifically changed using the .OPTIONS command.

The primitive IO_LEVEL selects one of four possible A-to-D and D-to-A interface subcircuits from the device's I/O model. In the header of this subcircuit, IO_LEVEL is set to 0. This means that it takes on the value of the DIGIOLVL parameter. DIGIOLVL defaults to 1 unless specifically changed using the .OPTIONS command.

Pin-to-pin delay (PINDLY primitive)

The delay and constraint specifications for the model are specified using the PINDLY primitive. The PINDLY primitive is evaluated every time any of its inputs or outputs change. See the *Digital Devices* chapter in the online *OrCAD PSpice A/D Reference Manual* for more information.

For the 74160, we have five delay paths, the four flip-flop outputs to subcircuit outputs QA...QD to QA_O...QD_O, and RCO to RCO_O. The five paths are seen in the Delay & Constraint section of the design. For delay paths, the number of inputs must equal the number of outputs. Since the 74160 does not have TRI-STATE outputs, there are no enable signals for this example, but there are ten reference nodes. The first four (CLK, LOADBAR, ENT, and CLRBAR) are used for both the pin-to-pin delay specification and the constraint checking. The last six (ENP, A, B, C, D, and EN) are used only for the constraint checking.

The PINDLY primitive also allows constraint checking of the model. It can verify the setup, hold times, pulse width, and frequency. It also has a general mechanism to allow for user-defined conditions to be reported. The constraint checking only reports timing violations; it does not affect the propagation delay or the logic state of the device. Since the timing parameters are generally specified at the pin level of the actual device, the checking is normally done at

the interface pins of the subcircuit after the appropriate buffering has been done.

BOOLEAN

The keyword **BOOLEAN** begins the boolean assignments which define temporary variables that can be used later in the **PINDLY** primitive. The form is:

boolean variable = {boolean expression}

The curly braces are required.

In the 74160 model, the boolean expressions are actually reference functions. There are three reference functions available: **CHANGED**, **CHANGED_LH**, and **CHANGED_HL**. The format is:

function name (node, delta time)

For our example, we define the variable **CLOCK** as a logical **TRUE** if there has been a LO-to-HI transition of the **CLK** signal at simulation time. We define **CONTENT** as **TRUE** if there has been any transition of the **ENT** signal at the simulation time.

Boolean operators take the following boolean values as operands:

- reference functions
- transition functions
- previously assigned boolean variables
- boolean constants **TRUE** and **FALSE**

Transition functions have the general form of:

TRN_pn

For a complete list of reference functions and transition functions, see the *Digital Devices* chapter in the online *OrCAD PSpice A/D Reference Manual*.

PINDLY

PINDLY contains the actual delay and constraint expressions for each of the outputs.

The CASE function defines a more complex, rule-based *<delay expression>* and works as a rule section mechanism for establishing path delays. Each boolean expression in the CASE function is evaluated in order until one is encountered that produces a TRUE result. Once a TRUE expression is found, the delay expression portion of the rule is associated with the output node being evaluated, and the remainder of the CASE function is ignored. If none of the expressions evaluate to TRUE, then the DEFAULT delay is used. Since it is possible for none of the expressions to yield a TRUE result, you must include a default delay in every CASE function. Also note that the expressions must be separated by a comma.

In the PINDLY section of the PINDLY primitive in the model listing, the four output nodes (QA_O through QD_O) all use the same delay rules. The CASE function is evaluated independently for each of the outputs in turn. The first delay expression is:

```
CLOCK & LOADBAR=='1 & TRN_LH, DELAY(-1,13NS,20NS)
```

This means that if CLOCK is TRUE, and LOADBAR is equal to 1, and QA_O is transitioning from 0 to 1, then the values of -1, 13ns, and 20ns are used for the MINIMUM, TYPICAL, and MAXIMUM propagation delay for the CLK-to-QA data output of the chip. In this case, the manufacturer did not supply a minimum prop delay, so we used the value -1 to tell the simulator to derive a value from what was given. If this statement is TRUE, then the simulator assigns the values and move on to the CASE function for QB_O and eventually RCO_O.

For instances where one or more propagation delay parameters are not supplied by the data sheet, the simulator derives a value from what is known and the values specified for the .OPTION DIGMNTYSCALE and DIGTYMXSCALE.

When the typical value for a delay parameter is known but the minimum is not, the simulator uses the formula:

$$TP_{xxMN} = DIGMNTYSCALE \times TP_{xxTY}$$

where the value of DIGMNTYSCALE is between 0.1 and 1.0 with the default value being 0.4. If the typical is known and the maximum is not, then the simulator uses the formula:

$$TP_{xxMX} = DIGTYMXSCALE \times TP_{xxTY}$$

where the value of DIGTYMXSCALE is greater than 1.0 with the default being 1.6. If the typical value is not known, and both the minimum and maximum are, then the typical value used by the simulator will be the average of the minimum and maximum propagation delays. If only one of min or max is known, then the typical delay is calculated using the appropriate formula as listed above. If all three are unknown, then they all default to a value of 0.

Constraint checker (CONSTRAINT primitive)

The CONSTRAINT primitive provides a general constraint checking mechanism to the digital device modeler. It performs setup and hold time checks, pulse width checks, frequency checks, and includes a general mechanism to allow user-defined conditions to be reported. See the *Digital Devices* chapter in the online *OrCAD PSpice A/D Reference Manual* for more information.

Setup_Hold

The expressions in the SETUP_HOLD specification may be listed in any order.

CLOCK defines the node that is to be used as the reference for the setup / hold / release specification. The assertion edge must be LH or HL (for example, a transition from logic state 0 to 1 or from 1 to 0.)

DATA specifies which node(s) is to have its setup / hold time measured.

SETUPTIME defines the minimum time that all DATA nodes must be stable prior to the assertion edge of the clock. The time value must be a nonnegative constant or expression and is measured in seconds. If the device has different setup / hold times depending on whether the data is HI or LOW at the clock change, you can use either or both of the following forms:

```
SETUPTIME_LO = <time value>
SETUPTIME_HI = <time value>
```

If either of the time values is 0, then no check is done for that case.

HOLDTIME is used in the same way as SETUPTIME and also has the alternate _LH and _HL formats and 0 value condition.

RELEASETIME causes the simulator to perform a special-purpose setup check. Release time (also referred to as recovery time in some data sheets) refers to the minimum time that a signal can go inactive before the active clock edge. Again, the _LH and _HL forms are available. The difference between RELEASETIME and SETUPTIME checking is that simultaneous CLOCK / DATA transitions are never allowed (this assumes a nonzero hold time). RELEASETIME is usually not used in conjunction with SETUPTIME or HOLDTIME.

Width

WIDTH does the minimum pulse-width checking. MIN_HI / MIN_LO is the minimum time that the node can remain HI/LOW. The value must be a nonnegative constant, or expression. A value of 0 means that any pulse width is allowed. At least one of MIN_HI or MIN_LO must be used within a WIDTH section.

Freq

FREQ checks the frequency. MINFREQ / MAXFREQ is the minimum / maximum frequency that is allowed on the node in question. The value must be a nonnegative floating point constant or expression measured in hertz. At least one of MINFREQ or MAXFREQ must be used within a FREQ section.

AFFECTS clauses (not used in this example) can be included in constraints to describe how the simulator should associate the failure of a constraint check with the outputs (paths through the device) of the PINDLY. This information does not affect the logic state of the outputs but provides causality detail used by the error tracking mechanism in PSpice A/D waveform analysis.

74160 example

In the 74160 example, we are checking that the maximum clock frequency (CLK) is not more than 25 MHz and the pulse width is 25 ns. We are also checking that the CLRBAR signal has a minimum LO pulse width of 20 ns, and that the 4 data inputs (A, B, C, D) have a setup / hold time of 20 ns in reference to the CLK signal. We are also checking that ENP and ENT have a setup / hold time of 20 ns with respect to the 0 to 1 transition of the CLK signal, but only when the conditions in the WHEN statement are met. All of the delay and constraint checking values were

taken directly from the actual data sheet. This makes the delay modeling both easy and accurate.

All of the above primitives and modeling methods, as well as a few special cases that are not covered here, can be found in the *Digital Devices* chapter of the online *OrCAD PSpice A/D Reference Manual*.

* 74160 Synchronous 4-bit Decade Counters with asynchronous clear

* Modeled using LOGICEXP, PINDLY, & CONSTRAINT devices

```
.SUBCKT 74160 CLK_I ENP_I ENT_I CLRBAR_I LOADBAR_I A_I B_I
C_I D_I
+ QA_0 QB_0 QC_0 QD_0 RCO_0
+ OPTIONAL: DPWR=$G_DPWR DGND=$G_DGND
+ PARAMS: MNTYMXDLY=0 IO_LEVEL=0
*
U160LOG LOGICEXP(14,20) DPWR DGND
+ CLK_I ENP_I ENT_I CLRBAR_I LOADBAR_I A_I B_I C_I D_I
+ QDBAR QA QB QC QD
+ CLK ENP ENT CLRBAR LOADBAR A B C D
+ CLKBAR RCO JA JB JC JD KA KB KC KD EN
+ DO_GATE IO_STD IO_LEVEL={IO_LEVEL}
+ LOGIC:
+ CLK = { CLK_I } ;Buffering
+ ENP = { ENP_I }
+ ENT = { ENT_I }
+ CLRBAR = { CLRBAR_I }
+ LOADBAR = { LOADBAR_I }
+ A = { A_I }
+ B = { B_I }
+ C = { C_I }
+ D = { D_I }
+ CLKBAR = { ~CLK } ;Logic expressions
+ LOAD = { ~LOADBAR }
+ EN = { ENP & ENT }
+ I1A = { LOAD | EN }
+ I2A = { ~(LOAD & A) }
+ JA = { I1A & ~(LOAD & I2A) }
+ KA = { I1A & I2A }
+ I1B = { (QA & EN & QDBAR) | LOAD }
+ I2B = { ~(LOAD & B) }
+ JB = { I1B & ~(LOAD & I2B) }
+ KB = { I1B & I2B }
+ I1C = { (QA & EN & QB) | LOAD }
+ I2C = { ~(LOAD & C) }
+ JC = { I1C & ~(LOAD & I2C) }
+ KC = { I1C & I2C }
+ I1D = { ((QC & QB & QA & EN) | (EN & QA & QD)) | LOAD }
+ I2D = { ~(LOAD & D) }
+ JD = { I1D & ~(LOAD & I2D) }
```

```

+   KD = { I1D & I2D }
+   RCO = { QD & QA & ENT }
+
UJKFF JKFF(4) DPWR DGND $D_HI CLRBAR CLKBAR JA JB JC JD KA
KB KC KD
+ QA QB QC QD QABAR QBBAR QCBAR QDBAR DO_EFF IO_STD
U160DLY PINDLY (5,0,10) DPWR DGND
+ RCO QA QB QC QD
+ CLK LOADBAR ENT CLRBAR ENP A B C D EN
+ RCO_O QA_O QB_O QC_O QD_O
+ IO_STD MNTYMXDLY={MNTYMXDLY} IO_LEVEL={IO_LEVEL}
+ BOOLEAN:
+   CLOCK = { CHANGED_LH(CLK,0) }
+   CNTENT = { CHANGED(ENT,0) }
+ PINDLY:
+   QA_O QB_O QC_O QD_O = {
+     CASE(
+       CLOCK & LOADBAR=='1' & TRN_LH, DELAY(-1,13NS,20NS),
+       CLOCK & LOADBAR=='1' & TRN_HL, DELAY(-1,15NS,23NS),
+       CLOCK & LOADBAR=='0' & TRN_LH, DELAY(-1,17NS,25NS),
+       CLOCK & LOADBAR=='0' & TRN_HL, DELAY(-1,19NS,29NS),
+       CHANGED_HL(CLRBAR,0), DELAY(-1,26NS,38NS),
+       DELAY(-1,26NS,38NS)
+     )
+   }
+   RCO_O = {
+     CASE(
+       CNTENT, DELAY(-1,11NS,16NS),
+       CLOCK, DELAY(-1,23NS,35NS),
+       DELAY(-1,23NS,35NS)
+     )
+   }
+   FREQ:
+     NODE = CLK
+     MAXFREQ = 25MEG
+   WIDTH:
+     NODE = CLK
+     MIN_LO = 25NS
+     MIN_HI = 25NS
+   WIDTH:
+     NODE = CLRBAR
+     MIN_LO = 20NS
+   SETUP_HOLD:
+     DATA(4) = A B C D
+     CLOCK LH = CLK
+     SETUPTIME = 20NS
+     WHEN = { (LOADBAR!='1' ^ CHANGED(LOADBAR,0)) &
+       CLRBAR!='0' }
+   SETUP_HOLD:
+     DATA(2) = ENP ENT
+     CLOCK LH = CLK
+     SETUPTIME = 20NS
+     WHEN = { CLRBAR!='0' & (LOADBAR!='0' ^
+       CHANGED(LOADBAR,0))
+       & CHANGED(EN,20NS) }

```

```
+ SETUP_HOLD:
+   DATA(1) = LOADBAR
+   CLOCK LH = CLK
+   SETUPTIME = 25NS
+   WHEN = { CLRBAR!=='0' }
+ SETUP_HOLD:
+   DATA(1) = CLRBAR
+   CLOCK LH = CLK
+   RELEASETIME_LH = 20NS
.ENDS
```

Part three

Setting Up and Running Analyses

Part Three describes how to set up and run analyses and provides setup information specific to each analysis type.

- [Chapter 8, Setting up analyses and starting simulation](#), explains the procedures general to all analysis types to set up and start the simulation.
- [Chapter 9, DC analyses](#), describes how to set up DC analyses, including DC sweep, bias point detail, small-signal DC transfer, and DC sensitivity.
- [Chapter 10, AC analyses](#), describes how to set up AC sweep and noise analyses.
- [Chapter 11, Transient analysis](#), describes how to set up transient analysis and optionally Fourier components. This chapter also explains how to use the Stimulus Editor to create time-based input.
- [Chapter 12, Parametric and temperature analysis](#), describes how to set up parametric and temperature analyses, and how to run post-simulation performance analysis in Probe on the results of these analyses.

-
- **Chapter 13, Monte Carlo and sensitivity/worst-case analyses**, describes how to set up Monte Carlo and sensitivity / worst-case analyses for statistical interpretation of your circuit's behavior.
 - **Chapter 14, Digital simulation**, describes how to set up a digital simulation analysis on either a digital-only or mixed-signal circuit.
 - **Chapter 15, Mixed analog/digital simulation**, explains how PSpice A / D processes the analog and digital interfaces in mixed-signal circuits.
 - **Chapter 16, Digital worst-case timing analysis**, describes how PSpice A / D performs digital worst-case timing analysis and the kinds of hazards that this analysis can help you detect.

Setting up analyses and starting simulation

8

Chapter overview

This chapter provides an overview of setting up analyses and starting simulation that applies to any analysis type. The other chapters in [Part three, Setting Up and Running Analyses](#) provide specific analysis setup information for each analysis type.

This chapter includes the following sections:

- [Analysis types on page 8-288](#)
- [Setting up analyses on page 8-289](#)
- [Starting a simulation on page 8-299](#)

Analysis types

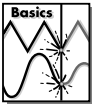
PSpice A / D supports analyses that can simulate analog-only, mixed-signal, and digital-only circuits.

PSpice A / D fully supports digital analysis by simulating the timing behavior of digital devices within a standard transient analysis, including worst-case (min / max) timing. For mixed analog / digital circuits, all of the above-mentioned analyses can be run. If the circuit is digital-only, only the transient analysis can be run.

Table 4 provides a summary of the available PSpice A / D analyses and the corresponding Analysis type options where the analysis parameters are specified. In Capture, switch to the PSpice view, then from the PSpice menu, choose New Simulation Profile .

Table 4 *Classes of PSpice A/D analyses*

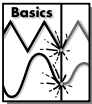
Analysis	Analysis type or Option	Swept variable
Standard analyses		
DC sweep	DC Sweep	source parameter temperature
Bias point	Bias Point	
Small-signal DC transfer	Bias Point	
DC sensitivity	Bias Point	
Frequency response	AC Sweep / Noise	frequency
Noise (requires a frequency response analysis)	AC Sweep / Noise	frequency
Transient response	Time Domain (Transient)	time
Fourier (requires transient response analysis)	Time Domain (Transient)	time
Simple multi-run analyses		
Parametric	Parametric Sweep	



Note Parametric Analysis is not supported in PSpice A/D Basics.

Table 4 *Classes of PSpice A/D analyses (continued)*

Analysis	Analysis type or Option	Swept variable
Temperature	Temperature	(Sweep)
Statistical analyses		
Monte Carlo	Monte Carlo / Worst Case	
Sensitivity / worst-case	Monte Carlo / Worst Case	



Note Monte Carlo/Worst Case Analysis is not supported in PSpice A/D Basics.

The waveform analyzer calculates and displays the results of PSpice A / D simulations for swept analyses. The waveform analyzer also generates supplementary analysis information in the form of lists and tables, and saves this in the simulation output file.

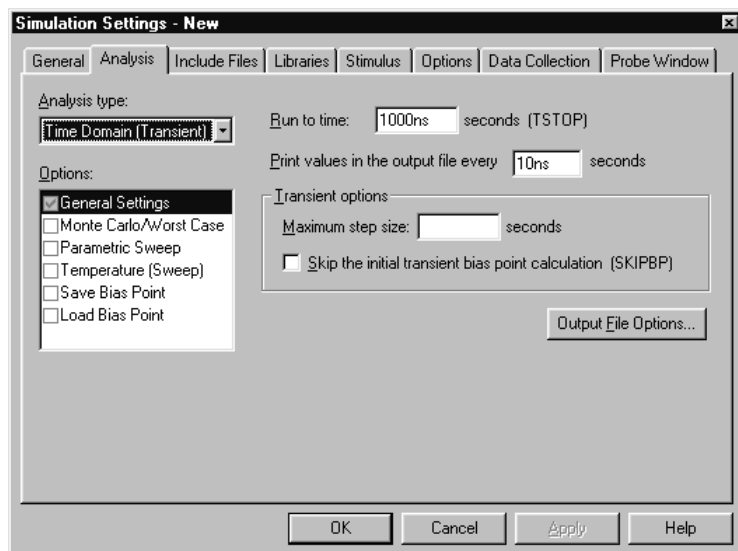
See [Part four, Viewing results](#), for information about using waveform analysis in PSpice A/D.

Setting up analyses

To set up one or more analyses

- 3 From the PSpice menu, choose New Simulation Profile.
- 4 Enter the name of the profile and click OK.
- 5 Click the Analysis tab if it is not already the active tab in the dialog box.

Specific information for setting up each type of analysis is discussed in the following chapters.



See [Output variables on page 8-292](#) for a description of the output variables that can be entered in the Simulation Settings dialog box displayed for an analysis type.

Specific information for setting up each type of analysis is discussed in the following chapters.

- 6 Enter the necessary parameter values and select the appropriate check boxes to complete the analysis specifications.
- 7 Set up any other analyses you want to perform for the circuit by selecting any of the remaining analysis types and options, then complete their setup dialog boxes.

Execution order for standard analyses

For normal simulations that are run from a simulation profile, or in batch mode, only the particular analysis type that is specified will be executed.

During simulation of a circuit file, the analysis types are performed in the order shown in [Table 5](#). Each type of analysis is conducted only once per run.

Several of the analyses (small-signal transfer, DC sensitivity, and frequency response) depend upon the bias point calculation. Because so many analyses use the bias point, PSpice A/D calculates this automatically. PSpice A/D's bias point calculation computes initial states of digital components as well as the analog components.

Table 5 *Execution order for standard analyses*

-
1. DC sweep
 2. Bias point
 3. Frequency response
 4. Noise
 5. DC sensitivity
 6. Small-signal DC transfer
 7. Transient response
 8. Fourier components
-

Output variables

Certain analyses (such as noise, Monte Carlo, sensitivity / worst-case, DC sensitivity, Fourier, and small-signal DC transfer function) require you to specify output variables for voltages and currents at specific points on the schematic. Depending upon the analysis type, you may need to specify the following:

- Voltage on a net, a pin, or at a terminal of a semiconductor device
- Current through a part or into a terminal of a semiconductor device
- A device name

If output variables or other information are required, select Output File Options in the Monte Carlo / Worst Case dialog box and enter the required parameters.

Voltage

Specify voltage in the following format:

$$v[\textit{modifiers}](\textit{<out id>}, \textit{<out id>}) \quad (1)$$

where *<out id>* is:

$$\textit{<net id>} \text{ or } \textit{<pin id>} \quad (2)$$

$$\textit{<net id>} \text{ is a fully qualified net name} \quad (3)$$

$$\textit{<pin id>} \text{ is } \textit{<fully qualified device name>:} \textit{<pin name>} \quad (4)$$

A fully qualified net name (as referred to in line 3 above) is formed by prefixing the visible net name (from a label applied to one of the segments of a wire or bus, or an offpage port connected to the net) with the full hierarchical path, separated by periods. At the top level of hierarchy, this is just the visible name.

A fully qualified device name (from line 4 above) is distinguished by specifying the full hierarchical path followed by the device's part reference, separated by period characters. For example, a resistor with part reference R34 inside part Y1 placed on a top-level

schematic page is referred to as Y1.R34 when used in an output variable.

A *<pin id>* (from line 4) is uniquely distinguished by specifying the full part name (as described above) followed by a colon, and the pin name. For example, the pins on a capacitor with reference designator C31 placed on a top-level page and pin names 1 and 2 would be identified as C31:1 and C31:2, respectively.

Current

Specify current in the following format:

i[modifiers](<out device>[:modifiers])

where *<out device>* is a fully qualified device name.

Modifiers

The basic syntax for output variables can be modified to indicate terminals of semiconductors and AC specifications. The modifiers come before *<out id>* or *<out device>*. Or, when specifying terminals (such as source or drain), the modifier is the pin name contained in *<out id>*, or is appended to *<out device>* separated by a colon.

Modifiers can be specified as follows:

- For voltage:

v[AC suffix](<out id>[, out id])

v[terminal](<out device>)*

- For current:

i[AC suffix](<out device>[:terminal])

i[terminal][AC suffix](<out device>[:])

where

terminal specifies one or two terminals for devices with more than two terminals, such as D (drain), G (gate), S (source)

- AC suffix specifies the quantity to be reported for an AC analysis, such as M (magnitude), P (phase), G (group delay)
- out id* specifies either the *<net id>* or *<pin id>* (*<fully qualified device name>:<pin name>*)
- out device* specifies the *<fully qualified device name>*

These building blocks can be used for specifying output variables as shown in [Table 6](#) (which summarizes the accepted output variable formats) and [Tables 7](#) through [10](#) (which list valid elements for two-terminal, three- or four-terminal devices, transmission line devices, and AC specifications).

Table 6 PSpice A/D output variable formats

Format	Meaning
V[ac](<i>< + out id ></i>)	voltage at <i>out id</i>
V[ac](<i>< +out id >,< - out id ></i>)	voltage across + and - <i>out id</i> 's
V[ac](<i>< 2-terminal device out id ></i>)	voltage at a <i>2-terminal device out id</i>
V[ac](<i>< 3 or 4-terminal device out id ></i>) or V< <i>x</i> >[ac](<i>< 3 or 4-terminal out device ></i>)	voltage at non-grounded terminal <i>x</i> of a <i>3 or 4-terminal device</i>
V< <i>x</i> >< <i>y</i> >[ac](<i>< 3 or 4-terminal out device ></i>)	voltage across terminals <i>x</i> and <i>y</i> of a <i>3 or 4-terminal device</i>
V[ac](<i>< transmission line out id ></i>) or V< <i>z</i> >[ac](<i>< transmission line out device ></i>)	voltage at one end <i>z</i> of a <i>transmission line device</i>

Table 6 *PSpice A/D output variable formats (continued)*

Format	Meaning
I[ac](<i>< 3 or 4-terminal out device</i> >:<x>) or I<x>[ac](<i>< 3 or 4-terminal out device</i> >)	current through non-grounded terminal x of a <i>3 or 4-terminal</i> <i>out device</i>
I[ac](<i>< transmission line out device</i> >:<z>) or I<z>[ac](<i>< 3 or 4-terminal out device</i> >)	current through one end z of a <i>transmission</i> <i>line out device</i>
< <i>DC sweep variable</i> >	voltage or current source name

Table 7 *Element definitions for 2-terminal devices*

Device type	<out id> or <out device> device indicator	Output variable examples
capacitor	C	V(CAP:1) I(CAP)
diode	D	V(D23:1) I(D23)
voltage-controlled voltage source	E	V(E14:1) I(E14)
current-controlled current source	F	V(F1:1) I(F1)
voltage-controlled current source	G	V(G2:1) I(G2)
current-controlled voltage source	H	V(HSOURCE:1) I(HSOURCE)
independent current source	I	V(IDRIV:+) I(IDRIV)
inductor	L	V(L1:1) I(L1)

Table 7 *Element definitions for 2-terminal devices*

Device type	<out id> or <out device> device indicator	Output variable examples
resistor	R	V(RC1:1) I(RC1)
voltage-controlled switch	S	V(SWITCH:+) I(SWITCH)
independent voltage source	V	V(VSRC:+) I(VSRC)
current-controlled switch	W	V(W22:-) I(W22)

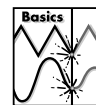
Table 8 *Element definitions for 3- or 4-terminal devices*

Device type	<out id> or <out device> device indicator	<pin id>	Output variable examples
GaAs MESFET	B	D (Drain terminal)	V(B11:D)
		G (Gate terminal)	ID(B11)
		S (Source terminal)	
Junction FET	J	D (Drain terminal)	VG(JFET)
		G (Gate terminal)	I(JFET:G)
		S (Source terminal)	

Table 8 *Element definitions for 3- or 4-terminal devices*

Device type	<out id> or <out device> device indicator	<pin id>	Output variable examples
MOSFET	M	B (Bulk, substrate terminal) D (Drain terminal) G (Gate terminal) S (Source terminal)	VDG(M1) ID(M1)
bipolar transistor	Q	B (Base terminal) C (Collector terminal) E (Emitter terminal) S (Source terminal)	V(Q1:B) I(Q1:C)
IGBT	Z	C (Collector terminal) E (Emitter terminal) G (Gate terminal)	V(Z1:C) I(Z1:C)

Note The IGBT device type is not supported in PSpice A/D Basics.

Table 9 *Element definitions for transmission line devices*

Device type	<out id> or <out device> device indicator	<z>	Output variable examples
transmission line	T	A (Port A) B (Port B)	V(T32:A+) I(T32:B-)

Table 10 *Element definitions for AC analysis specific elements*

<ac suffix> device symbol	Meaning	Output variable examples
(none)	magnitude (default)	V(V1) I(V1)
M	magnitude	VM(CAP1:1) IM(CAP1:1)
DB	magnitude in decibels	VDB(R1)
P	phase	IP(R1)
R	real part	VR(R1)
I	imaginary part	VI(R1)

The INOISE, ONOISE, DB(INOISE), and DB(ONOISE) output variables are predefined for use with noise (AC sweep) analysis.

Starting a simulation

After you have used Capture to enter your circuit design and have set up the analyses to be performed, you can start a simulation by choosing Run from the PSpice menu. When you enter and set up your circuit this way, Capture automatically generates the simulation files and starts PSpice A/D.

There may be situations, however, when you want to run PSpice A/D outside of Capture. You may want to simulate a circuit that was not created in Capture, for example, or you may want to run simulations of multiple circuits in batch mode.

This section includes the following:

- [Starting a simulation from Capture](#), below
- [Starting a simulation outside of Capture on page 8-300](#)
- [Setting up batch simulations on page 8-300](#)
- [The PSpice A/D simulation window on page 8-301](#)

Starting a simulation from Capture

After you have set up the analyses for the circuit, you can start a simulation from Capture in either of the following ways:

- From the PSpice menu select Run.
- Click the Simulate button on the PSpice toolbar.



Starting a simulation outside of Capture

To start PSpice A/D outside of Capture

- 1 From the Start menu, point to the OrCAD program group, then choose PSpice A/D.
- 2 From the File menu, choose Open Simulation.
- 3 Do one of the following:
 - Double-click on the simulation profile filename (*.SIM) in the list box.
 - Enter the simulation profile filename (*.SIM) in the File name text box and click Open.
- 4 From the Simulation menu, choose Edit Settings to modify any of the analysis setup parameters.
- 5 From the Simulation menu, choose Run (or click the Run toolbar button) to begin the simulation.

Setting up batch simulations

Multiple simulations can be run in batch mode when starting PSpice A/D directly with circuit file input. You can use batch mode, for example, to run a number of simulations overnight. There are two ways to do this, as described below.

Multiple simulation setups within one circuit file

Multiple circuit / simulation descriptions can be concatenated into a single circuit file and simulated all at once with PSpice A/D. Each circuit / simulation description in the file must begin with a title line and end with a .END statement.

The simulator reads all the circuits in the circuit file and then processes each one in sequence. The data file and simulation output file contain the outputs from each circuit in the same order as they appeared in the circuit

file. The effect is the same as if you had run each circuit separately and then concatenated all of the outputs.

Running simulations with multiple circuit files

You can direct PSpice A/D to simulate multiple circuit files using either of the following methods.

Method 1

- 1 From the Start menu, point to the OrCAD program group, then choose PSpice A/D.
- 2 Select Open Simulation from the File menu from the PSpice A/D window.
- 3 Do one of the following:
 - Type each file name in the File Name text box separated by a space.
 - Use the combination keystrokes and mouse clicks in the list box as follows: **Ctrl**+click to select file names one at a time, and **Shift**+click to select groups of files.

Method 2

- 1 From the Start menu, point to the OrCAD program group, then choose PSpice A/D.
- 2 Update the command line in the following way:
 - Include a list of circuit file names separated by spaces.

Circuit file names can be fully qualified or can contain the wild card characters (* and ?).

The PSpice A/D simulation window

The PSpice A/D Simulation Window is an MDI (Multiple Document Interface) application. This implies that you can open and display multiple files at the same time in this

window. For instance, you can have a waveform file (.DAT), a circuit file (.CIR), and a simulation output file (.OUT) open and displayed in different child windows within this one window.

The PSpice A/D Simulation Window consists of three sections: the main window section where the open files are displayed, the output window section where output information such as informational, warning, and error messages from the simulator are shown, and the simulation status window section where detailed status information about the simulation are shown. These three sections are shown in Figure 56.

The windows in these sections may be resized, moved, and reordered as needed.

The simulation window also includes a menu bar and toolbars for controlling the simulation and the waveform display.

Title bar The title bar of the simulation window (the area at the top of the window) identifies the name of the currently open simulation (either simulation profile or circuit file) and the name of the currently active document displayed in the main window area. For example, the simulation window shown in Figure 56 indicates that simulation profile Example-TRAN is currently open and the active document displayed is Example-Example-TRAN.DAT.

Menus and Toolbars The menus accessed from the menu bar include commands to set up and control the simulator, customize the window display characteristics, and configure the way the waveforms are displayed. The toolbar buttons duplicate many of the more frequently used commands.

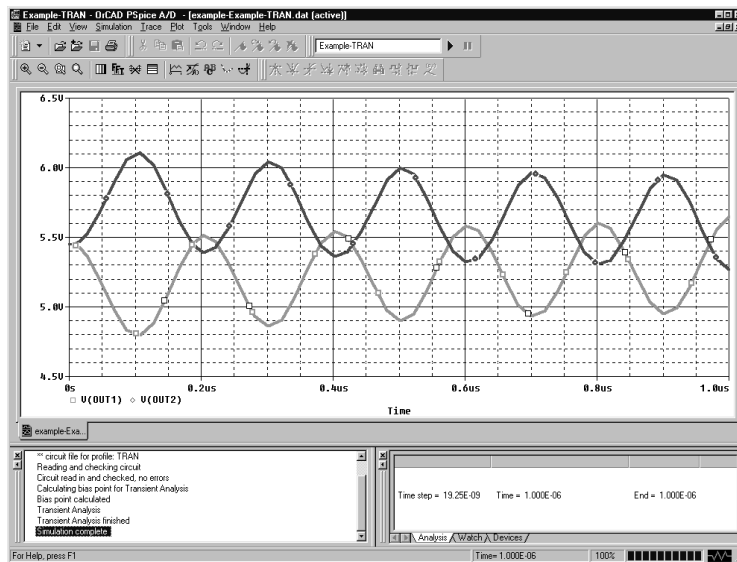


Figure 57 PSpice A/D simulation window

Main window section The top central portion (by default) of the simulation window is the main window section where documents (such as waveforms, circuit description, output information etc.) are displayed within child windows. These windows are tabbed by default. The tabs at the bottom left show the names of the documents that each child window contains. Clicking on a tab brings that child window to the foreground. Figure 56 shows the tabbed document windows for Example-Example-TRAN.DAT and Example-Example-TRAN.OUT.

You can configure the display of these windows to suit your preferences and to make the analysis of the circuit quick and readily understandable. These windows can also be resized, moved, and reordered to suit your needs.

Output window section The lower left portion of the simulation window provides a listing of the output from the simulation. It shows informational, warning, and error messages from the simulation. You can resize and relocate this window to make it easier to read.

Simulation status window section The lower right portion of the simulation window presents a set of tabbed windows that show detailed status about the simulation. There are three tabbed windows in this section: the Analysis window, the Watch Variable window, and the Devices window. The Analysis window provides a running log of values of simulation variables (parameters such as Temperature, Time Step, and Time). The Watch Variable window displays watch variables and their values. These are the variables setup to be monitored during simulation. The Devices window displays the devices that are being simulated.

DC analyses

9

Chapter overview

This chapter describes how to set up DC analyses and includes the following sections:

- [DC Sweep on page 9-306](#)
- [Bias point on page 9-315](#)
- [Small-signal DC transfer on page 9-317](#)
- [DC sensitivity on page 9-320](#)

DC Sweep

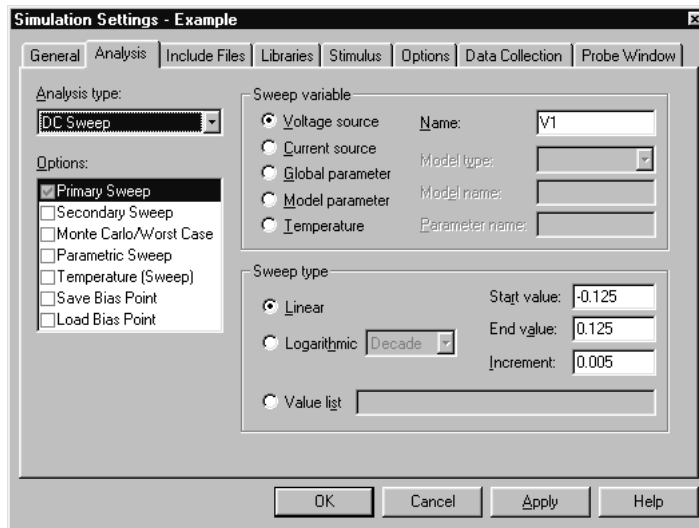
Minimum requirements to run a DC sweep analysis

Minimum circuit design requirements

Table 11 *DC sweep circuit design requirements*

Swept variable type	Requirement
voltage source	voltage source with a DC specification (VDC, for example)
temperature	none
current source	current source with a DC specification (IDC, for example)
model parameter	PSpice A / D model (.MODEL)
global parameter	global parameter defined with a parameter block (.PARAM)

Minimum program setup requirements



- 1 In Capture, select New Simulation Profile or Edit Simulation Settings from the PSpice menu. (If this is a new simulation, enter the name of the profile and click OK.)

The Simulation Settings dialog box appears.

- 2 Under Analysis type, select DC Sweep.
- 3 For the Primary Sweep option, enter the necessary parameter values and select the appropriate check boxes to complete the analysis specifications.
- 4 Click OK to save the simulation profile.
- 5 Select Run under the PSpice menu to start the simulation.

Note Do not specify a DC sweep and a parametric analysis for the same variable.

Overview of DC sweep

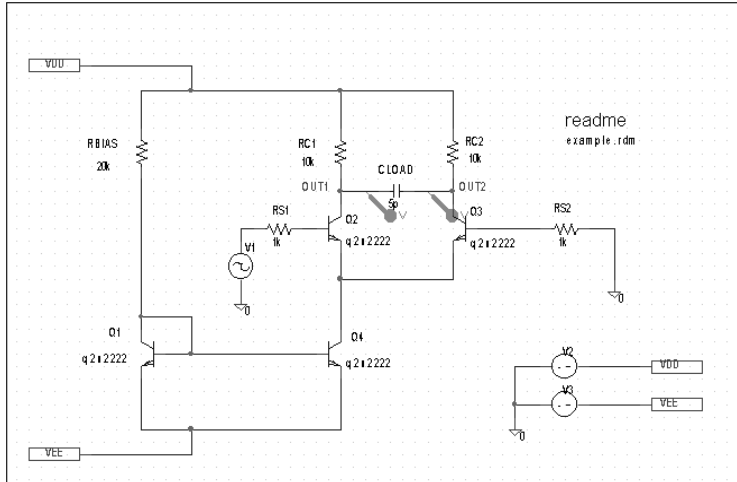
The DC sweep analysis causes a DC sweep to be performed on the circuit. DC sweep allows you to sweep a source (voltage or current), a global parameter, a model parameter, or the temperature through a range of values. The bias point of the circuit is calculated for each value of the sweep. This is useful for finding the transfer function of an amplifier, the high and low thresholds of a logic gate, and so on.

For the DC sweep analysis specified in Figure 58, the voltage source V1 is swept from -0.125 volts to 0.125 volts by steps of 0.005. This means that the output has $(0.125 - (-0.125)) / 0.005 + 1 = 51$ steps or simulation points.

A source with a DC specification (such as VDC or IDC) must be used if the swept variable is to be a voltage type or current source. To set the DC value, select Properties from the Edit menu, then click on the cell under the DC column and type in its value.

The default DC value of V1 is overridden during the DC sweep analysis and is made to be the swept value. All of the other sources retain their values.

After running the analysis, the simulation output file (EXAMPLE.OUT for the EXAMPLE.OPJ circuit in Figure 58) contains a table of voltages relating V1, node OUT1, and node OUT2.



The example circuit EXAMPLE.OPJ is provided with the OrCAD program installation.

Figure 58 Example schematic EXAMPLE.OPJ.

To calculate the DC response of an analog circuit, PSpice A/D removes time from the circuit. This is done by treating all capacitors as open circuits, all inductors as shorts, and using only the DC values of voltage and current sources. A similar approach is used for digital devices: all propagation delays are set to zero, and all stimulus generators are set to their time-zero values.

In order to solve the circuit equations, PSpice A/D uses an iterative algorithm. For analog devices, the equations are continuous, and for digital devices, the equations are Boolean. If PSpice A/D cannot get a self-consistent result after a certain number of iterations, the analog/digital devices are forced to the X value, and more iterations are done. Since X as input to a digital component gives X as output, the Boolean equations can always be solved this way.

If a digital node cannot be driven by known values during the DC iterations (for instance, the output of a flip-flop with the clock line held low), then its DC state will be X. Depending on the circuit, some, none, or all of the digital nodes may have the state X when the bias point is calculated.

Setting up a DC stimulus

To run a DC sweep or small-signal DC transfer analysis, you need to place and connect one or more independent sources and then set the DC voltage or current level for each source.

To set up a DC stimulus

- 1 Place and connect one of these symbols in your schematic:

Table 12

For voltage input	
Use this...	When you are running...
VDC	A DC Sweep and /or Bias Point (transfer function) analysis only.
VSRC	Multiple analysis types including DC Sweep and /or Bias Point (transfer function).

Table 13

For current input	
Use this...	When you are running...
IDC	A DC Sweep and /or Bias Point (transfer function) analysis only.
ISRC	Multiple analysis types including DC Sweep and /or Bias Point (transfer function).

If you are planning to run an AC or transient analysis in addition to a DC analysis, see the following:

- [Using time-based stimulus parts with AC and DC properties on page 3-118](#) for other source symbols that you can use.
- [Using VSRC or ISRC parts on page 3-119](#) to find out how to specify the TRAN attribute for a time-based input signal when using VSRC or ISRC symbols.

- 2 Double-click the symbol instance to display the Parts spreadsheet appears.
- 3 Click in the cell under the DC column to edit its value.
- 4 Define the DC specification as follows:

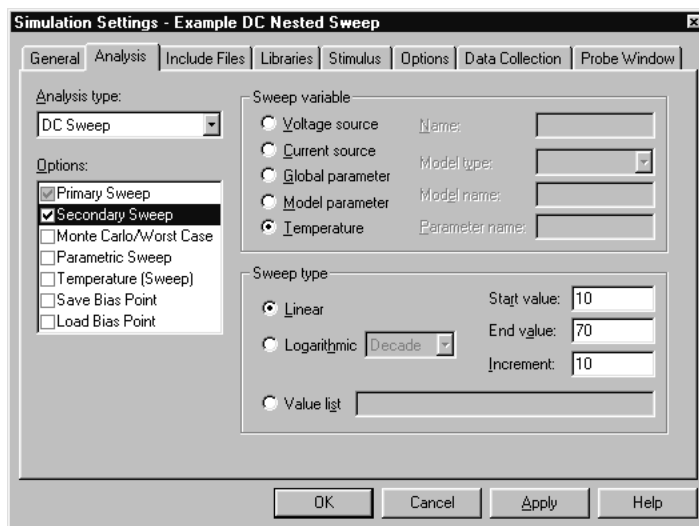
Table 14

Set this attribute...	To this value...
DC	<i>DC_level</i> where <i>DC_level</i> is in volts or amps (units are optional).

- Click OK twice to exit the dialog boxes.

Nested DC sweeps

A second sweep variable can be selected after a primary sweep value has been specified in the DC Sweep dialog box. When you specify a secondary sweep variable, it forms the outer loop for the analysis. That is, for every increment of the second sweep variable, the first sweep variable is stepped through its entire range of values.



To set up a nested sweep

- Under Options, select the Secondary Sweep box for the DC Sweep Analysis type.

- 2 Enter the necessary parameter values and select the appropriate check boxes to complete the analysis specifications.

Curve families for DC sweeps

When a nested DC sweep is performed, the entire curve family is displayed. That is, the nested DC sweep is treated as a single data section (or you can think of it as a single PSpice A/D run).

For the circuit shown in Figure 59, you can set up a DC sweep analysis with an outer sweep of the voltage source VD and an inner sweep of the voltage source VG as listed in [Table 1](#).

Table 1 Curve family example setup

	Outer sweep	Nested sweep
Swept Var Type	voltage source	voltage source
Sweep Type	linear	linear
Name	VD	VG
Start Value	0	0
End Value	5	2
Increment	0.1	0.5

When the DC sweep analysis is run, add a current marker at the drain pin of M1 and display the simulation results in PSpice A/D. The result will look like Figure 60.

To add a load line for a resistor, add a trace that computes the load line from the sweep voltage. Assume that the X axis variable is the sweep voltage V_VD, which runs from 0 to 5 volts. The expression which will add a trace that is the load line for a 50 kohm resistor is:

$$(5V - V_VD) / 50K$$

This can be useful for determining the bias point for each member of a curve family as shown in Figure 61.

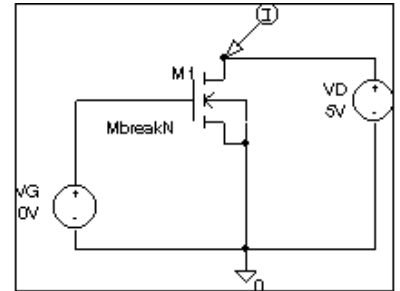


Figure 59 Curve family example schematic.

In Capture, from the PSpice menu, point to Markers, then choose Mark Current Into Pin to add a current marker.

V_VD is the hierarchical name for VD created by netlisting the schematic.

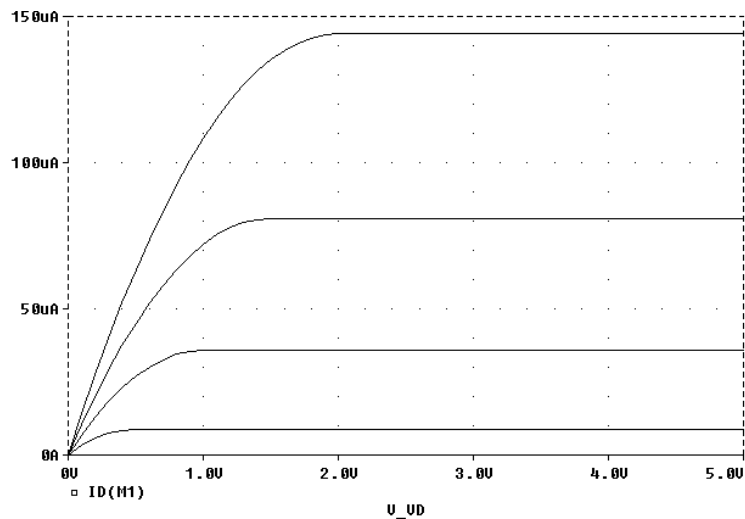


Figure 60 *Device curve family.*

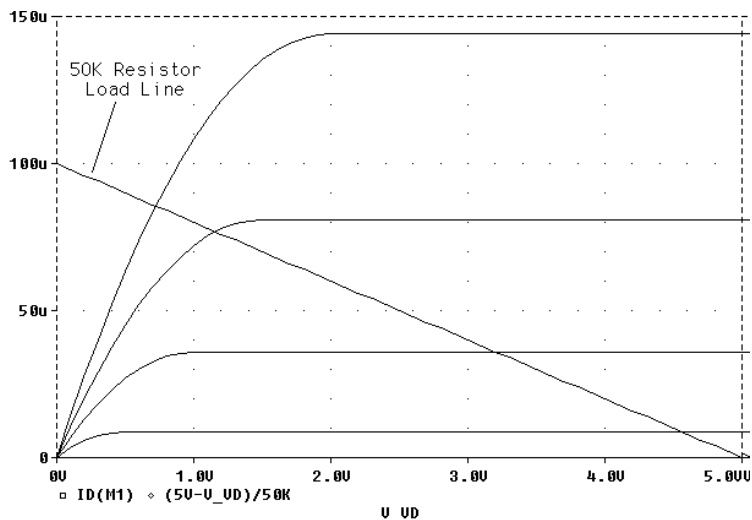


Figure 61 *Operating point determination for each member of the curve family.*

Bias point

Minimum requirements to run a bias point analysis

Minimum circuit design requirements

None.

Minimum program setup requirements

- 1 Under Analysis type in the Simulation Settings dialog box, select Bias Point.
- 2 For the General Settings option, enter the necessary parameter values and select the appropriate check boxes to complete the analysis specifications.
- 3 Click OK to save the simulation profile.
- 4 In Capture, from the PSpice menu, select Run to start the simulation.

Overview of bias point

The bias point is calculated for any analysis whether or not the Bias Point analysis is enabled in the Simulation Settings dialog box. However, additional information is reported when the Bias Point analysis is enabled.

When Bias Point analysis is not enabled, only analog node voltages and digital node states are reported to the output file.

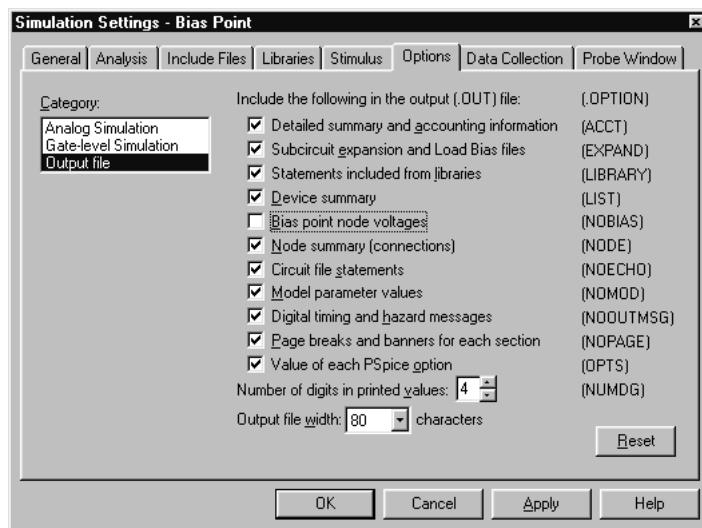
Also see [Save and load bias point on page A-542](#).

When the Bias Point analysis is enabled, the following information is reported to the output file:

- a list of all analog node voltages
- a list of all digital node states
- the currents of all voltage sources and their total power
- a list of the small-signal parameters for all devices

If Bias Point is enabled, you can suppress the reporting of the bias point analog and digital node values, as follows:

- 1 Under the Options tab in the Simulation Settings dialog box, select Output file in the Category box.
- 2 Uncheck the box for Bias point node voltages (NOBIAS).



Small-signal DC transfer

Minimum requirements to run a small-signal DC transfer analysis

Minimum circuit design requirements

- The circuit should contain an input source, such as VSRC.

Minimum program setup requirements

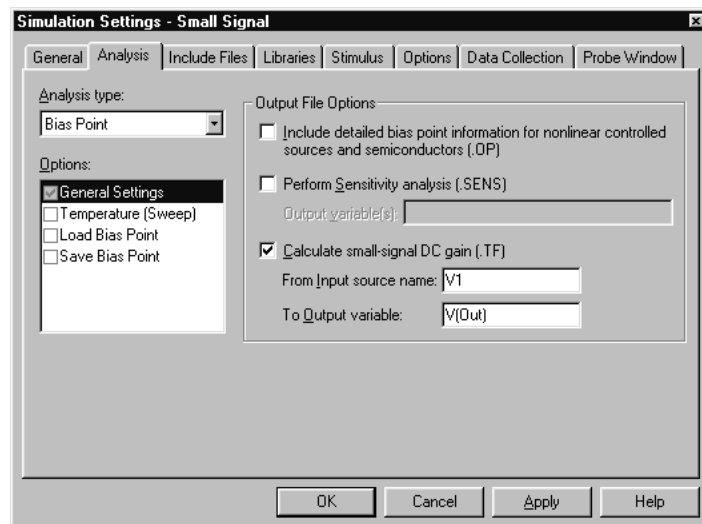
- 1 Under Analysis type in the Simulation Settings dialog box, select Bias Point.
- 2 Specify the name of the input source desired. See [Output variables on page 8-292](#) for a description of output variable formats.
- 3 Click OK to save the simulation profile.
- 4 In Capture, from the PSpice menu, select Run to start the simulation.

Overview of small-signal DC transfer

The small-signal DC transfer analysis calculates the small-signal transfer function by transforming the circuit around the bias point and treating it as a linear circuit. The small-signal gain, input resistance, and output resistance are calculated and reported.

The digital devices themselves are not included in the small-signal analysis. A gate, for example, does not have a frequency response. Instead, all the digital devices hold the states that were calculated when solving for the bias point. However, for N and O devices in the analog / digital interface subcircuits, the analog side has a well-defined linear equivalent.

To calculate the small-signal gain, input resistance, and output resistance



- 1 In the Bias Point dialog box, select Calculate small-signal DC gain (.TF).
- 2 Specify the value for either an output voltage or the current through a voltage source in the To Output variable box.

For example, entering V(a,b) as the output variable specifies that the output variable is the output voltage between two nets, a and b. Entering I(VDRIV) as the output variable specifies that the output variable is the current through a voltage source VDRIV.

- 3 Specify the input source name in the Calculate small-signal DC gain (.TF) portion of the Bias Point dialog box.

The gain from the input source to the output variable is calculated along with the input and output resistances.

For example, if you enter V(OUT2) as the output variable and V1 as the input source, the input resistance for V1 is calculated, the output resistance for V(OUT2) is calculated, and the gain from V1 to V(OUT2) is calculated. All calculations are reported to the simulation output file.

DC sensitivity

Minimum requirements to run a DC sensitivity analysis

Minimum circuit design requirements

None.

Minimum program setup requirements

- 1 In the Bias Point dialog box, select Perform Sensitivity analysis (.SENS).
- 2 Enter the required value(s) in the Output variable(s) box.
- 3 Click OK to save the simulation profile. (Be sure you give the new profile an appropriate name under the General tab prior to saving.)
- 4 In Capture, from the PSpice menu, select Run to start the simulation.

Overview of DC sensitivity

DC sensitivity analysis calculates and reports the sensitivity of one node voltage to each device parameter for the following device types:

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

The sensitivity is calculated by linearizing all devices around the bias point. Purely digital devices hold the states calculated when solving for the bias point as discussed in [Small-signal DC transfer on page 9-317](#).

AC analyses

10

Chapter overview

This chapter describes how to set up AC sweep and noise analyses.

- [AC sweep analysis on page 10-324](#) describes how to set up an analysis to calculate the frequency response of your circuit. This section also discusses how to define an AC stimulus and how PSpice A / D treats nonlinear devices in an AC sweep.
- [Noise analysis on page 10-333](#) describes how to set up an analysis to calculate device noise contributions and total input and output noise.

AC sweep analysis

Setting up and running an AC sweep

The following procedure describes the minimum setup requirements for running an AC sweep analysis. For more detail on any step, go to the pages referenced in the sidebars.

To set up and run an AC sweep

To find out how, see [Setting up an AC stimulus on page 10-325](#).

- 1 Place and connect a voltage or current source with an AC input signal.
- 2 From the PSpice menu, select New Simulation Profile or Edit Simulation Settings. (If this is a new simulation, enter the name of the profile and click OK.)

The Simulation Settings dialog box appears.

To find out how, see [Setting up an AC analysis on page 10-327](#).

- 3 Choose AC Sweep / Noise in the Analysis type list box.
- 4 Specify the required parameters for the AC sweep or noise analysis you want to run.
- 5 Click OK to save the simulation profile.
- 6 From the PSpice menu, select Run to start the simulation.

What is AC sweep?

AC sweep is a frequency response analysis. PSpice A/D calculates the small-signal response of the circuit to a combination of inputs by transforming it around the bias point and treating it as a linear circuit. Here are a few things to note:

To find out more, see [How PSpice A/D treats nonlinear devices on page 10-331](#).

- Nonlinear devices, such as voltage- or current-controlled switches, are transformed to linear

circuits about their bias point value before PSpice A/D runs the linear (small-signal) analysis.

- Digital devices hold the states that PSpice A/D calculated when solving for the bias point.
- Because AC sweep analysis is a linear analysis, it only considers the gain and phase response of the circuit; it does not limit voltages or currents.

The best way to use AC sweep analysis is to set the source magnitude to one. This way, the measured output equals the gain, relative to the input source, at that output.

Setting up an AC stimulus

To run an AC sweep analysis, you need to place and connect one or more independent sources and then set the AC magnitude and phase for each source.

To set up an AC stimulus

- 1 Place and connect one of these symbols in your schematic:

Table 2

For voltage input	
Use this...	When you are running...
VAC	An AC sweep analysis only.
VSRC	Multiple analysis types including AC sweep.

Table 3

For current input	
Use this...	When you are running...
IAC	An AC sweep analysis only.
ISRC	Multiple analysis types including AC sweep.

Note Unlike DC sweep, the AC Sweep/Noise dialog box not include an input source option. Instead, each independent source in your circuit contains its own AC specification for magnitude and phase.

If you are planning to run a DC or transient analysis in addition to an AC analysis, see [If you want to specify multiple stimulus types on page 3-118](#) for additional information and source symbols that you can use.

- 2 Double-click the symbol instance to display the Parts spreadsheet.
- 3 Click in the cell under the appropriate property column to edit its value. Depending on the source symbol that you placed, define the AC specification as follows:

Table 4

For VAC or IAC	
Set this property...	To this value...
ACMAG	AC magnitude in volts (for VAC) or amps (for IAC); units are optional.
ACPHASE	Optional AC phase in degrees.

Table 5

For VSRC or ISRC	
Set this property...	To this value...
AC	<i>Magnitude_value</i> [<i>phase_value</i>] where <i>magnitude_value</i> is in volts or amps (units are optional) and the optional <i>phase_value</i> is in degrees.

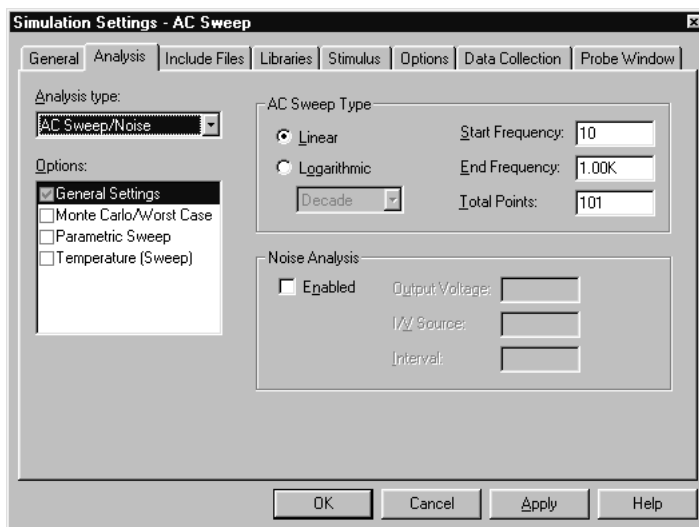
If you are also planning to run a transient analysis, see [Using VSRC or ISRC parts on page 3-119](#) to find out how to specify the TRAN property.

Setting up an AC analysis

To set up the AC analysis

- 1 From the PSpice menu, choose New Simulation Profile or Edit Simulation Settings. (If this is a new simulation, enter the name of the profile and click OK.)

The Simulation Settings dialog box appears.



- 2 Choose AC Sweep / Noise in the Analysis type list box.
- 3 Under Options, select General Settings if it is not already enabled.
- 4 Set the number of sweep points as follows:

Table 6

To sweep frequency...	Do this...
linearly	Under AC Sweep Type, click Linear, and enter the total number of points in the sweep in the Total Points box.
logarithmically by decades	Under AC Sweep Type, click Logarithmic, select Decade (default), and enter the total number of points per decade in the Total Points box.
logarithmically by octaves	Under AC Sweep Type, click Logarithmic, select Octave, and enter the total number of points per octave in the Total Points box.

If you also want to run a noise analysis, then before clicking OK, complete the Noise Analysis frame in this dialog box as described in [Setting up a noise analysis on page 10-335](#).

- 5 In the Start Frequency and End Frequency text boxes, enter the starting and ending frequencies, respectively, for the sweep.
- 6 Click OK to save the simulation profile.

AC sweep setup in example.opj

If you look at the example circuit, EXAMPLE.OPJ, provided with your OrCAD programs, you'll find that its AC analysis is set up as shown in Figure 63.

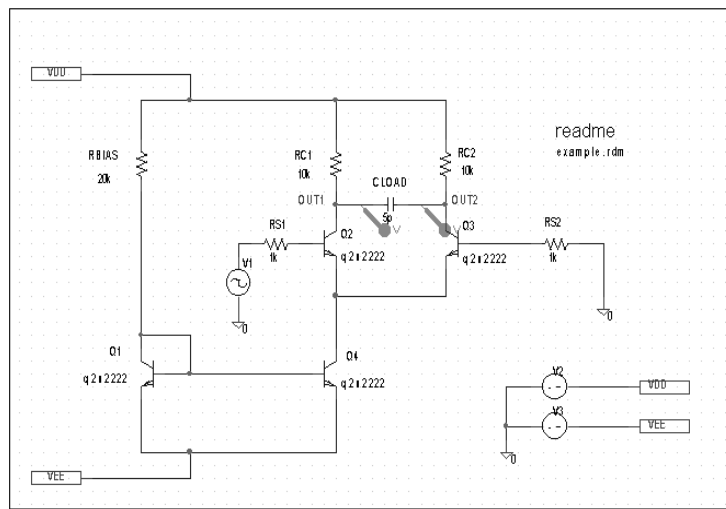


Figure 62 Circuit diagram for EXAMPLE.OPJ.

Frequency is swept from 100 kHz to 10 GHz by decades, with 10 points per decade. The V1 independent voltage source is the only input to an amplifier, so it is the only AC stimulus to this circuit. Magnitude equals 1 V and relative phase is left at zero degrees (the default). All other voltage sources have zero AC value.

Note The source, V1, is a VSIN source that is normally used for setting up sine wave signals for a transient analysis. It also has an AC property so that you can use it for an AC analysis.

To find out more about VSIN and other source symbols that you can use for AC analysis, see [Using time-based stimulus parts with AC and DC properties on page 3-118](#).

Note The source, V1, is a VSIN source that is normally used for setting up sine wave signals for a transient analysis. It also has an AC property so that you can use it for an AC analysis.

To find out more about VSIN and other source symbols that you can use for AC analysis, see [Using time-based stimulus parts with AC and DC properties on page 3-118](#).

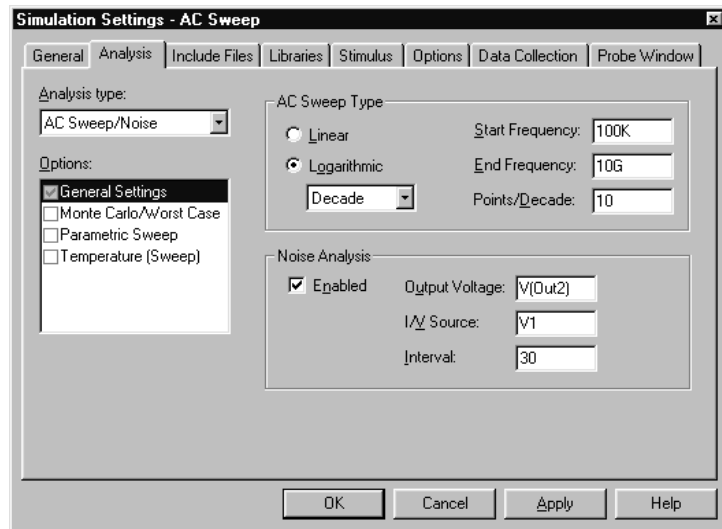


Figure 63 AC analysis setup for EXAMPLE.OPJ.

Frequency is swept from 100 kHz to 10 GHz by decades, with 10 points per decade. The V1 independent voltage source is the only input to an amplifier, so it is the only AC stimulus to this circuit. Magnitude equals 1 V and relative phase is left at zero degrees (the default). All other voltage sources have zero AC value.

How PSpice A/D treats nonlinear devices

An AC Sweep analysis is a linear or small-signal analysis. This means that nonlinear devices must be linearized to run the analysis.

What's required to transform a device into a linear circuit

In order to transform a device (such as a transistor amplifier) into a linear circuit, you must do the following:

- 1 Compute the DC bias point for the circuit.
- 2 Compute the complex impedance and/or transconductance values for each device at this bias point.
- 3 Perform the linear circuit analysis at the frequencies of interest by using simplifying approximations.

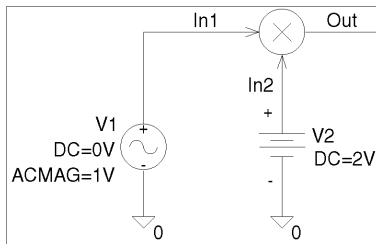
Example: Replace a bipolar transistor in common-emitter mode with a constant transconductance (collector current proportional to base-emitter voltage) and a number of constant impedances.

What PSpice A/D does

PSpice A/D automates this process for you. PSpice A/D computes the partial derivatives for nonlinear devices at the bias point and uses these to perform small-signal analysis.

Example: nonlinear behavioral modeling block

Suppose you have an analog behavioral modeling block that multiplies $V(1)$ by $V(2)$. Multiplication is a nonlinear operation. To run an AC sweep analysis on this block, the block needs to be replaced with its linear equivalent. To determine the linear equivalent block, PSpice A/D needs a known bias point.



Using a DC source

Consider the circuit shown here. At the DC bias point, PSpice A/D calculates the partial derivatives which determine the linear response of the multiplier as follows:

$$\begin{aligned} V(Out) &= V(In1) \cdot \frac{\partial V(Out)}{\partial V(In1)} + V(In2) \cdot \frac{\partial V(Out)}{\partial V(In2)} \\ &= V(In1) \cdot V(In2) + V(In2) \cdot V(In1) \end{aligned}$$

For this circuit, this equation reduces to:

$$V(Out) = V(In1) \cdot 2 + V(In2) \cdot 0$$

This means that the multiplier acts as an amplifier of the AC input with a gain that is set by the DC input.



Caution: multiplying AC sources

Suppose that you replace the 2 volt DC source in this example with an AC source with amplitude 1 and no DC value (DC=0). When PSpice A/D computes the bias point, there are no DC sources in the circuit, so all nodes are at 0 volts at the bias point. The linear equivalent of the multiplier block is a block with gain 0, which means that there is no output voltage at the fundamental frequency.

This is exactly how a double-balanced mixer behaves. In practice, this is a simple multiplier.

Note *A double-balanced mixer with inputs at the same frequency would produce outputs at DC at twice the input frequency, but these terms cannot be seen with a linear, small-signal analysis.*

Noise analysis

Setting up and running a noise analysis

The following procedure describes the minimum setup requirements for running a noise analysis. For more detail on any step, go to the pages referenced in the sidebars.

To set up and run an AC sweep

- 1 Place and connect a voltage or current source with an AC input signal.
- 2 Set up the AC sweep simulation specifications.
- 3 Set up the noise simulation specifications and enable the analysis in the AC Sweep / Noise portion of the Simulation Settings dialog box.
- 4 Click OK to save the simulation profile.
- 5 From the PSpice menu, choose Run to start the simulation.

To find out how, see [Setting up an AC stimulus on page 10-325](#).

To find out how, see [Setting up an AC analysis on page 10-327](#).

To find out how, see [Setting up a noise analysis on page 10-335](#).

What is noise analysis?

When running a noise analysis, PSpice A/D calculates and reports the following for each frequency specified for the AC Sweep / Noise analysis:

- Device noise, which is the noise contribution propagated to the specified output net from every resistor and semiconductor device in the circuit; for semiconductor devices, the device noise is also broken down into constituent noise contributions where applicable
- Total output and equivalent input noise

Example: Diodes have separate noise contributions from thermal, shot, and flicker noise.

Table 7

This value...	Means this...
Output noise	RMS sum of all the device contributions propagated to a specified output net
Input noise	equivalent noise that would be needed at the input source to generate the calculated output noise in an ideal (noiseless) circuit

How PSpice A/D calculates total output and input noise

To calculate total noise at an output net, PSpice A/D computes the RMS sum of the noise propagated to the net by all noise-generating devices in the circuit.

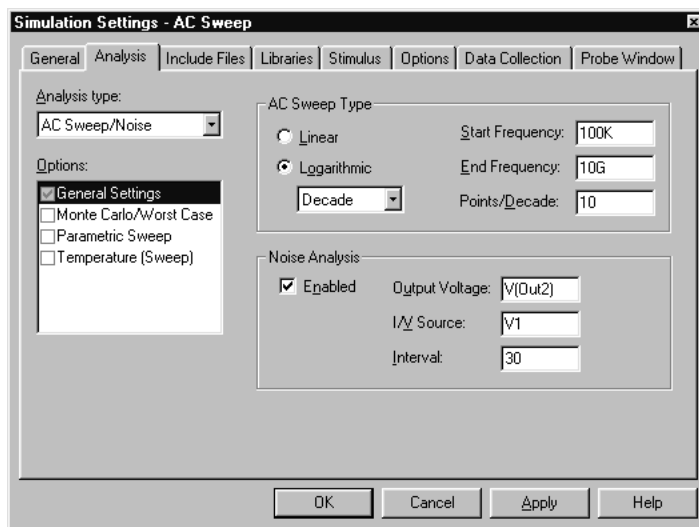
To calculate the equivalent input noise, PSpice A/D then divides total output noise by the gain from the input source to the output net. This results in the amount of noise which, if injected at the input source into a noiseless circuit, would produce the total noise originally calculated for the output net.

Setting up a noise analysis

To set up the noise analysis

- 1 From the PSpice menu, choose New Simulation Profile or Edit Simulation Settings. (If this is a new simulation, enter the name of the profile and click OK.)

The Simulation Settings dialog box appears.



- 2 Choose AC Sweep / Noise in the Analysis type list box.
- 3 Under Options, select General Settings if it is not already enabled.
- 4 Specify the AC sweep analysis parameters as described on page [10-327](#).
- 5 Enable the Noise Analysis check box.
- 6 Enter the noise analysis parameters as follows:

To find out more about valid syntax, see [Output variables on page 8-292](#).

Example: U1.V2

Note *In the Probe window, you can view the device noise contributions at every frequency specified in the AC sweep. The Interval parameter has no effect on what PSpice A/D writes to the Probe data file.*

Table 8

In this text box...	Type this...
Output Voltage	A voltage output variable of the form $V(\text{node}, [\text{node}])$ where you want the total output noise calculated.
I / V Source	The name of an independent current or voltage source where you want the equivalent input noise calculated. Note <i>If the source is in a lower level of a hierarchical schematic, separate the names of the hierarchical devices with periods (.).</i>
Interval	An integer n designating that at every n^{th} frequency, you want to see a table printed in the PSpice output file (.out) showing the individual contributions of all of the circuit's noise generators to the total noise.

7 Click OK to save the simulation profile.

Analyzing Noise in the Probe window

You can use these output variable formats to view traces for device noise contributions and total input or output noise at every frequency in the analysis.

For a break down of noise output variables by supported device type, see [Table 11 on page 17-526](#).

To view this...	Use this output variable...	Which is represented by this equation*...
Flicker noise for a device	NFID(device_name) NFIB(device_name)	$\text{noise} \propto k_f \cdot \frac{I_f^{a_f}}{f^b}$
Shot noise for a device	NSID(device_name) NSIB(device_name) NSIC(device_name)	For diodes and BJTs: $\text{noise} \propto 2qI$ For GaAsFETs, JFETs, and MOSFETs: $\text{noise} \propto 4kT \cdot \frac{dI}{dV} \cdot \frac{2}{3}$
Thermal noise for the RB, RC, RD, RE, RG, or RS constituent of a device, respectively	NRB(device_name) NRC(device_name) NRD(device_name) NRE(device_name) NRG(device_name) NRS(device_name)	$\text{noise} \propto \frac{4kT}{R}$
Thermal noise generated by equivalent resistances in the output of a digital device	NRLO(device_name) NRHI(device_name)	$\text{noise} \propto \frac{4kT}{R}$
Total noise for a device	NTOT(device_name)	Sum of all contributors in device_name
Total output noise for the circuit	NTOT(ONoise)	$\sum_{\text{device}} NTOT(\text{device})$
RMS-summed output noise for the circuit	V(ONoise)	RMS sum of all contributors ($\sqrt{NTOT(ONoise)}$)
Equivalent input noise for the circuit	V(INoise)	$\frac{V(ONoise)}{\text{gain}}$

* To find out more about the equations that describe noise behavior, refer to the appropriate device type in the *Analog Devices* chapter in the *OrCAD PSpice Reference Manual*.

About noise units

Table 9

This type of noise output variable...	Is reported in these units...
Device contribution of the form Nxxx	$(volts)^2/(Hz)$
Total input or output noise of the form V(ONOISE) or V(INOISE)	$(volts)/(\sqrt{Hz})$

Example

You can run a noise analysis on the circuit shown in Figure 62 on page 10-329.

To run a noise analysis on the example:

In Capture, open the EXAMPLE.DSN circuit provided with your OrCAD programs in the ORCAD\CAPTURE\SAMPLES subdirectory.

- 1 From the PSpice menu, choose New Simulation Profile or Edit Simulation Settings. (If this is a new simulation, enter the name of the profile and click OK.)

The Simulation Settings dialog box appears.
- 2 Choose AC Sweep / Noise in the Analysis type list box.
- 3 Under Options, select General Settings if it is not already enabled.
- 4 Enable the Noise Analysis check box.
- 5 Enter the following parameters for the noise analysis:

Output Voltage	V(OUT2)
I / V Source	V1
Interval	30

For a description of the Interval parameter, see page [10-336](#).

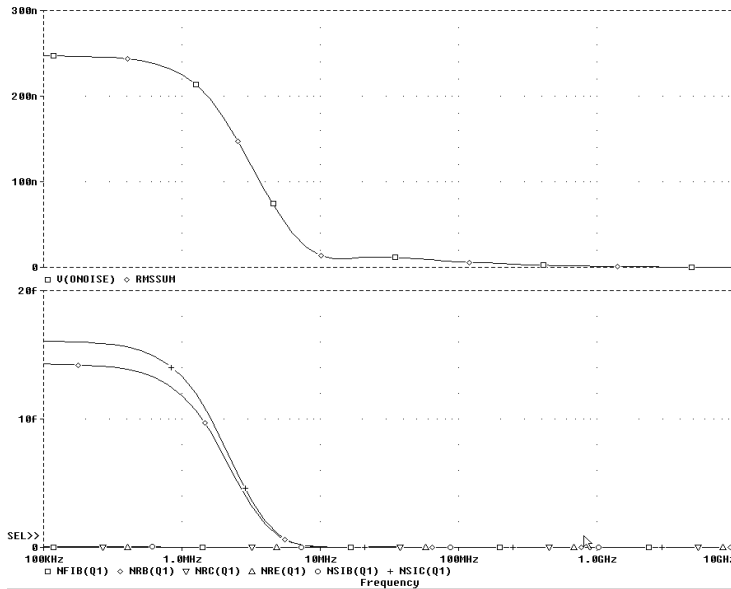
These settings mean that PSpice A / D will calculate noise contributions and total output noise at net OUT2 and equivalent input noise from V1.

Figure 63 shows Probe traces for Q1's constituent noise sources as well as total noise for the circuit after simulating. Notice that the trace for RMSSUM (at the top of the plot), which is a macro for the trace expression

$$\text{SQRT}(\text{NTOT}(\text{Q1}) + \text{NTOT}(\text{Q2}) + \text{NTOT}(\text{Q3}) + \dots),$$

exactly matches the total output noise, V(ONoise), calculated by PSpice A/D.

To find out more about PSpice macros, refer to PSpice A/D online Help.



Note The source, V1, is a VSIN source that is normally used for setting up sine wave signals for a transient analysis. It also has an AC property so that you can use it for an AC analysis.

To find out more about VSIN and other source symbols that you can use for AC analysis, see [Using time-based stimulus parts with AC and DC properties on page 3-118](#).

Figure 64 Device and total noise traces for EXAMPLE.DSN.

Frequency is swept from 100 kHz to 10 GHz by decades, with 10 points per decade. The V1 independent voltage source is the only input to an amplifier, so it is the only AC stimulus to this circuit. Magnitude equals 1 V and relative phase is left at zero degrees (the default). All other voltage sources have zero AC value.

Transient analysis

11

Chapter overview

This chapter describes how to set up a transient analysis and includes the following sections:

- [Overview of transient analysis on page 11-342](#)
- [Defining a time-based stimulus on page 11-344](#)
- [Transient \(time\) response on page 11-356](#)
- [Internal time steps in transient analyses on page 11-358](#)
- [Switching circuits in transient analyses on page 11-359](#)
- [Plotting hysteresis curves on page 11-359](#)
- [Fourier components on page 11-361](#)

Overview of transient analysis

Minimum requirements to run a transient analysis

Minimum circuit design requirements

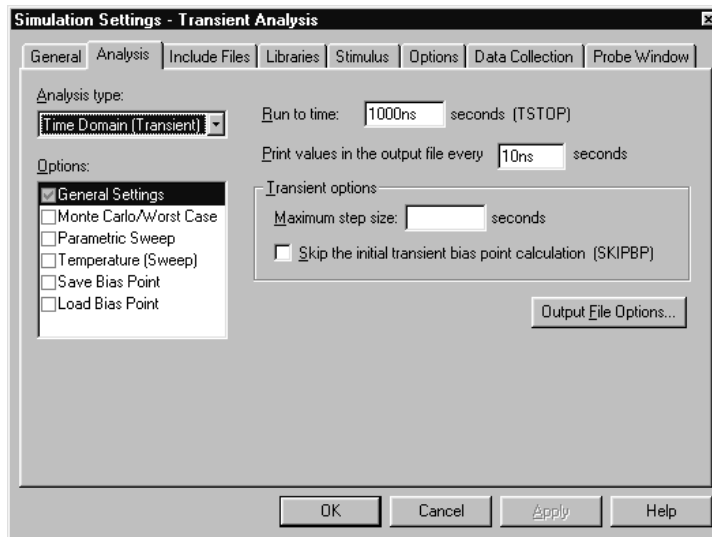
Circuit should contain one of the following:

- An independent source with a transient specification (see [Table 10](#))
- An initial condition on a reactive element
- A controlled source that is a function of time

Minimum program setup requirements

See [Setting up analyses on page 8-289](#) for a description of the Analysis Setup dialog box.

- 1 From the PSpice menu, choose New Simulation Profile or Edit Simulation Settings. (If this is a new simulation, enter the name of the profile and click OK.)
The Simulation Settings dialog box appears.
- 2 From the Analysis type list box, select Time Domain (Transient).
- 3 Specify the required parameters for the transient analysis you want to run.
- 4 Click OK to save the simulation profile.
- 5 From the PSpice menu, choose Run to start the simulation.



Defining a time-based stimulus

Overview of stimulus generation

Symbols that generate input signals for your circuit can be divided into two categories:

- those whose transient behavior is characterized graphically using the Stimulus Editor
- those whose transient behavior is characterized by manually defining their properties within Capture

Their symbols are summarized in [Table 10](#).

Table 10 Stimulus symbols for time-based input signals

Specified by...	Symbol name	Description
Using the Stimulus Editor	VSTIM	voltage source
	ISTIM	current source
	DIGSTIM1	digital stimuli
	DIGSTIM2	
	DIGSTIM4	
	DIGSTIM8	
	DIGSTIM16	
	DIGSTIM32	
Defining symbol attribute	VSRC	voltage sources
	VEXP	
	VPULSE	
	VPWL	
	VPWL_RE_FOREVER	
	VPWL_F_RE_FOREVER	
	VPWL_N_TIMES	
	VPWL_F_N_TIMES	
	VSFFM	
	VSIN	

Note The Stimulus Editor is not included in PSpice A/D Basics.

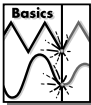


Table 10 *Stimulus symbols for time-based input signals*

Specified by...	Symbol name	Description
	ISRC	current sources
	IEXP	
	IPULSE	
	IPWL	
	IPWL_RE_FOREVER	
	IPWL_F_RE_FOREVER	
	IPWL_N_TIMES	
	IPWL_F_N_TIMES	
	ISFFM	
	ISIN	digital clock signal
	DIGCLOCK	
	STIM1	digital stimuli
	STIM4	
	STIM8	
	STIM16	
	FILESTIM1	digital file stimuli
	FILESTIM2	
	FILESTIM4	
	FILESTIM8	
	FILESTIM16	
	FILESTIM32	

To use any of these source types, you must place the symbol in your schematic and then define its transient behavior.

Each property-characterized stimulus has a distinct set of attributes depending upon the kind of transient behavior it represents. For VPWL_F_XXX, IPWL_F_XXX, and FSTIM, a separate file contains the stimulus specification.

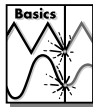
As an alternative, the Stimulus Editor utility automates the process of defining the transient behavior of stimulus devices. The Stimulus Editor allows you to create analog stimuli which generate sine wave, repeating pulse, exponential pulse, single-frequency FM, and piecewise linear waveforms. It also facilitates creating digital stimuli with complex timing relations. This applies to both stimulus symbols placed in your schematic as well as new ones that you might create.

For information on digital stimuli characterized by property, see [Chapter 14, Digital simulation](#).

The stimulus specification created using the Stimulus Editor is saved to a file, automatically configured into the schematic, and associated with the corresponding VSTIM, ISTIM, or DIGSTIM part instance or symbol definition.

The Stimulus Editor utility

Note *The Stimulus Editor is not included in PSpice A/D Basics.*



OrCAD program versions without the Stimulus Editor must use the characterized-by-property sources listed in [Table 10 on page 11-344](#).

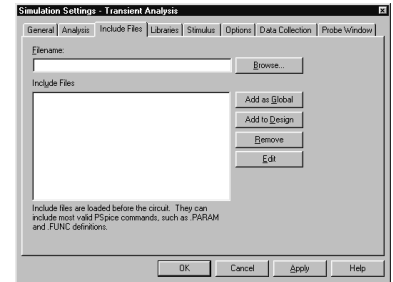
The Stimulus Editor is a utility that allows you to quickly set up and verify the input waveforms for a transient analysis. You can create and edit voltage sources, current sources, and digital stimuli for your circuit. Menu prompts guide you to provide the necessary parameters, such as the rise time, fall time, and period of an analog repeating pulse, or the complex timing relations with repeating segments of a digital stimulus. Graphical feedback allows you quickly verify the waveform.

Stimulus files

The Stimulus Editor produces a file containing the stimuli with their transient specification. These stimuli are defined as simulator device declarations using the V (voltage source), I (current source), and U STIM (digital stimulus generator) forms. Since the Stimulus Editor produces these statements automatically, you will never have to be concerned with their syntax. However, if you are interested in a detailed description of their syntax, see the descriptions of V and I devices in the *Analog Devices* chapter and stimulus generator in the *Digital Devices* chapter of the the online *OrCAD PSpice A/D Reference Manual*.

Configuring stimulus files

The Include Files tab in the Simulation Settings dialog box allows you to view the list of stimulus files pertaining to your current schematic. You can also manually add, delete, or change the stimulus file configuration in this tab dialog box. The list box displays all of the currently configured stimulus files. One file is specified per line. Files can be configured as either global to the Capture environment or local to the current design. Global files are marked with an asterisk (*) after the file name.



When starting the Stimulus Editor from Capture, stimulus files are automatically configured (added to the list) as local to the current design. Otherwise, new stimulus files can be added to the list by entering the file name in the Filename text box and then clicking the Add to design (local configuration) or Add as global (global configuration) button.

Starting the Stimulus Editor

The Stimulus Editor is fully integrated with Capture and can be run from either the schematic editor or symbol editor.

You can start the Stimulus Editor by doing the following:

- 6 Select one or more stimulus instances in the schematic
- 7 From the Edit menu, choose PSpice Stimulus.

When you first start the Stimulus Editor, you may need to adjust the scale settings to fit the trace you are going to add. You can use Axis Settings on the Plot menu or the corresponding toolbar button to change the displayed data, the extent of the scrolling region, and the minimum resolution for each of the axes. Displayed Data Range parameters determine what portion of the stimulus data set will be presented on the screen. Extent of Scrolling Region parameters set the absolute limits on the viewable range. Minimum Resolution parameters determine the

smallest usable increment (example: if it is set to 1 msec, then you cannot add a data point at 1.5 msec).

Defining stimuli

See [Chapter 14, Digital simulation](#), for detailed information about creating digital stimuli.

- 1 Place stimulus part instances from the symbol set: VSTIM, ISTIM and DIGSTIMn.
- 2 Click the source instance to select it.
- 3 From the Edit menu, choose PSpice Stimulus to start the Stimulus Editor.
- 4 Fill in the transient specification according to the dialogs and prompts.

Piecewise linear and digital stimuli can be specified by direct manipulation of the input waveform display.
- 5 From the File menu, choose Save to save the edits.

Example: piecewise linear stimulus

- 1 Open an existing schematic or start a new one.
- 2 From the Place menu, choose Part and browse the SOURCE.OLB part library file for VSTIM (and select it).
- 3 Place the part. It looks like a regular voltage source with an implementation property displayed.
- 4 Click the implementation label and type Vfirst. This names the stimulus that you are going to create.
- 5 If you are working in a new schematic, use Save from the File menu to save it. This is necessary since the schematic name is used to create the default stimulus file name.
- 6 Click the VSTIM part to select it.
- 7 From the Edit menu, choose PSpice Stimulus. This starts the Stimulus Editor and displays the New Stimulus dialog box. You can see that the stimulus already has the name of Vfirst.
- 8 Select PWL in the dialog box and click OK. The cursor looks like a pencil. The message in the status bar at the bottom of the screen lets you know that you are in the process of adding new data points to the stimulus. The

left end of the bottom status bar displays the current coordinates of the cursor.

- 9 Move the cursor to (200ns, 1) and click the left mouse button. This adds the point. Notice that there is automatically a point at (0,0). Ignore it for now and continue to add a couple more points to the right of the current one.

- 10 Click-right to stop adding points.

- 11 From the File menu, choose Save.

If you make a mistake or want to make any changes, reshape the trace by dragging any of the handles to a new location. The dragged handle cannot pass any other defined data point.

To delete a point, click its handle and press **[Del]**.

To add additional points, either choose Add Point from the Edit menu, press **[Alt]+[A]**, or click the Add Point toolbar button.

At this point you can return to Capture, edit the current stimulus, or go on to create another.

This example creates a 10 k sine wave with the amplitude parameterized so that it can be swept during a simulation.

Example: sine wave sweep

- 1 Open an existing schematic or start a new one.
- 2 Place a VSTIM part on your schematic.
- 3 To name the stimulus, double-click the implementation property and type Vsin.
- 4 Click the VSTIM part to select it.
- 5 From the PSpice menu, choose Edit Stimulus to start the Stimulus Editor.
- 6 Define the stimulus parameter for amplitude:
 - a From the New Stimulus dialog box, choose Cancel.
 - b From the Tools menu, choose Parameters.
 - c Enter AMP=1 in the Definition text box, and click OK.

- d From the Stimulus menu, choose New or click the New Stimulus button in the toolbar.
 - e Give the stimulus the name of Vsin.
 - f Select SIN as the type of stimulus to be created, and click OK.
- 7 Define the other stimulus properties:
 - a Enter 0 for Offset Value.
 - b Enter {AMP} for Amplitude. The curly braces are required. They indicate that the expression needs to be evaluated at simulation time.
 - c Enter 10k for Frequency and click OK.
 - d From the File menu, choose Save.
- 8 Within Capture, place and define the PARAM symbol:
 - a From the Place menu, choose Part.
 - b Either browse SPECIAL.OLB for the PARAM part or type in the name.
 - c Place the part on your schematic and double-click it.
 - d Click New to add a new user property.
 - e Set the value property name to AMP (no curly braces).
 - f Set the value of the VALUE1 property to 1.
- 9 Set up the parametric sweep and other analyses:
 - a From the PSpice menu, choose Stimulus Editor, and click the Parametric Sweep button.
 - b Select Global Parameter in the Swept Var. Type frame.
 - c Select Linear in the Sweep type frame.
- 10 Enter AMP in the Name text box.
- 11 Specify values for the Start Value, End Value, and Increment text boxes.

You can now set up your usual Transient, AC, or DC analysis and run the simulation.

Creating new stimulus symbols

- 1 Use the Capture part editor to edit or create a part with the following properties:

Implementation Type	PSpice Stimulus
Implementation	name of the stimulus model
STIMTYPE	type of stimulus; valid values are ANALOG or DIGITAL; if this property is nonexistent, the stimulus is assumed to be ANALOG

Editing a stimulus

To edit an existing stimulus

- 1 Start the Stimulus Editor and select Get from the Stimulus menu.
- 2 Double-click the trace name (at the bottom of the X axis for analog and to the left of the Y axis for digital traces.) This opens the Stimulus Attributes dialog box where you can modify the attributes of the stimulus directly and immediately see the effect of the changes.

To edit a PWL stimulus

- 1 Double click the trace name. This displays the handles for each defined data point.
- 2 Click any handle to select it. To reshape the trace, drag it to a new location. To delete the data point, press **Del**.
- 3 To add additional data points, either select Add from the Edit menu or click the Add Point button.
- 4 Right-click to end adding new points.

PWL stimuli are a little different since they are a series of time/value pairs.

To select a time and value scale factor for PWL stimuli

- 1 Select the PWL trace by clicking on its name.
- 2 Select Attributes from the Edit menu or click the corresponding toolbar button.

This provides a fast way to scale a PWL stimulus.

Deleting and removing traces

To delete a trace from the displayed screen, select the trace name by clicking on its name, then press **[Del]**. This will only erase the display of the trace, not delete it from your file. The trace is still available by selecting Get from the Stimulus menu.

To remove a trace from a file, select Remove from the Stimulus menu.

Note *Once a trace is removed, it is no longer retrievable. Delete traces with caution.*

Manual stimulus configuration

Stimuli can be characterized by manually starting the Stimulus Editor and saving their specifications to a file. These stimulus specifications can then be associated to stimulus instances in your schematic or to stimulus symbols in the symbol library.

To manually configure a stimulus

- 1 Start the Stimulus Editor by double-clicking on the Stimulus Editor icon in the OrCAD program group.
- 2 Open a stimulus file by selecting Open from the File menu. If the file is not found in your current library search path, you are prompted for a new file name.
- 3 Create one or more stimuli to be used in your schematic. For each stimulus:
 - a Name it whatever you want. This name will be used to associate the stimulus specification to the stimulus instance in your schematic, or to the symbol in the symbol library.
 - b Provide the transient specification.
 - c From the File menu, choose Save.

- 4 In the schematic page editor, configure the Stimulus Editor's output file into your schematic:
 - a From the Pspice menu, choose Edit Simulation Settings.
 - a In the Simulation Settings dialog box, select the Include Files tab.
 - b Enter the file name specified in step 2.
 - c If the stimulus specifications are for local use in the current design, click the Add to design button. For global use by any design, use Add as global instead.
 - d Click OK.
- 5 Modify either the stimulus instances in the schematic or symbols in the symbol library to reference the new stimulus specification.
- 6 Associate the transient stimulus specification to a stimulus instance:
 - a Place a stimulus part in your schematic from the part set: VSTIM, ISTIM, and DIGSTIMn.
 - b Click the VSTIM, ISTIM, or DIGSTIMn instance.
 - c From the Edit menu, choose Properties.
 - d Click the Implementation cell, type in the name of the stimulus, and click Apply.
 - e Complete specification of any VSTIM or ISTIM instances by selecting Properties from the Edit menu and editing their DC and AC attributes.

Click the DC cell and type its value.

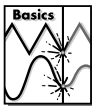
Click the AC cell, type its value, and then click Apply.
 - f Close the property editor spreadsheet.
- 7 To change stimulus references globally for a part:
 - a Select the part you want to edit.
 - a From the Edit menu, choose Part to start the part editor.

See [Chapter 5, Creating parts for models](#), for a description of how to create and edit parts.

- b Create or change the part definition, making sure to define the following properties:
Implementation stimulus name as defined in the Stimulus Editor

Transient (time) response

Note Transient (time) response analysis is not supported in PSpice A/D Basics.



The Transient response analysis causes the response of the circuit to be calculated from TIME = 0 to a specified time. A transient analysis specification is shown for the circuit EXAMPLE.OPJ in Figure 65. (EXAMPLE.OPJ is shown in Figure 66.)

The analysis is to span the time interval from 0 to 1000 nanoseconds and values should be reported to the simulation output file every 20 nanoseconds.

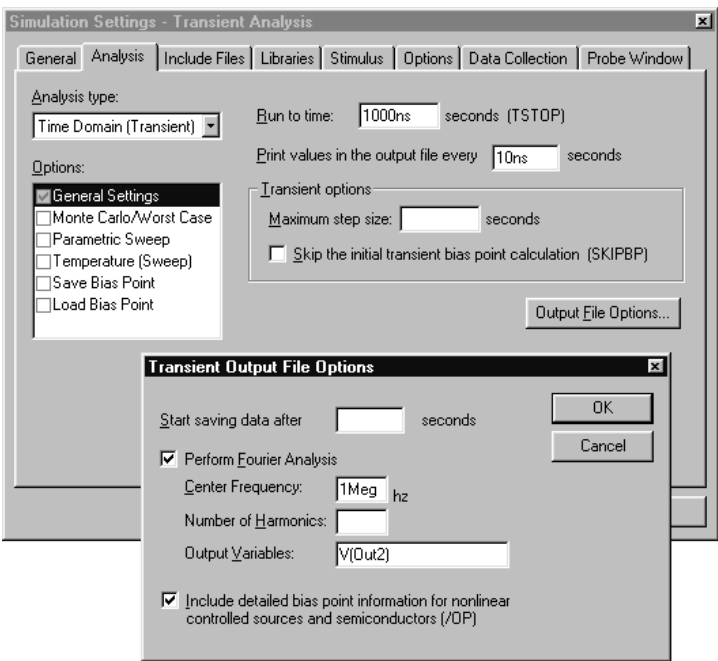
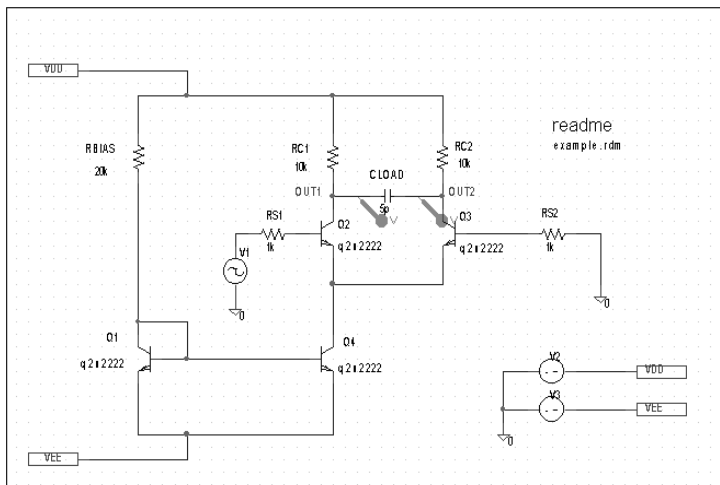


Figure 65 Transient analysis setup for EXAMPLE.OPJ.

During a transient analysis, any or all of the independent sources may have time-varying values. In EXAMPLE.OPJ, the only source which has a time-varying value is V1 (VSIN part) with attributes:

V_{OFF} = 0v
 V_{AMPL} = 0.1v
 FREQ = 5Meg

V1's value varies as a 5 MHz sine wave with an offset voltage of 0 volts and a peak amplitude of 0.1 volts. In general, more than one source has time-varying values; for instance, two or more clocks in a digital circuit.



The example circuit EXAMPLE.OPJ is provided with the OrCAD program installation.

Figure 66 Example schematic EXAMPLE.OPJ.

The transient analysis does its own calculation of a bias point to start with, using the same technique as described for DC sweep. This is necessary because the initial values of the sources can be different from their DC values. To report the small-signal parameters for the transient bias point, use the Transient command and enable Detailed Bias Point. Otherwise, if you simply want the result of the transient run itself, you should only enable the Transient command.

In the simulation output file EXAMPLE.OUT, the bias-point report for the transient bias point is labeled INITIAL TRANSIENT SOLUTION.

Internal time steps in transient analyses

During analog analysis, PSpice A/D maintains an internal time step which is continuously adjusted to maintain accuracy while not performing unnecessary steps. During periods of inactivity, the internal time step is increased. During active regions, it is decreased. The maximum internal step size can be controlled by specifying it in the Step Ceiling text box in the Transient dialog box. PSpice A/D will never exceed either the step ceiling value or two percent of the total transient run time, whichever is less.

The internal time steps used may not correspond to the time steps at which information has been requested to be reported. The values at the print time steps are obtained by second-order polynomial interpolation from values at the internal steps.

See [Chapter 14, Digital simulation](#), for more information on the digital timing analysis of PSpice A/D.

When simulating mixed analog/digital circuits, there are actually two time steps: one analog and one digital. This is necessary for efficiency. Since the analog and digital circuitry usually have very different time constants, any attempt to lock them together would greatly slow down the simulation. The time step shown on the PSpice A/D display during a transient analysis is that of the analog section.

Switching circuits in transient analyses

Running transient analysis on switching circuits can lead to long run times. PSpice A/D must keep the internal time step short compared to the switching period, but the circuit's response extends over many switching cycles.

One method of avoiding this problem is to transform the switching circuit into an equivalent circuit without switching. The equivalent circuit represents a sort of quasi steady-state of the actual circuit and can correctly model the actual circuit's response as long as the inputs do not change too fast.

This technique is described in:

V. Bello, "Computer Program Adds SPICE to Switching-Regulator Analysis,"
Electronic Design, March 5, 1981.

Plotting hysteresis curves

Transient analysis can be used to look at a circuit's hysteresis. Consider, for instance, the circuit shown in Figure 67 (netlist in Figure 68).

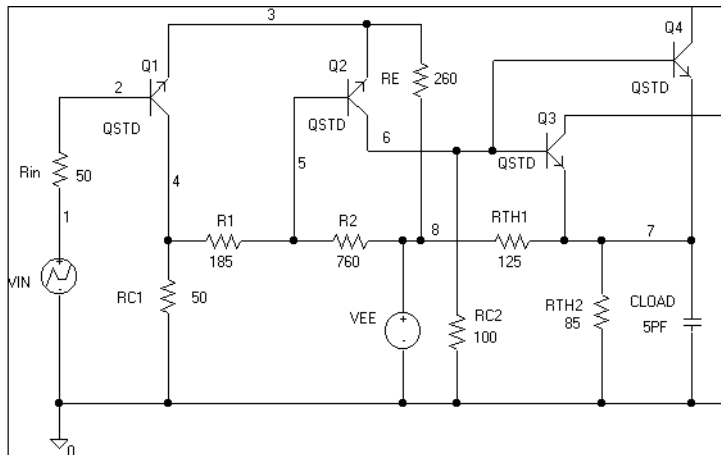


Figure 67 ECL-compatible Schmitt trigger.

```

* Capture Netlist

R_RIN      1 2 50
R_RC1      0 3 50
R_R1       3 5 185
R_R2       5 8 760
R_RC2      0 6 100
R_RE       4 8 260
R_RTH2     7 0 85
C_CLOAD    0 7 5PF
V_VEE      8 0 dc -5
V_VIN      1 0
+PWL 0 -8 1MS -1.0V 2MS -1.8V
R_RTH1     8 7 125
Q_Q1       3 2 4 QSTD
Q_Q2       6 5 4 QSTD
Q_Q3       0 6 7 QSTD
Q_Q4       0 6 7 QSTD

```

Figure 68 Netlist for Schmitt trigger circuit.

The QSTD model is defined as:

```

.MODEL QSTD NPN( is=1e-16 bf=50 br=0.1 rb=50 rc=10
tf=.12ns tr=5ns
+ cje=.4pF pe=.8 me=.4 cjc=.5pF pc=.8 mc=.333 ccs=1pF
va=50)

```

Instead of using the DC sweep to look at the hysteresis, use the transient analysis, (Print Step = .01ms and Final Time = 2ms) sweeping VIN from -1.8 volts to -1.0 volts and back down to -1.8 volts, very slowly. This has two advantages:

- it avoids convergence problems
- it covers both the upward and downward transitions in one analysis

After the simulation, in the Probe window in PSpice A/D, the X axis variable is initially set to be Time. By selecting X Axis Settings from the Plot menu and clicking on the Axis Variable button, you can set the X axis variable to be V(1). Then use Add on the Trace menu to display V(7), and change the X axis to a user defined data range from -1.8V to -1.0V (Axis Settings on the Plot menu). This plots the output of the Schmitt trigger against its input, which is the desired outcome. The result looks similar to Figure 69.

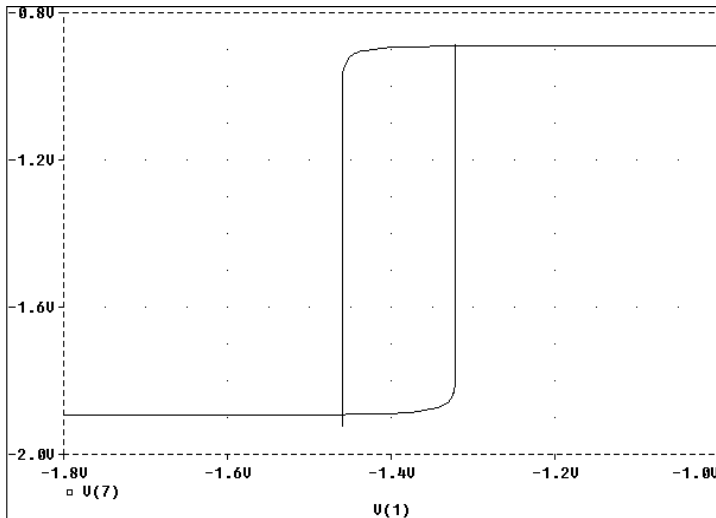


Figure 69 Hysteresis curve example: Schmitt trigger.

Fourier components

Fourier analysis is enabled through the Output File Options dialog box under the Time Domain (Transient) Analysis type. Fourier analysis calculates the DC and Fourier components of the result of a transient analysis. By default, the first through ninth components are computed; however, more can be specified.

When selecting Fourier to run a harmonic decomposition analysis on a transient waveform, only a portion of the waveform is used. Using the Probe window in PSpice A/D, a Fast Fourier Transform (FFT) of the complete waveform can be calculated and its spectrum displayed.

In the example shown in Figure 65 on page 11-356, the voltage waveform at node OUT2 from the transient analysis is to be used and the fundamental frequency is to be one megahertz for the harmonic decomposition. The period of fundamental frequency is one microsecond (inverse of the fundamental frequency). Only the last one microsecond of the transient analysis is used, and that

Note You must do a transient analysis in order to do a Fourier analysis. The sampling interval used during the Fourier transform is equal to the print step specified for the transient analysis.

portion is assumed to repeat indefinitely. Since V1's sine wave does indeed repeat every one microsecond, this is sufficient. In general, however, you must make sure that the fundamental Fourier period fits the waveform in the transient analysis.

Parametric and temperature analysis

12

Chapter overview

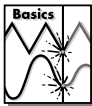
This chapter describes how to set up parametric and temperature analyses. Parametric and temperature are both simple multi-run analysis types.

This chapter includes the following sections:

- [Parametric analysis on page 12-364](#)
- [Temperature analysis on page 12-373](#)

Parametric analysis

Note Parametric analysis is not supported in PSpice A/D Basics.



Minimum requirements to run a parametric analysis

Minimum circuit design requirements

- Set up the circuit according to the swept variable type as listed in [Table 1](#).
- Set up a DC sweep, AC sweep, or transient analysis.

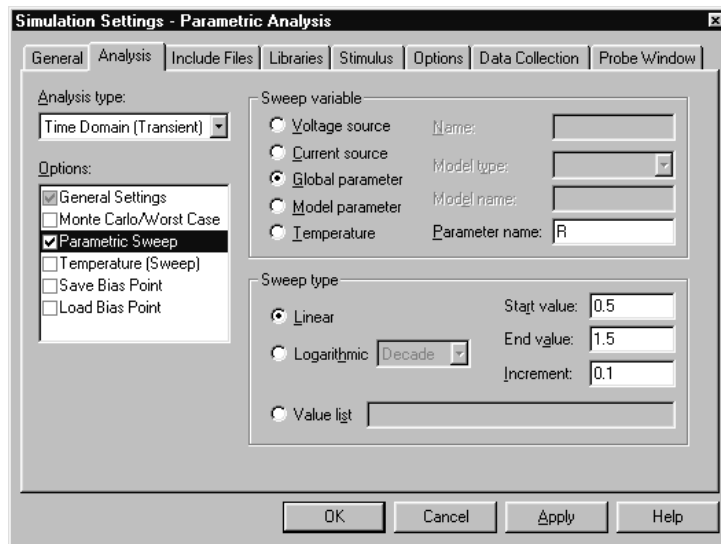
Table 1 Parametric analysis circuit design requirements

Swept variable type	Requirement
voltage source	voltage source with a DC specification (VDC, for example)
temperature	none
current source	current source with a DC specification (IDC, for example)
model parameter	PSpice A / D model
global parameter	global parameter defined with a parameter block (PARAM)

Minimum program setup requirements

- 1 In the Simulation Settings dialog box, from the Analysis type list box, select Time Domain (Transient).
- 2 Under Options, select Parametric Sweep if it is not already enabled.
- 3 Specify the required parameters for the sweep.

See [Setting up analyses on page 8-289](#) for a description of the Simulation Settings dialog box.



- 4 Click OK to save the simulation profile.
- 5 From the PSpice menu, choose Run to start the simulation.

Note *Do not specify a DC sweep and a parametric analysis for the same variable.*

Overview of parametric analysis

Parametric analysis performs multiple iterations of a specified standard analysis while varying a global parameter, model parameter, component value, or operational temperature. The effect is the same as running the circuit several times, once for each value of the swept variable.

See [Parametric analysis on page 2-82](#) for a description of how to set up a parametric analysis.

RLC filter example

This example shows how to perform a parametric sweep and analyze the results with performance analysis.

Use performance analysis to derive values from a series of simulator runs and plot these values versus a parameter that varies between the simulator runs.

For this example, the derived values are the overshoot and the rise time versus the damping resistance of the filter.

Entering the design

The schematic representation for the RLC filter (RLCFILT.OPJ) is shown in Figure 70.

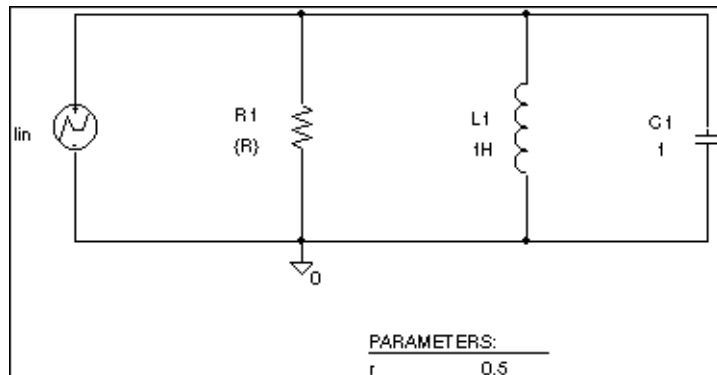


Figure 70 Passive filter schematic.

This series of PSpice A/D runs varies the value of resistor R1 from 0.5 to 1.5 ohms in 0.1 ohm steps. Since the time-constant of the circuit is about one second, perform a transient analysis of approximately 20 seconds.

Create the circuit in OrCAD Capture by placing a piecewise linear independent current source (IPWL from SOURCE.OLB). Set the current source properties as follows:

AC = 1a
 T1 = 0s
 I1 = 0a
 T2 = 10ms

```

I2 = 0a
T3 = 10.1ms
I3 = 1a

```

Place an instance of a resistor and set its VALUE property to the expression, {R}. To define R as a global parameter, place a PARAM pseudocomponent and use the Property Editor to create a new property R and set its value to 0.5. Place an inductor and set its value to 1H, place a capacitor and set its value to 1, and place an analog ground symbol (0 from SOURCE.OLB). Wire the schematic symbols together as shown in Figure 70.

Running the simulation

Run PSpice A/D with the following analyses enabled:

transient	print step:	100ms
	final time:	20s
parametric	swept var. type:	global parameter
	sweep type:	linear
	name:	R
	start value:	0.5
	end value:	1.5
	increment:	0.1

After setting up the analyses, start the simulation by choosing Run from the PSpice menu.

Using performance analysis to plot overshoot and rise time

After performing the simulation that creates the data file RLCFILT.DAT, you can calculate the specified performance analysis goal functions.

When the simulation is finished, a list appears containing all of the sections (runs) in the data file produced by PSpice A/D. To use the data from every run, select All and click OK in the Available Selections dialog box. In the case of Figure 71, the trace I(L1) from the ninth section was added by specifying the following in the Add Traces dialog box:

```
I(L1)@9
```

To display the Add Traces dialog box, from the Trace menu, choose Add Trace or click the Add Trace toolbar button.



Troubleshooting tip

More than one PSpice A/D run or data section is required for performance analysis. Because one data value is derived for each waveform in a related set of waveforms, at least two data points are required to produce a trace.

Use Eval Goal Function (from the Trace menu) to evaluate a goal function on a single waveform and produce a single data point result.



The genrise and overshoot goal functions are contained in the file PSPICE.PRB in the OrCAD directory.

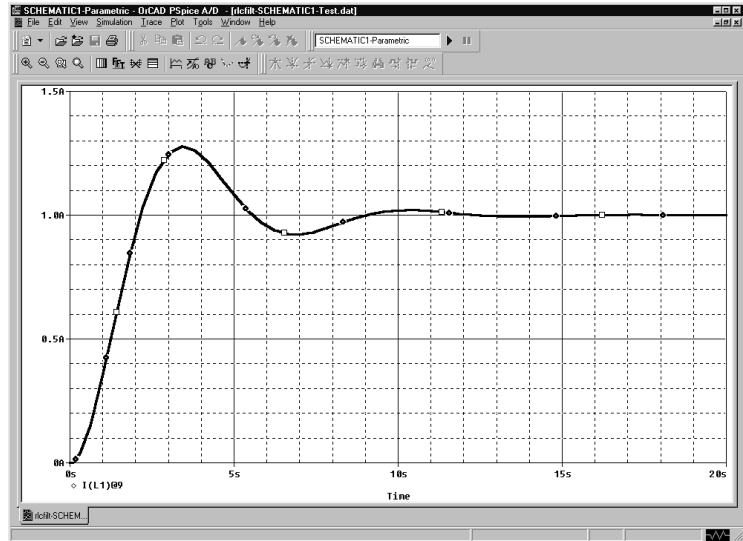


Figure 71 Current of L1 when R1 is 1.5 ohms.

To run performance analysis

- 1 From the Trace menu, choose Performance Analysis .
- 2 Click OK.

PSpice resets the X-axis variable for the graph to be the parameter that changed between PSpice A/D runs. In the example, this is the R parameter.

To see the rise time for the current through the inductor L1, click the Add Trace toolbar button and then enter:

```
genrise( I(L1) )
```

Figure 72, shows how the rise time decreases as the damping resistance increases for the filter.

Another Y axis can be added to the plot for the overshoot of the current through L1 by selecting Add Y Axis from the Plot menu. The Y axis is immediately added. Now click the Add Trace toolbar button and enter:

```
overshoot( I(L1) )
```

Figure 72 shows how the overshoot increases with increasing resistance.

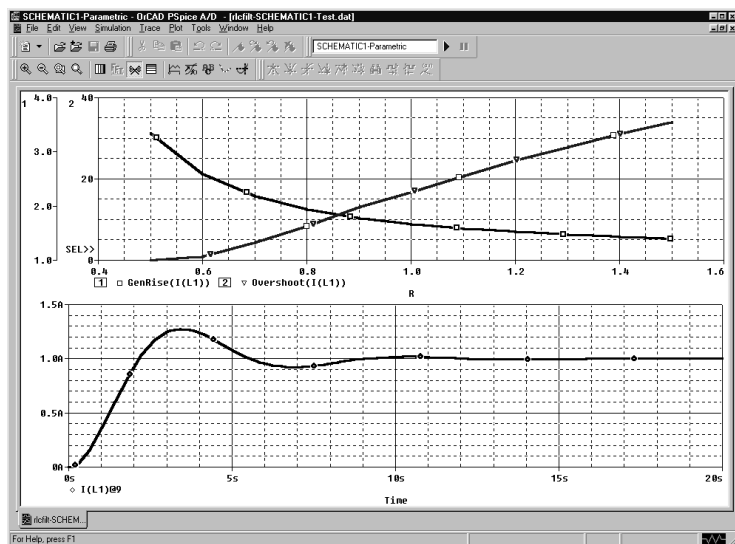


Figure 72 Rise time and overshoot vs. damping resistance.

This technique for measuring branch capacitances works well in both simple and complex circuits.

Example: frequency response vs. arbitrary parameter

You can view a plot of the linear response of a circuit at a specific frequency as one of the circuit parameters varies (such as the output of a band pass filter at its center frequency vs. an inductor value).

In this example, the value of a nonlinear capacitance is measured using a 10 kHz AC signal and plotted versus its bias voltage. The capacitance is in parallel with a resistor, so a trace expression is used to calculate the capacitance from the complex admittance of the R-C pair.

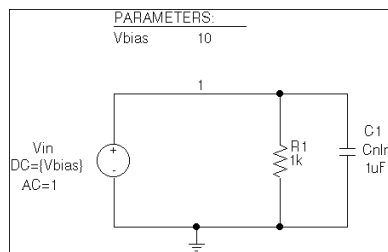


Figure 73 RLC filter example circuit.

Setting up the circuit

Enter the circuit in Capture as shown in Figure 73

To create the capacitor model in the schematic editor:

- 1 Place a CBREAK part.
- 2 Select it so that it is highlighted.
- 3 From the Edit menu, choose PSpice Model.
- 4 In the Model Text frame, enter the following:

```
.model Cnln CAP(C=1 VC1=-0.01 VC2=0.05)
```
- 5 From the File menu, choose Save.

Set up the circuit for a parametric AC analysis (sweep Vbias), and run PSpice A/D. Include only the frequency of interest in the AC sweep.

To display the results

Use PSpice to display the capacitance calculated at the frequency of interest versus the stepped parameter.

- 1 Simulate the circuit.
- 2 Load all AC analysis sections.
- 3 From the Trace menu, choose Add Trace or click the Add Trace toolbar button.
- 4 Add the following trace expression:



```
IMG(-I(Vin)/V(1,0))/(2*3.1416*Frequency)
```

Or add the expression:

```
CvF(-I(Vin)/V(1,0))
```

Where CvF is a macro which measures the effective capacitance in a complex conductance. Macros are defined using the Macros command on the Trace menu. The CvF macro should be defined as:

```
CvF(G)= IMG(G)/(2*3.1416*Frequency)
```

Note *-I(Vin)/V(1) is the complex admittance of the R-C branch; the minus sign is required for correct polarity.*

To use performance analysis to plot capacitance vs. bias voltage

- 1 From the Trace menu, choose Performance Analysis.
- 2 Click Wizard.
- 3 Click Next>.
- 4 Click YatX in the Choose a Goal Function list, and then click Next>.
- 5 In the Name of Trace text box, type the following:

```
CvF(-I(Vin)/V(1))
```
- 6 In the X value to get Y value at text box, type 10K.
- 7 Click Next>.

The wizard displays the gain trace for the first run to text the goal function (YatX).

- 8 Click Finish.

The resultant plot is shown in Figure 74.

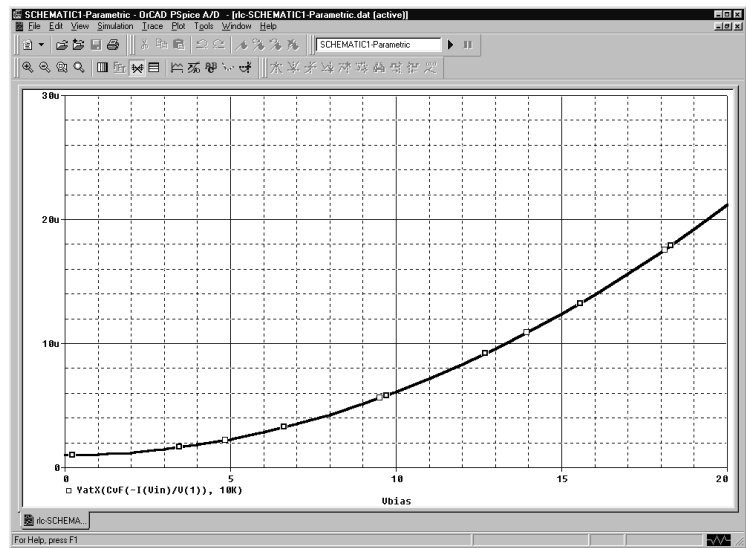


Figure 74 Plot of capacitance versus bias voltage.

Temperature analysis

Minimum requirements to run a temperature analysis

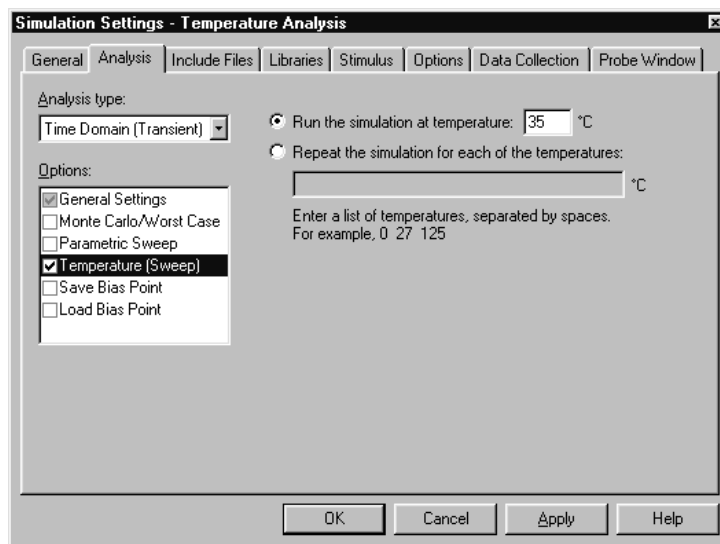
Minimum circuit design requirements

None.

Minimum program setup requirements

- 1 In the Simulation Settings dialog box, from the Analysis type list box, select Time Domain (Transient).
- 2 Under Options, select Temperature Sweep if it is not already enabled.
- 3 Specify the required parameters for the sweep.

See [Setting up analyses on page 8-289](#) for a description of the Simulation Settings dialog box.



- 4 Click OK to save the simulation profile.
- 5 From the PSpice menu, choose Run to start the simulation.

Overview of temperature analysis

Running multiple analyses for different temperatures can also be achieved using parametric analysis (see [Parametric analysis on page 12-364](#)). With parametric analysis, the temperatures can be specified either by list, or by range and increments within the range.

For a temperature analysis, PSpice A/D reruns standard analyses set in the Simulation Settings dialog box at different temperatures.

You can specify zero or more temperatures. If no temperature is specified, the circuit is run at 27°C. If more than one temperature is listed, the simulation runs once for each temperature in the list.

Setting the temperature to a value other than the default results in recalculating the values of temperature-dependent devices. In EXAMPLE.OPJ (see Figure 75), the temperature for all of the analyses is set to 35°C. The values for resistors RC1 and RC2 are recomputed based upon the CRES model which has parameters TC1 and TC2 reflecting linear and quadratic temperature dependencies.

Likewise, the Q3 and Q4 device values are recomputed using the Q2N2222 model which also has temperature-dependent parameters. In the simulation output file, these recomputed device values are reported in the section labeled TEMPERATURE ADJUSTED VALUES.

The example circuit EXAMPLE.OPJ is provided with the OrCAD program installation.

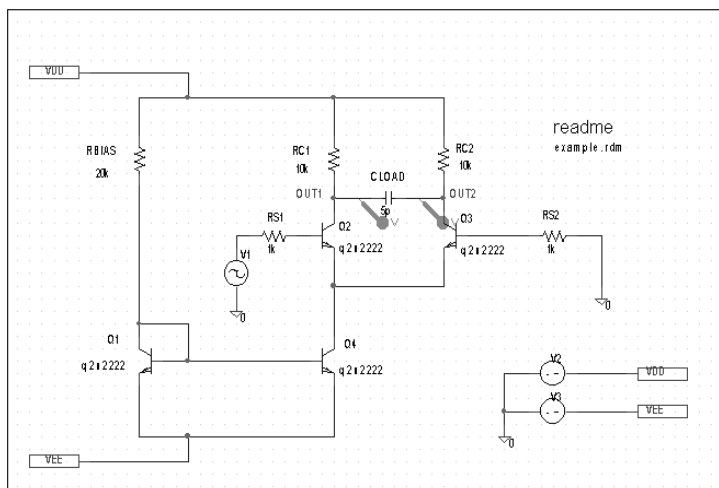


Figure 75 Example schematic EXAMPLE.OPJ.

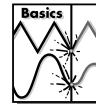
Monte Carlo and sensitivity / worst-case analyses

13

Chapter overview

This chapter describes how to set up Monte Carlo and sensitivity / worst-case analyses and includes the following sections:

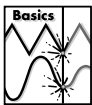
- [Statistical analyses on page 13-376](#)
- [Monte Carlo analysis on page 13-381](#)
- [Worst-case analysis on page 13-398](#)



Note This entire chapter describes features that are not included in PSpice A/D Basics.

Statistical analyses

Note *Statistical analyses are not supported in PSpice A/D Basics.*



Monte Carlo and sensitivity / worst-case are statistical analyses. This section describes information common to both types of analyses.

See [Monte Carlo analysis on page 13-381](#) for information specific to Monte Carlo analyses, and see [Worst-case analysis on page 13-398](#) for information specific to sensitivity / worst-case analyses.

Overview of statistical analyses

Generating statistical results

As the number of Monte Carlo or worst-case runs increases, simulation takes longer and the data file gets larger. Large data files may be slow to open and slow to draw traces.

One way to work around this is to set up an overnight batch job to run the simulation and execute commands. You can even set up the batch job to produce a series of plots on paper to be ready for you in the morning.

The Monte Carlo and worst-case analyses vary the lot or device tolerances of devices between multiple runs of an analysis (DC, AC, or transient). Before running the analysis, you must set up the model and / or lot tolerances of the model parameter to be investigated.

A Monte Carlo analysis performs a Monte Carlo (statistical) analysis of the circuit. A worst-case analysis performs a sensitivity and worst-case analysis of the circuit.

Sensitivity / worst-case analyses are different from Monte Carlo analyses in that they compute the parameters using the sensitivity data rather than using random numbers.

You can run either a Monte Carlo or a worst-case analysis, but you *cannot* run both at the same time. Multiple runs of the selected analysis are done while parameters are varied. You can select only one analysis type (AC, DC, or transient) per run. The selected analysis is repeated in subsequent passes of the analysis.

Output control for statistical analyses

Monte Carlo and sensitivity / worst-case analyses generate the following types of reports:

- Model parameter values used for each run (that is, the values with tolerances applied)
- Waveforms from each run, as a function of specifying data collection, or by specifying output variables in the analysis set up
- Summary of all the runs using a collating function

Output is saved to the data file for use by the waveform analyzer. For Monte Carlo analyses, you can use the performance analysis feature to produce histograms of derived data.

For information about performance analysis, see [RLC filter example on page 12-366](#).

For information about histograms, see [Creating histograms on page 13-395](#).

Model parameter values reports

To produce a list of the model parameters actually used for each run,

- 1 In the Simulation Settings dialog box, click the Analysis tab.
- 2 From the Analysis type list, select an analysis type.
- 3 Under Options, select Monte Carlo / Worst Case.
- 4 Click the More Settings button.
- 5 Select List model parameter values.
- 6 Click OK to close the Simulation Settings dialog box.

This list is written to the simulation output file at the beginning of the run and contains the parameters for each device, as opposed to the parameters for each .MODEL statement. This is because devices can have different parameter values when using a model statement containing a DEV tolerance.

Note that for midsize and large circuits, the List option can produce a large output file.

Waveform reports

For Monte Carlo analyses, there are five variations of the output that you can specify in the Save data from text box on the Monte Carlo dialog box. Options:

In excess of about 10 runs, the waveform display can look more like a band than a set of individual waveforms. This can be useful for seeing the typical spread for a particular output variable. As the number of runs increases, the spread more closely approximates the actual worst-case limits for the circuit.

<none>	No output is generated
All	Forces all output to be generated (including nominal run)
First*	Generates output only during the first <i>n</i> runs
Every*	Generates output for every <i>n</i> th run
Runs(list)*	Does specified analysis and generates outputs only for the listed runs (up to 25 values can be specified in the list)

The * indicates that you can set the number of runs in the runs text box.

Values for the output variables specified in the selected analyses are saved to the simulation output file and data file.

Note *Even a modest number of runs can produce large output files.*

Collating functions

You can further compress the results of Monte Carlo and worst-case analyses. If you use the collating function, a single number represents each run. (Click the Output File Options button and select a function from the Find list.) A table of deviations per run is reported in the simulation output file.

Collating functions are listed in [Table 1](#).

Table 1 *Collating functions used in statistical analyses*

Function	Description
YMAX	Find the greatest difference in each waveform from the nominal
MAX	Find the maximum value of each waveform
MIN	Find the minimum value of each waveform
RISE_EDGE	Find the first occurrence of the waveform crossing above a specified threshold value
FALL_EDGE	Find the first occurrence of the waveform crossing below a specified threshold value

Refer to *Temperature Effects on Monte Carlo Analysis* in the *Application Notes* manual for more information.

Temperature considerations in statistical analyses

The statistical analyses perform multiple runs, as does the temperature analysis. Conceptually, the Monte Carlo and worst-case loops are inside the temperature loop.

However, since both temperature and tolerances affect the model parameters, OrCAD recommends not using temperature analysis when using Monte Carlo or worst-case analysis.

Also, you cannot sweep the temperature in a DC sweep analysis or put tolerances on temperature coefficients while performing one of these statistical analyses. In EXAMPLE.DSN, the temperature value is fixed at 35°C.

The example schematic EXAMPLE.DSN is provided on the OrCAD installation CD.

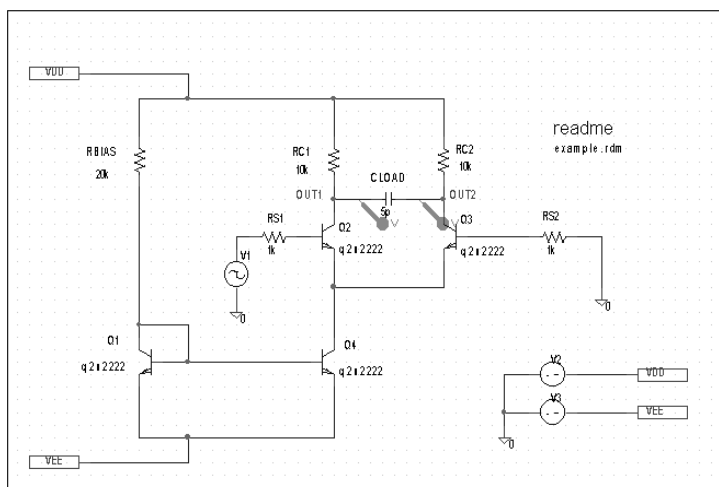


Figure 76 Example schematic EXAMPLE.DSN.

Monte Carlo analysis

The Monte Carlo analysis calculates the circuit response to changes in part values by randomly varying all of the model parameters for which a tolerance is specified. This provides statistical data on the impact of a device parameter's variance.

With Monte Carlo analysis, model parameters are given tolerances, and multiple analyses (DC, AC, or transient) are run using these tolerances.

For EXAMPLE.DSN in Figure 76 on page 13-380, you can analyze the effects of variances in the values of resistors RC1 and RC2 by assigning a model description to these resistors that includes a 5% device tolerance on the multiplier parameter R.

Then you can perform a Monte Carlo analysis. First, the simulator performs a DC analysis with the nominal R multiplier value for RC1 and RC2. Then it performs a set number of additional runs with the R multiplier varied independently for RC1 and RC2 within a 5% tolerance.

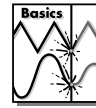
To modify example.dsn and set up simulation

- 1 Replace RC1 and RC2 with RBREAK parts, setting property values to match the resistors that are being replaced (VALUE=10k) and reference designators to match previous names.
- 2 Select PSpice Model from the Edit menu. Create the model CRES as follows:

```
.MODEL CRES RES( R=1 DEV=5% TC1=0.02
+ TC2=0.0045 )
```

From the File menu, choose Save. By default, Capture saves the definition to the model library EXAMPLE.LIB and automatically configures the file for local use with the current schematic.

- 3 In Capture, set up a new Monte Carlo analysis as shown in Figure 77. The analysis specification tells PSpice A/D to do one nominal run and four Monte



Note Monte Carlo analysis is not supported in PSpice A/D Basics.

Monte Carlo analysis is frequently used to predict yields on production runs of a circuit.

TC1 is the linear temperature coefficient.
TC2 is the quadratic temperature coefficient.

Carlo runs, saving the DC analysis output from those five runs.

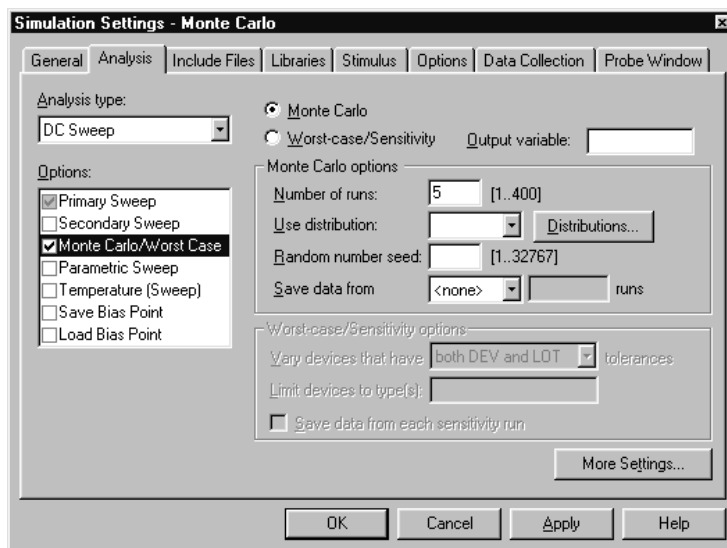


Figure 77 Monte Carlo analysis setup for EXAMPLE.DSN.

PSpice A / D starts by running *all* of the analyses enabled in the Simulation Settings dialog box with all parameters set to their nominal values.

However, with Monte Carlo enabled, PSpice A / D saves the DC sweep analysis results for later reference and comparison. After the nominal analyses are finished, PSpice A / D performs the additional specified analysis runs (in this example, DC sweep).

Subsequent runs use the same analysis specification as the nominal run with one major exception: instead of using the nominal parameter values, the tolerances are applied to set new parameter values and thus, new part values.

There is a trade-off in choosing the number of Monte Carlo runs. More runs provide better statistics, but they require more time. The amount of time scales directly with the number of runs: 20 transient analyses take 20 times as long as one transient analysis. During Monte Carlo runs, the PSpice A / D status display includes the current run number and the total number of runs left.

PSpice A/D offers a facility to generate histograms of data derived from Monte Carlo waveform families through the performance analysis feature.

For information about performance analysis, see [RLC filter example on page 12-366](#).

For information about histograms, see [Creating histograms on page 13-395](#).

Reading the summary report

The summary report generated in this example (see Figure 78) specifies that the waveform generated from V(OUT1, OUT2) should be the subject of the collating function YMAX. In each of the last four runs, the new V(OUT1, OUT2) waveform is compared to the nominal V(OUT1, OUT2) waveform for the first run, calculating the maximum deviation in the Y direction (YMAX collating function). The deviations are printed in order of size along with their run number .

```

****      SORTED DEVIATIONS OF V(OUT1,OUT2) TEMPERATURE =   35.000 DEG C

              MONTE CARLO SUMMARY
*****

Mean Deviation =   -.2477
Sigma          =    .3035

  RUN      MAX DEVIATION FROM NOMINAL
Pass   3      .5729 (1.89 sigma) lower at U_U1 =   -.02
          ( 94.885% of Nominal)
Pass   4      .3549 (1.17 sigma) lower at U_U1 =   -.02
          ( 96.832% of Nominal)
Pass   2      .3122 (1.03 sigma) lower at U_U1 =   -.02
          ( 97.212% of Nominal)
Pass   5      .2493 ( .82 sigma) higher at U_U1 =   -.02
          (102.23% of Nominal)

```

Figure 78 Summary of Monte Carlo runs for *EXAMPLE.OPJ*.

With the List option enabled, a report is also generated showing the parameter value used for each device in each run. In this case (see Figure 79), run three shows the highest deviation.

```
* C:\ORCAD\EX\EXAMPLE.SCH

****      UPDATED MODEL PARAMETERS      TEMPERATURE =  35.000 DEG C

              MONTE CARLO PASS      3

*****

**** CURRENT MODEL PARAMETERS FOR DEVICES REFERENCING cres
              R_RC1      R_RC2
R      1.0304E+00      9.5053E-01
```

Figure 79 *Parameter values for Monte Carlo pass three.*

Example: Monte Carlo analysis of a pressure sensor

This example shows how the performance of a pressure sensor circuit with a pressure-dependent resistor bridge is affected by manufacturing tolerances, using Monte Carlo analysis to explore these effects.

Drawing the schematic

To begin, construct the bridge as shown in Figure 80.

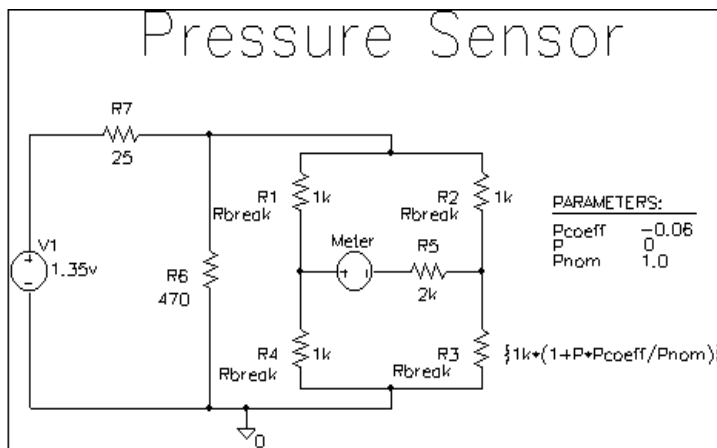


Figure 80 Pressure sensor circuit.

Here are a few things to know when placing and connecting the part:

- To get the part you want to place, from the Place menu, choose Part.
- To rotate a part before placing it, press **[R]**.
- For V1 and Meter, place a generic voltage source using the VSRC part. When you place the source for the meter, change its name by double-clicking the part and typing Meter in the Reference cell in the Parts Spreadsheet.
- For R1-R7, place a resistor using the R part.
- Place the analog ground using the 0 ground symbol.



- To connect the parts, from the Place menu, choose Wire.
- To move values or reference designators, click the value or reference designator to select it, then drag it to the new location.

Note Because the Meter source is used to measure current, it has no DC value and can be left unchanged.

Defining part values

Define the part values as shown in Figure 80. For the pressure sensor, you need to do the following:

- Change the resistor values for R3, R5, R6, and R7 from their default value of 1 k.
- Set the DC value for the V1 voltage source.

To change resistor values

- 1 Double-click the value for a resistor.
- 2 Type the new value. Depending on the resistor you are changing, set its value to one of the following (refer to Figure 80).

Table 1

If you are changing this resistor...	Type this...
R3	{1k*(1+P*Pcoeff/Pnom)}
R5	2k
R6	470
R7	25

Note The value for R3—{1k*(1+P*Pcoeff/Pnom)}—is an expression that represents linear dependence of resistance on pressure. To complete the definition for R3, you will create and define global parameters for Pcoeff, P, and Pnom later on in this example

- 3 Repeat steps 1-2 for each resistor on your schematic page.

To set the DC value for the V1 source and make it visible

- 1 Double-click the V1 source part.
- 2 In the Parts Spreadsheet, click in the cell under the DC column.
- 3 Type 1.35v.

- 4 Click the Display button.
- 5 In the Display Format frame, choose the Value Only option to make the DC value (1.35v) visible on the schematic.
- 6 Click OK, then click Apply to apply the changes you have made to the part.
- 7 Close the Parts Spreadsheet.

Setting up the parameters

To complete the value specification for R3, define the global parameters Pcoeff, P, and Pnom.

To define and initialize Pcoeff, P, and Pnom

- 1 Place a PARAM part on the schematic page.
- 2 Double-click the PARAM part to display the Parts Spreadsheet.
- 3 For each parameter, create a new property by clicking New and typing its name. Enter its corresponding value by clicking in the cell under the new property name and typing its value. Specify the parameter name and corresponding value as follows.

Table 2

Property	Value
Pcoeff	-0.06
P	0
Pnom	1.0

- 4 Click Apply to save the changes you have made then close the Parts Spreadsheet.

When PSpice A/D runs a Monte Carlo analysis, it uses tolerance values to determine how to vary model parameters during the simulation.

Using resistors with models

To explore the effects of manufacturing tolerances on the behavior of this circuit, you set device (DEV) and (LOT) tolerances on the model parameters for resistors R1, R2, R3, and R4 in a later step (see page [13-389](#)). This means you need to use resistor parts that have model associations.

Because R parts do not have associated models (and therefore no model parameters), change the resistor parts to Rbreak parts that do have models.

To replace R1, R2, R3, and R4 with the RBREAK part

- 1 Click R1 to select it.
- 2 Hold down the **[Ctrl]** key and click R2, R3 and R4 to add them to the selection set.
- 3 Press **[Delete]** to delete the selection set.
- 4 From the Place menu, choose Part.
- 5 Type RBREAK in the Part text box. (If RBREAK is not available, click the Add Library button and select BREAKOUT.OLB to configure it for use in Capture.)
- 6 Click OK.
- 7 Manually place the RBREAK part in the circuit diagram where R1, R2, R3 and R4 were located.
- 8 Double-click on each RBREAK part and change the reference designators as desired.

Saving the design

Before editing the models for the Rbreak resistors, save the schematic.

To save the design

- 1 From Capture's File menu, choose Save.

Defining tolerances for the resistor models

This section shows how to assign device (DEV) and lot (LOT) tolerances to the model parameters for resistors R1, R2, R3, and R4 using the model editor.

To assign 2% device and 10% lot tolerances to the resistance multiplier for R1

- 1 Select R1.
- 2 From the Edit menu, choose PSpice Model.
Capture searches the libraries for the Rbreak model definition and makes a copy to create an instance model.
- 3 To change the instance model name from Rbreak to Rmonte1, do the following:
 - a In the Model Text frame, double-click Rbreak.
 - b Type RMonte1.
- 4 To add a 2% device tolerance and a 10% lot tolerance to the resistance multiplier, do the following:
 - a Add the following to the .MODEL statement (after R=1):

```
DEV=2% LOT=10%
```

The model editing window should look something like Figure 81.

- 5 From the File menu, choose Save.

By default, Capture saves the RMonte1 .MODEL definition to the design_name.lib library, which is

You can use the model editor to change the .MODEL or .SUBCKT syntax for a model definition. To find out more about the model editor, see [Editing model text on page 4-152](#), or refer to the online *PSpice Reference Manual*.

To find out more about adding model libraries to the configuration, see [Configuring model libraries on page 4-162](#).

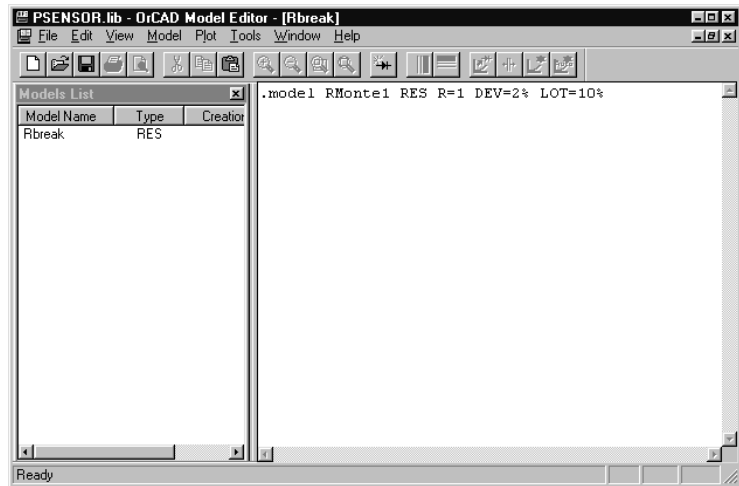


Figure 81 Model definition for RMonte1.

PSENSOR.LIB. Capture also automatically configures the library for local use.

To have resistors R2 and R4 use the same tolerances as R1

- 1 In Capture's schematic page editor, select R2 and R4.
- 2 From the Edit menu, select Properties.
- 3 In the R2 row, click in the cell under the Implementation column and type RMonte1.
- 4 In the R4 row, click in the cell under the Implementation column and type RMonte1.

To assign 5% device tolerance to the resistance multiplier for R3

- 1 Select R3.
- 2 From the Edit menu, select PSpice Model.
- 3 In the Model Text frame, change the .MODEL statement to:

```
.model RTherm RES R=1 DEV=5%
```

- 4 From the File menu, choose Save.

Your schematic page should look like Figure 82.

Setting up the analyses

This section shows how to define and enable a DC analysis that sweeps the pressure value and a Monte Carlo analysis that runs the DC sweep with each change to the resistance multipliers.

To set up the DC sweep

- 1 In the PSpice menu, choose New Simulation Profile or Edit Simulation Settings. (If this is a new simulation, enter the name of the profile and click OK.)

See [Setting up analyses on page 8-289](#) for a description of the Simulation Settings dialog box.

The Simulation Settings dialog box appears.

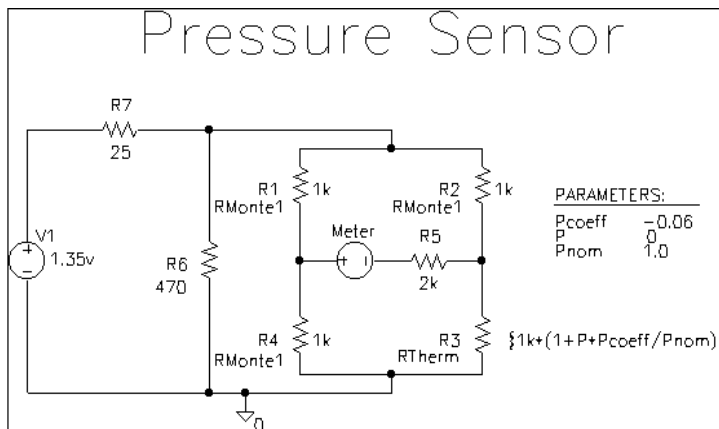


Figure 82 Pressure sensor circuit with RMonte1 and RTherm model definitions.

- 2 Select DC Sweep in the Analysis type list box.
- 3 In the Sweep Variable frame, select Global Parameter.
- 4 Enter the following values:

Table 3

In this text box...	Type this...
Parameter name	P
Start value	0
End value	5.0
Increment	0.1

To set up the Monte Carlo analysis

- 1 Select the Monte Carlo / Worst Case option.
- 2 Check Monte Carlo if it is not already selected.
- 3 In the Number of runs text box, type 10.
- 4 In the Save data from list box, select All.
- 5 Type I(Meter) in the Output variable text box.
- 6 Click OK to save the simulation profile.

Running the analysis and viewing the results

To complete setup, simulate, and view results

- 1 From Capture’s PSpice menu, choose Run to start the simulation

When the simulation is complete, PSpice A / D automatically displays the selected waveform. Because PSpice A / D ran a Monte Carlo analysis, it saved multiple runs or sections of data. These are listed in the Available Sections dialog box.
- 2 From PSpice A / D’s Trace menu, choose Performance Analysis.
- 3 Click the Select sections button.
- 4 In the Available Sections dialog box, click the All button.
- 5 Click OK.

- 6 To display current through the Meter voltage source, do the following:
 - a From Capture's PSpice menu, point to markers and choose Current into Pin.
 - b Place a current probe on the left-hand pin of the Meter source.
- 7 Switch to the Probe window to see the family of curves for I(Meter) as a function of P.

Note *For more on analyzing Monte Carlo results in PSpice A/D, see the next section on Monte Carlo histograms.*

Another way to view the family of curves without using schematic markers is as follows:

- 1 From PSpice A/D's Trace menu, choose Add Trace.
- 2 In the Simulation Output Variables list, double-click I(Meter).

Monte Carlo Histograms

You can display data derived from Monte Carlo waveform families as histograms. This is part of the performance analysis feature.

In this example, you simulate a fourth-order Chebyshev active filter, running a series of 100 AC analyses while randomly varying resistor and capacitor values for each run. Then, having defined performance analysis goal functions for bandwidth and center frequency, you observe the statistical distribution of these quantities for the 100 runs.

Monte Carlo analysis is frequently used to predict yields on production runs of a circuit.

For more information about performance analysis, see [RLC filter example on page 12-366](#).

Chebyshev filter example

The Chebyshev filter is designed to have a 10 kHz center frequency and a 1.5 kHz bandwidth. The schematic page for the filter is shown in Figure 83. The stimulus specifications for V1, V2, and V3 are:

V1: DC=-15
 V2: DC=+15
 V3: AC=1

The parts are rounded to the nearest available 1% resistor and 5% capacitor value. In this example, note how the

bandwidth and the center frequency vary when 1% resistors and 5% capacitors are used in the circuit.

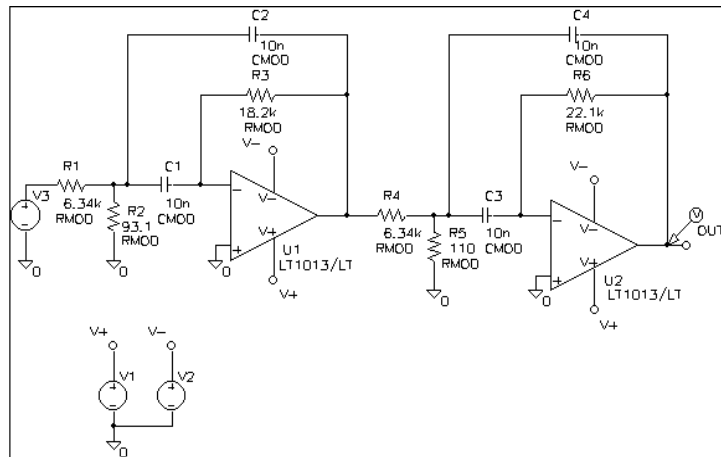


Figure 83 Chebyshev filter.

Creating models for Monte Carlo analysis

To vary the resistors and capacitors in the filter circuit, create models for these parts on which you can set device tolerances for Monte Carlo analysis. The BREAKOUT.OLB library contains generic devices for this purpose. The resistors and capacitors in this schematic are the Rbreak and Cbreak parts from BREAKOUT.OLB.

Using the Model Editor, modify the models for these parts as follows:

```
.model RMOD RES(R=1 DEV=1%)
.model CMOD CAP(C=1 DEV=5%)
```

Setting up the analysis

To analyze the filter, set up both an AC analysis and a Monte Carlo analysis. The AC analysis sweeps 50 points per decade from 100 Hz to 1 MHz. The Monte Carlo analysis is set to take 100 runs. The analysis type is AC and the output variable is V(OUT).

To set up the analysis

- 1 From PSpice A/D's Trace menu, choose Performance Analysis.
- 2 In the Save data from list box, choose All.
- 3 Click OK.

Creating histograms

Because the data file can become quite large when running a Monte Carlo analysis, to view just the output of the filter, you place a voltage probe at the output of the filter.

To collect data for the marked node only

- 1 From the PSpice menu, choose New Simulation Profile or Edit Simulation Settings from the PSpice menu. (If this is a new simulation, enter the name of the profile and click OK.)

The Simulation Settings dialog box appears.

- 2 On the Data Collection tab, choose the At Probes only option.
- 3 Click OK.

To run the simulation and load Probe with data

- 1 From Capture's PSpice menu, choose Run to start the simulation.

When the simulation is complete, PSpice A/D automatically displays the selected waveform. Because PSpice A/D ran a Monte Carlo analysis, it saved multiple runs or sections of data. These are listed in the Available Sections dialog box.

- 2 From PSpice A/D's Trace menu, choose Performance Analysis.
- 3 Click the Select sections button.
- 4 In the Available Sections dialog box, click All.

For information about performance analysis, see [RLC filter example on page 12-366](#).

You can also display this histogram by using the performance analysis wizard to display Bandwidth (VDB(OUT) , 1).

- 5 Click OK.

To display a histogram for the 1 dB bandwidth

- 1 From PSpice A / D's Plot menu, choose Axis Settings.
- 2 Select the X Axis tab.
- 3 In the Processing Options frame, select the Performance Analysis check box.
- 4 Click OK.

The histogram display appears. The Y axis is the percent of samples.

- 5 From the Trace menu, choose Goal Functions.
- 6 Choose Bandwidth.
- 7 Click Eval.
- 8 Enter VDB(OUT) in the Name of trace to search text box.
- 9 Enter 1 in the db level down for bandwidth calc text box.
- 10 Click OK, then click Close to view the histogram.

To change the number of histogram divisions

- 1 From the Tools menu, choose Options.
- 2 In the Number of Histogram Divisions text box, replace 10 with 20.
- 3 Click OK.

The histogram for 1 dB bandwidth is shown in Figure 84.

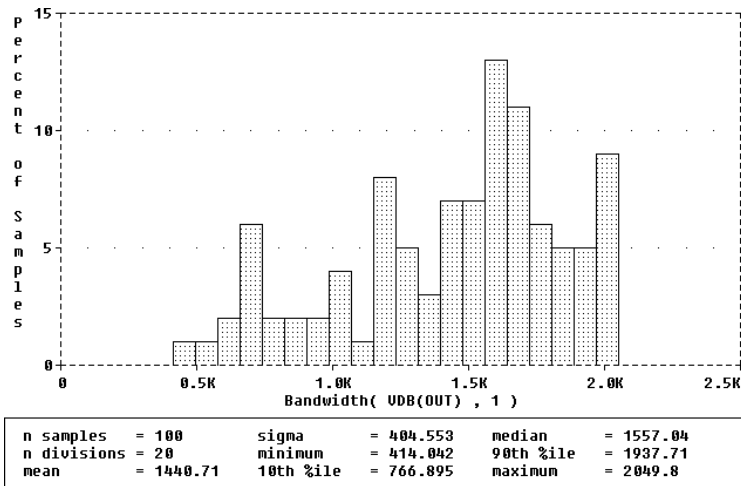


Figure 84 1 dB bandwidth histogram.

The statistics for the histogram are shown along the bottom of the display. The statistics show the number of Monte Carlo runs, the number of divisions or vertical bars that make up the histogram, mean, sigma, minimum, maximum, 10th percentile, median, and 90th percentile.

You can also show the distribution of the center frequency of the filter.

To display the center frequency

- 1 From the Trace menu, choose Goal Functions.
- 2 Choose CenterFreq.
- 3 Click Eval.
- 4 Enter `VDB(OUT)` in the Name of trace to search text box.
- 5 Enter 1 in the db level down for measurement text box.
- 6 Click OK, then click Close to view the histogram.

The new histogram replaces the previous histogram. To display both histograms at once, choose Add Plot to Window on the Plot menu before choosing Add from the Trace menu. The histogram of the center frequency is as shown in Figure 85.

If needed, you can turn off the statistical data display as follows:

- 1 From the Tools menu, choose Options.
- 2 Clear the Display Statistics check box.
- 3 Click Save, and then OK.

Ten percent of the goal function values is less than or equal to the 10th percentile number, and 90% of the goal function values is greater than or equal to that number.

If there is more than one goal function value that satisfies this criteria, then the 10th percentile is the midpoint of the interval between the goal function values that satisfy the criteria. Similarly, the median and 90th percentile numbers represent goal function values such that 50% and 90% (respectively) of the goal function values are less than or equal to those numbers.

Sigma is the standard deviation of the goal function values.

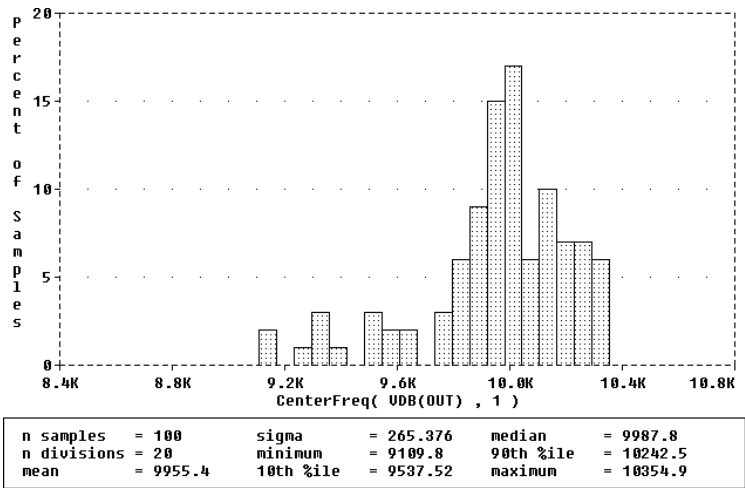
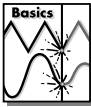


Figure 85 Center frequency histogram.

Worst-case analysis

Note Worst-case analysis is not supported in PSpice A/D Basics.



This section discusses the analog worst-case analysis feature of PSpice A/D. The information provided in this section explains how to use worst-case analysis properly and with realistic expectations.

Overview of worst-case analysis

Worst-case analysis is used to find the worst probable output of a circuit or system given the restricted variance of its parameters. For instance, if the values of R1, R2, and R3 can vary by $\pm 10\%$, then the worst-case analysis attempts to find the combination of possible resistor values which result in the worst simulated output. As with any other analysis, there are three important parts: inputs, procedure, and outputs.

Inputs

In addition to the circuit description, you need to provide two pieces of information:

- the parameter tolerances
- a definition of what *worst* means

You can set tolerances on any number of the parameters that characterize a model.

The criterion for determining the *worst* values for the relevant model parameters is defined in the `.WC` statement as a function of any standard output variable in a specified range of the sweep.

In a given range, reduce the measurement to a single value by one of these five collating functions:

MAX	Maximum output variable value
MIN	Minimum output variable value
YMAX	Output variable value at the point where it differs the most with the nominal run
RISE_EDGE (value)	Sweep value where the output variable value crosses <i>above</i> a given threshold value
FALL_EDGE (value)	Sweep value where the output variable value crosses <i>below</i> a given threshold value

You can define Worst as the highest (HI) or lowest (LO) possible collating function relative to the nominal run.

Procedure

To establish the initial value of the collating function, worst-case analysis begins with a nominal run using all model parameters at their nominal values.

Next, multiple sensitivity analyses determine the individual effect of each model parameter on the collating function. This is accomplished by varying model parameters, one at a time, in consecutive simulations. The

You can define models for nearly all primitive analog circuit parts, such as resistors, capacitors, inductors, and semiconductor devices. PSpice A/D reads the standard model parameter tolerance syntax specified in the `.MODEL` statement. For each model parameter, PSpice A/D uses the nominal, minimum, and maximum probable values, and the DEV and/or LOT specifiers; the probability distribution type (such as UNIFORM or GAUSS) is ignored.

You can use analog behavioral models to measure waveform characteristics other than those detected by the available collating functions, such as rise time or slope. You can also use analog behavioral models to incorporate several voltages and currents into one output variable to which a collating function may be applied. See [Chapter 6, Analog behavioral modeling](#), for more information.

This procedure saves time by performing the minimum number of simulations required to make an educated guess at the parameter values that produce the worst results. It also has some limitations, which are described in the following sections.

direction (*better* or *worse*) in which the collating function changes with a small increase in each model parameter is recorded.

Finally, for the worst-case run, each parameter value is taken as far from its nominal as allowed by its tolerance, in the direction which should cause the collating function to be its worst (given by the HI or LO specification).

Outputs

A summary of the sensitivity analysis is printed in the PSpice A / D output file (.OUT). This summary shows the percent change in the collating function corresponding to a small change in each model parameter. If a .PROBE statement is included in the circuit file, then the results of the nominal and worst-case runs are saved for viewing in the Probe window.

Caution: An important condition for correct worst-case analysis

Worst-case analysis is not an optimization process; it does not *search* for the set of parameter values that result in the worst result.

It assumes that the worst case occurs when each parameter has been either pushed to one of its limits or left at its nominal value as indicated by the sensitivity analysis. **It shows the true worst-case results when the collating function is *monotonic* within all tolerance combinations.**

Otherwise, there is no guarantee. Usually you cannot be certain whether this condition is true, but insight into the operation of the circuit may alert you to possible anomalies.

Worst-case analysis example

The schematic shown in Figure 86 is for an amplifier circuit that is a biased BJT. This circuit is used to demonstrate how a simple worst-case analysis works. It also shows how *non-monotonic* dependence of the output on a single parameter can adversely affect the worst-case analysis.

Because an AC (small-signal) analysis is being performed, setting the input to unity means that the output, $V_m([OUT])$, is the magnitude of the gain of the amplifier. The only variable declared in this circuit is the resistance of $Rb2$. Because the value of $Rb2$ determines the bias on the BJT, it also affects the amplifier's gain.

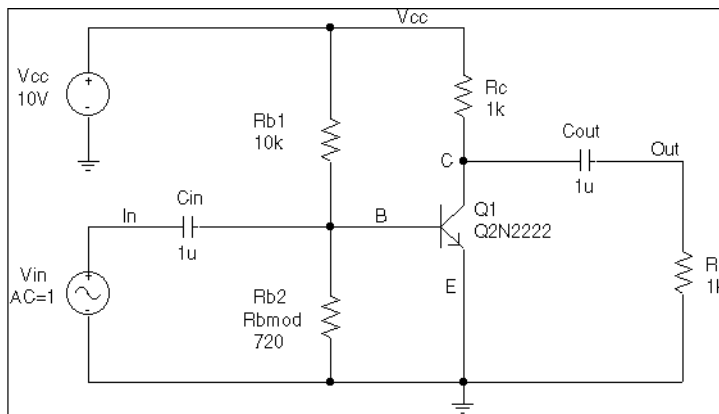


Figure 86 Simple biased BJT amplifier.

Figure 87 is the circuit file used to run one of the following:

- a parametric analysis (.STEP, shown enabled in the circuit file) that sets the value of resistor $Rb2$ by stepping model parameter R through values spanning the specified DEV tolerance range, or
- a worst-case analysis (shown disabled in the circuit file) that allows PSpice A/D to determine the worst-case value for parameter R based upon a sensitivity analysis.

Only one of these analyses can run in any given simulation.

Note *The AC and worst-case analysis specifications (.AC and .WC statements) are written so that the worst-case analysis tries to minimize $V_m([OUT])$ at 100 kHz.*

The netlist and circuit file in Figure 87 are set up to run either a parametric (.STEP) or worst-case (.WC) analysis of the specified AC analysis. These simulations demonstrate the conditions under which worst-case analysis works well and those that can produce misleading results when

```
* Worst-case analysis comparing monotonic and non-monotonic
* output with a variable parameter

.lib

***** Input signal and blocking capacitor *****
Vin In      0      ac      1
Cin In      B      1u

***** "Amplifier" *****
* gain increases with small increase in Rb2, but
* device saturates if Rb2 is maximized.
Vcc Vcc      0      10
Rc  Vcc      C      1k
Q1  C        B      0      Q2N2222
Rb1 Vcc      B      10k
Rb2 B        0      Rbmod 720
.model Rbmod res(R=1 dev 5%)          ; WC analysis results
                                       ; are correct
* .model Rbmod res(R=1.1 dev 5%)      ; WC analysis misled
                                       ; by sensitivity

***** Load and blocking capacitor *****
CoutC      Out      1u
Rl  Out      0      1k

* Run with either the .STEP or the .WC, but not both.
* This circuit file is currently set up to run the .STEP
* (.WC is commented out)

**** Parametric Sweep—providing plot of  $V_m([OUT])$  vs. Rb2 ****
.STEP Res Rbmod(R) 0.8 1.2 10m

***** Worst-case analysis *****
* run once for each of the .model definitions stated above)
* WC AC  $V_m([Out])$  min range 99k 101k list output all

.AC Lin 3 90k 110k
.probe
.end
```

Figure 87 *Amplifier netlist and circuit file.*

output is not monotonic with a variable parameter (see Figure 89 and Figure 90)

For demonstration, the parametric analysis is run first, generating the curve shown in Figure 89 and Figure 90. This curve, derived using the YatX goal function shown in Figure 88 illustrates the non-monotonic dependence of gain on *Rb2*.

```
YatX(1, X_value)=y1{1|sfxv(X_value)!1;}
```

Figure 88 *YatX Goal Function.*

To do this yourself, place the goal function definition in a PROBE.GF file in the circuit directory. Then start PSpice A/D, load all of the AC sweeps, set up the X axis for performance analysis, and add the following trace:

```
YatX(Vm([OUT]),100k)
```

Next, the parametric analysis is commented out and the worst-case analysis is enabled. Two runs are made using the two versions of the *Rbmod* .MODEL statement shown in the circuit file. The model parameter, R, is a multiplier which is used to scale the nominal value of any resistor referencing the *Rbmod* model (*Rb2* in this case).

The first .MODEL statement leaves the nominal value of *Rb2* at 720 ohms. The sensitivity analysis increments R by a small amount and checks its effect on Vm([OUT]). This slight increase in R causes an increase in the base bias voltage of the BJT, and increases the amplifier's gain, Vm([OUT]). The worst-case analysis correctly sets R to its minimum value for the lowest possible Vm([OUT]) (see Figure 89).

Note The YatX goal function is used on the simulation results for the parametric sweep (.STEP) defined in Figure 87. The resulting curves are shown in Figure 89 and Figure 90.

Consider a slightly different scenario: R_{b2} is set to 720 ohms so that maximizing it is not enough to saturate the BJT, but R_{b1} is variable also. The true worst case occurs when R_{b2} is maximized and R_{b1} is minimized. Checking their individual effects is not sufficient, even if the circuit were simulated four times with each resistor in turn set to its extreme values.

Output is monotonic within the tolerance range. Sensitivity analysis correctly points to the minimum value.

The second .MODEL statement scales the nominal value of R_{b2} by 1.1 to approximately 800 ohms. The gain still increases with a small increase in R , but a larger increase in R increases the base voltage so much that it drives the BJT into saturation and nearly eliminates the gain. The worst-case analysis is fooled by the sensitivity analysis into assuming that R_{b2} must be minimized to degrade the gain, but maximizing R_{b2} is much worse (see Figure 90). Note that even an optimizer, which checks the local gradients to determine how the parameters should be varied, is fooled by this circuit.

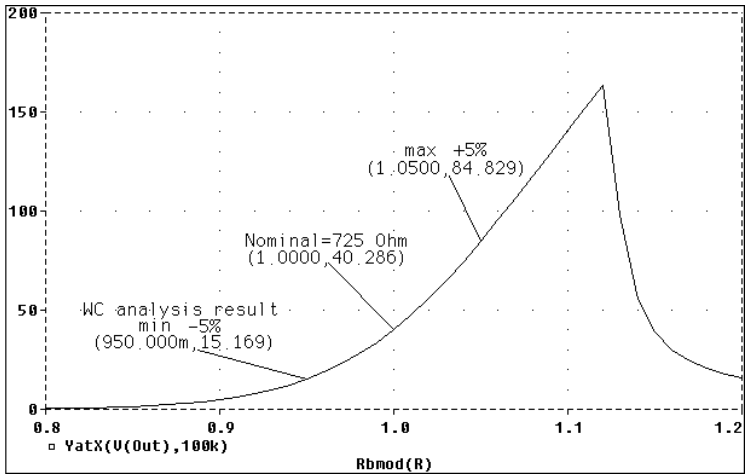


Figure 89 Correct worst-case results.

Output is non-monotonic within the tolerance range, thus producing incorrect worst-case results.

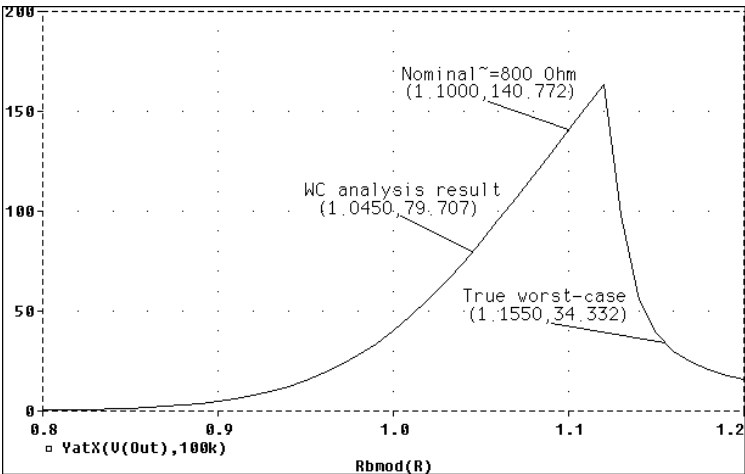


Figure 90 Incorrect worst-case results.

Tips and other useful information

VARY BOTH, VARY DEV, and VARY LOT

When VARY BOTH is specified in the .WC statement and a model parameter is specified with both DEV and LOT tolerances defined, the worst-case analysis may produce unexpected results. The sensitivity of the collating function is only tested with respect to LOT variations of such a parameter.

For example, during the sensitivity analysis, the parameter is varied once affecting all devices referring to it and its effect on the collating function is recorded. For the worst-case analysis, the parameter is changed for all devices by LOT + DEV in the determined direction. See the example schematic in Figure 91 and circuit file in Figure 92.

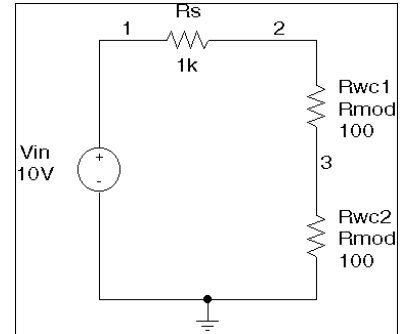


Figure 91 Schematic using VARY BOTH.

```
WCASE  VARY  BOTH   Test
Vin          1      0    10V
Rs           1      2    1K
Rwc1         2      3    Rmod   100
Rwc2         3      0    Rmod   100
.MODEL  Rmod  RES(R=1  LOT 10%  DEV 5%)
.DC Vin      LIST    10
.WC DC      V(3)    MAX      VARY BOTH      LIST    OUTPUT ALL
.ENDS
```

Figure 92 Circuit file using VARY BOTH.

In this case, V(3) is maximized if:

- *Rwc1* and *Rwc2* are both increased by 10% per the LOT tolerance specification, and
- *Rwc1* is decreased by 5% and *Rwc2* is increased by 5% per the DEV tolerance specification.

The final values for *Rwc1* and *Rwc2* should be 105 and 115, respectively. However, because *Rwc1* and *Rwc2* are varied together during the sensitivity analysis, it is assumed that both must be increased to their maximum for a maximum V(3). Therefore, both are increased by 15%.

The purpose of the technique is to reduce the number of simulations. For a more accurate worst-case analysis, you should first perform a worst-case analysis with VARY LOT, manually adjust the nominal model parameter values according to the results, then perform another analysis with VARY DEV specified.

Gaussian distributions

Parameters using Gaussian distributions are changed by 3σ (three times sigma) for the worst-case analysis.

YMAX collating function

This may result in maximizing or minimizing the output variable value over the entire range of the sweep. This collating function is useful when you know the direction in which the maximum deviation occurs.

The purpose of the YMAX collating function is often misunderstood. This function does not try to maximize the deviation of the output variable value from nominal. Depending on whether HI or LO is specified, it tries to maximize or minimize the output variable value itself at the point where maximum deviation occurred during sensitivity analysis.

RELTOL

During the sensitivity analysis, each parameter is varied (multiplied) by $1 + \text{RELTOL}$ where RELTOL is specified in a .OPTIONS statement, or defaults to 0.001.

Sensitivity analysis

The sensitivity analysis results are printed in the output file (.OUT). For each varied parameter, the percent change in the collating function and the sweep variable value at which the collating function was measured are given. The parameters are listed in *worst output* order; for example, the collating function was its worst when the first parameter printed in the list was varied.

When you use the YMAX collating function, the output file also lists mean deviation and sigma values. These are based on the changes in the output variable from nominal at every sweep point in every sensitivity run.

Manual optimization

You can use worst-case analysis to perform *manual optimization* with PSpice A/D. The monotonicity condition is usually met if the parameters have a very limited range.

Performing worst-case analysis with tight tolerances on the parameters produces sensitivity and worst-case results (in the output file). You can use these to decide how the parameters should be varied to achieve the desired response. You can then make adjustments to the nominal values in the circuit file, and perform the worst-case analysis again for a new set of gradients.

Parametric sweeps (.STEP), like the one performed in the circuit file shown in Figure 87, can be used to augment this procedure.

Monte Carlo analysis

Monte Carlo (.MC) analysis may be helpful when worst-case analysis cannot be used. Monte Carlo analysis can often be used to verify or improve on worst-case analysis results. Monte Carlo analysis randomly selects possible parameter values, which can be thought of as randomly selecting points in the *parameter space*. The worst-case analysis assumes that the worst results occur somewhere on the surface of this space, where parameters (to which the output is sensitive) are at one of their extreme values.

If this is not true, the Monte Carlo analysis may find a point at which the results are worse. To try this, replace .WC in the circuit file with .MC <#runs>, where <#runs> is the number of simulations you want to perform. More runs provide higher confidence results. The Monte Carlo summary in the output file lists the runs in decreasing order of collating function value.

To save disk space, do not specify any OUTPUT options.

Next, add the following option to the .MC statement, and simulate again.

```
OUTPUT LIST RUNS <worst_run#>
```

This performs only two simulations: the nominal and the worst Monte Carlo run. The parameter values used during the worst run are written to the output file, and the results of both simulations are saved.

Using Monte Carlo analysis with YMAX is a good way to obtain a conservative guess at the maximum possible deviation from nominal, since worst-case analysis usually cannot provide this information.

Digital simulation

14

Chapter overview

This chapter describes how to set up a digital simulation analysis and includes the following sections:

- [What is digital simulation? on page 14-410](#)
- [Steps for simulating digital circuits on page 14-410](#)
- [Concepts you need to understand on page 14-411](#)
- [Defining a digital stimulus on page 14-413](#)
- [Defining simulation time on page 14-426](#)
- [Adjusting simulation parameters on page 14-427](#)
- [Starting the simulation on page 14-429](#)
- [Analyzing results on page 14-430](#)

What is digital simulation?

Digital simulation is the analysis of logic and timing behavior of digital devices over time. PSpice A/D simulates this behavior during transient analysis. When computing the bias point, PSpice A/D considers the digital devices in addition to any analog devices in the circuit.

See [Tracking timing violations and hazards on page 14-435](#) for information about persistent hazards, and for descriptions of the warning messages.

PSpice A/D performs detailed timing analysis subject to the constraints specified for the devices. For example, flip-flops perform setup checks on the incoming clock and data signals. PSpice A/D reports any timing violations or hazards as messages written to the simulation output file and the waveform data file.

Steps for simulating digital circuits

There are six steps in the development and simulation of digital circuits:

For more information on drawing designs see your *OrCAD Capture User's Guide*. Steps 2 through 6 of this process are covered in this chapter.

- 1 Drawing the design.
- 2 Defining the stimuli.
- 3 Setting the simulation time.
- 4 Adjusting the simulation parameters.
- 5 Starting the simulation.
- 6 Analyzing the results.

Concepts you need to understand

States

When the circuit is in operation, digital nodes take on values or output states shown in Table 1. Each digital state has a *strength* component as well.

Strengths are described in the next section.

Table 1 *Digital states*

This state...	Means this...
0	Low, false, no, off
1	High, true, yes, on
R	Rising (changes from 0 to 1 sometime during the R interval)
F	Falling (changes from 1 to 0 sometime during the F interval)
X	Unknown: may be high, low, intermediate, or unstable
Z	High impedance: may be high, low, intermediate, or unstable

Note *States do not necessarily correspond to a specific, or even stable, voltage. A logical 1 level means only that the voltage is somewhere within the high range for the particular device family. The rising and falling levels only indicate that the voltage crosses the 0–1 threshold at some time during the R or F interval, not that the voltage change follows a particular slope.*

Strengths

For additional information on this topic see [Defining Output Strengths](#) on page 7-262 of **Chapter 7, Digital device modeling**.

When a digital node is driven by more than one device, PSpice A / D determines the correct level of the node. Each output has a *strength* value, and PSpice A / D compares the strengths of the outputs driving the node. The *strongest* driver determines the resulting level of the node. If outputs of the same strength but different levels drive a node, the node's level becomes X.

PSpice A / D supports 64 strengths. The lowest (weakest) strength is called Z. The highest (strongest) strength is called the *forcing* strength. The Z strength (called high impedance) is typically output by disabled tristate gates or open-collector output devices. PSpice A / D reports any nodes of Z strength (at any level) as Z, and reports all other nodes by the designations shown in [Digital states on page 14-411](#).

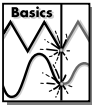
Defining a digital stimulus

A digital stimulus defines input to the digital portions of your circuit, playing a role similar to that played by the independent voltage and current sources for the analog portion of your circuit.

The following table summarizes the digital stimuli provided in the part libraries.

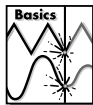
Table 2

If you want to specify the input signal by...	Then use this part...	For this type of digital input...
Using the Stimulus Editor	DIGSTIM	signal or bus stimulus
	DIGCLOCK	clock signal
Defining part properties	STIM1	one-bit stimulus
	STIM4	four-bit stimulus
	STIM8	eight-bit stimulus
	STIM16	sixteen-bit stimulus
	FILESTIM1	one-bit file-based stimulus
	FILESTIM2	two-bit file-based stimulus
	FILESTIM4	four-bit file-based stimulus
	FILESTIM8	eight-bit file-based stimulus
	FILESTIM16	sixteen-bit file-based stimulus
	FILESTIM32	thirty-two-bit file-based stimulus



Note The Stimulus Editor is not included in PSpice A/D Basics.

Note The DIGSTIMn part is not included in PSpice A/D Basics.



Using the DIGSTIMn part

Use the DIGSTIMn stimulus parts to define a stimulus for a net or bus using the Stimulus Editor.

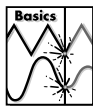
To use the DIGSTIM part

- 1 From Capture's Place menu, choose Part.
- 2 Place and connect the DIGSTIMn stimulus part from SOURCSTM.OLB to a wire or bus in your design.
- 3 Click the stimulus instance to select it.
- 4 From the Edit menu, choose PSpice Stimulus.

This starts the Stimulus Editor. A dialog box appears asking you whether you want to edit the named stimulus.

- 5 Click OK.
- 6 Define stimulus transitions; see [Defining input signals using the Stimulus Editor](#) below.

Note The Stimulus Editor is not included in PSpice A/D Basics.



If you have the Basics package, you can define clock signals using DIGCLOCK. To find out more, see [Using the DIGCLOCK part on page 14-422](#).

Defining input signals using the Stimulus Editor

Defining clock transitions

To create a clock stimulus

- 1 In the Stimulus Editor, select the stimulus that you want to use as a clock.
- 2 From the Stimulus menu, choose Change Type.
- 3 Under Type, choose Clock.
- 4 Click OK.

- 5 Enter values for the clock signal properties as described below.

Table 3

For this property...	Enter this...
Frequency	clock rate
Duty Cycle	percent of high versus low in decimal or integer units
Initial Value	starting value: 0 or 1
Time Delay	time after simulation begins when the clock stimulus takes effect

- 6 From the File menu, choose Save.

To change clock properties

- 1 In the Stimulus Editor, do one of the following:
 - Double-click the clock name to the left of the axis.
 - Click the clock name and from the Edit menu, choose properties.
- 2 Modify the clock properties as needed.
- 3 Click OK.

Defining signal transitions

You can do any of the following when defining digital signal transitions:

- Add a transition
- Move a transition
- Edit a transition
- Delete a transition

Note *These operations cannot be applied to a stimulus defined as a clock signal.*

Example: To create a clock signal with a clock rate of 20 MHz, 50% duty cycle, a starting value of 1, and time delay of 5 nsec, set the signal properties as follows:

Frequency = 20Meg
 Duty Cycle = 0.50 (or 50)
 Initial Value = 1
 Time Delay = 5ns

When you select a transition to edit, a red handle appears.

To add a transition

- 1 From the Stimulus Editor's Edit menu, choose Add.
- 2 Select the digital stimulus you want to edit.
- 3 Drag the new transition to its proper location on the waveform.
- 4 If you want to add more transitions, repeat steps [2](#) and [3](#).
- 5 When you finish, right-click to exit the edit mode.

To move a transition

- 1 Click the transition you want to move.
- 2 If needed, use **Shift**+click to select additional transitions on the same signal or different signals.
- 3 Reposition the transition (or transitions) by dragging.

Note *If you press **Shift** while dragging, then all selected transitions move by the same amount.*

To edit a transition

- 1 Do one of the following:
 - Select the transition you want to edit and from the Edit menu, choose Properties.
 - Double-click the transition you want to edit.
- 2 In the Stimulus properties dialog box, edit the timing and value of the transition.
- 3 Click OK.

To delete a transition

- 1 Click the transition you want to delete.
- 2 If needed, press **Shift**+click to select additional transitions on the same signal or different signals.
- 3 From the Edit menu, choose Delete.

Defining bus transitions

There are three steps for creating a bus:

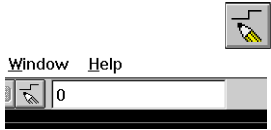
- 1 Creating the digital bus stimulus.
- 2 Introducing transitions.
- 3 Optionally defining the radix for bus values.

These steps are described in detail in the following procedures.

To create a digital bus stimulus

- 1 From the Stimulus menu, choose New.
- 2 In the Digital frame, choose Bus.
- 3 If needed, change the bus width from its default value of 8 bits. To do this, in the Width text box, type a different integer.
- 4 Click OK.

During any interval, the bits on the bus lines represent a value from zero through $(2^n - 1)$, where n is the number of bus lines. To set bus values, introduce transitions using either of the two methods described below.



Example: 12

Example: +12;H

Example: -12;0

To find out about valid radix values, see page [14-433](#).

To introduce transitions (method one)

- 1 From the Stimulus Editor’s Edit menu, choose Add.
- 2 In the digital value field on the toolbar (just right of the Add button), type a bus value in any of the following ways:

Table 4

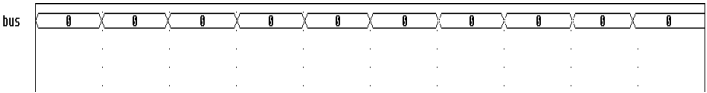
To get this effect...	Type this...
A literal value	<unsigned_number>[:radix]
An increment	+<unsigned_number>[:radix]
A decrement	-<unsigned_number>[:radix]

If you do not enter a radix value, the Stimulus Editor appends the default bus radix.

- 3 Click the waveform where you want the transition added.
- 4 Repeat steps [2](#) and [3](#) as needed.
- 5 When you finish, right-click to exit the editing mode.

To introduce transitions (method two)

- 1 From the Stimulus Editor’s Edit menu, choose Add.
- 2 Place the tip of the pencil-shaped pointer on the waveform, and click to create transitions as shown here:



- 3 When you finish creating transitions, right-click.
- 4 Click the transition at the start (far left) of the interval. A small diamond appears over the transition.
- 5 From the Edit menu, choose Properties.
- 6 In the Transition Type frame, choose Set Value, Increment, or Decrement.
- 7 Do one of the following to specify the bus value:

- Here are some other things that you can do:
- Move a transition left or right by clicking and dragging.
 - Delete a transition by selecting it and then, from the Edit menu, choosing Delete (or by pressing **Delete**).
 - Select more than one transition by holding down **Shift** while clicking.

- In the Value text box, type a value.
 - Select one of these defaults from the list: 0, All bits 1, X (Unknown), or Z (High impedance).
- 8 Click OK.
- 9 Repeat steps 4 through 8 for each transition.

To set the default bus radix

- 1 From the Tools menu, choose Options.
- 2 In the Bus Display Defaults frame, from the Radix list, select the radix you want as default.

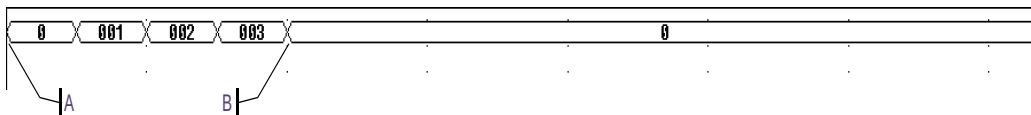
Table 5

Select this radix...	To show values in this notation...
Binary	base 2
Octal	base 8
Decimal	base 10
Hexadecimal	base 16

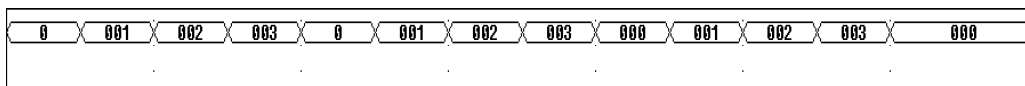
- 3 Click OK.

Adding loops

Suppose you have a stimulus that looks like this:



and you want to create a stimulus that consists of three consecutive occurrences of the sequence that starts at A and ends at B:



You can do this by using a standard text editor to edit a stimulus library file. Within this file is a sequence of transitions that produces the original waveform. With a text editor you can modify the stimulus definition so it repeats itself.

To add a loop

- 1 In the Stimulus Editor, save and close the stimulus file.
- 2 In a standard text editor (such as Notepad), open the stimulus file.
- 3 Enter the path for the stimulus file and click OK. (For example: c:\<path>\mydesign.STL.)
- 4 Find the set of consecutive lines comprising the sequence that you want to repeat.

Each relevant line begins with the time of the transition and ends with a value or change in value.

- 5 Before these lines, insert a line that uses this syntax:

+ Repeat for *n_times*

where *n_times* is one of the following:

- A positive integer representing the number of repetitions.
- The keyword FOREVER, which means repeat this sequence for an unlimited number of times (like a clock signal).

To find out more about the syntax of the stimulus commands used in the stimulus file, refer to the online *OrCAD PSpice A/D Reference Manual*.

Given the example shown on page [14-420](#), if you wanted to repeat the sequence shown from point A to point B three times, then you would modify the stimulus file as shown here (added lines are in bold):

```
+ Repeat for 3
+ +0s 000000000
+ 250us INCR BY 000000001
+ 500us 000000010
+ 750us INCR BY 000000001
+ 1ms 000000000
+ Endrepeat
```


- 6 Below these lines, insert a line that uses this syntax:
+ Endrepeat
- 7 From the File menu, choose Save.

For information on how to define a clock signal using the Stimulus Editor with the DIGSTIMn part, see [Defining clock transitions on page 14-414](#).

Using the DIGCLOCK part

The DIGCLOCK part allows you to define a clock signal by using the part’s properties.

To define a clock signal using DIGCLOCK

- 1 From Capture’s Place menu, choose Part.
- 2 Place and connect a DIGCLOCK part.
- 3 Double-click the part instance.
- 4 Define the properties as described below.

Table 6

For this property...	Specify this...
DELAY	Time before the first transition of the clock
ONTIME	Time in high state for each period
OFFTIME	Time in low state for each period
STARTVAL	Low state of clock (default:0)
OPPVAL	High state of clock (default: 1)

Using STIM1, STIM4, STIM8 and STIM16 parts

The STIMn parts have a single pin for connection. STIM1 is used for driving a single net. STIM4, STIM8 and STIM16 drive buses that are 4, 8 and 16 bits wide, respectively. The properties for all of these parts are the same as those shown in [Table 7](#) below.

Table 7 *STIMn part properties*

Property	Description
WIDTH	Number of output signals (nodes).
FORMAT	Sequence of digits defining the number of signals corresponding to a digit in any <i><value></i> term appearing in a <i>COMMANDn</i> property definition. Each digit must be either 1, 3, or 4 (binary, octal, hexadecimal, respectively); the sum of all digits in <i>FORMAT</i> must equal <i>WIDTH</i> .
IO_MODEL	I/O model describing the stimulus' driving characteristics.
IO_LEVEL	Interface subcircuit selection from one of the four analog / digital subcircuits provided with the part's I/O model.
DIG_PWR	Digital power pin used by the interface subcircuit.
DIG_GND	Digital ground pin used by the interface subcircuit.
TIMESTEP	Number of seconds per clock cycle or step.
COMMAND1- COMMAND16	Stimulus transition specification statements including time / value pairs, labels, and conditional constructs.

When placed, you must connect each part to the wire or bus of the corresponding radix. Generally, you only need to modify the *FORMAT*, *TIMESTEP*, and *COMMANDn* properties.

Typically, each *COMMANDn* property contains only one command line. It is possible to enter more than one command line per property by placing \n+ between command lines in a given definition. (The *n* must be lower case and no spaces between characters; spaces may precede or follow the entire key sequence.)

Refer to the online *OrCAD PSpice A/D Reference Manual* for information about command line syntax.

Refer to the online *OrCAD PSpice A/D Reference Manual* for more information about creating digital stimulus specifications and files.

Using the FILESTIMn parts

The FILESTIMn parts have a single pin for connection to the rest of the circuit. FILESTIM1 is used for driving a single net. FILESTIM2, FILESTIM4, FILESTIM8, FILESTIM16 and FILESTIM32 drive buses that are 2, 4, 8, 16 and 32 bits wide, respectively. You must define the digital stimulus specification in an external file. Using this technique, stimulus definitions can be created from scratch or extracted with little modification from another simulation’s output file.

Table 8 lists the properties of the FILESTIMn parts. The IO_MODEL, IO_LEVEL, and PSPICEDEFAULTNET properties describing this part’s I/O characteristics are provided with default values that rarely need modification. However, you must define the FILENAME property with the name of the external file containing the digital stimulus specification.

The SIGNAME property specifies the name of the signal inside the stimulus file which becomes the output from the FILESTIMn part. If left undefined, the name of the connected net (generally a labeled wire) determines which signal is used.

Table 8 FILESTIMn part properties

Property	Description
FILENAME	Nname of file containing the stimulus specification
SIGNAME	Name of output signal

Table 8 *FILESTIMn part properties*

Property	Description
IO_MODEL	I/O model describing the stimulus' driving characteristics
IO_LEVEL	Interface subcircuit selection from one of the four AtoD or DtoA subcircuits provided with the part's I/O model
PSPICEDEFAULTNET	Hidden digital power and ground pins used by the interface subcircuit. Name of the default net to use.

For example, a FILESTIMn part can be used to reset a counter, which could appear as shown in Figure 93 below.

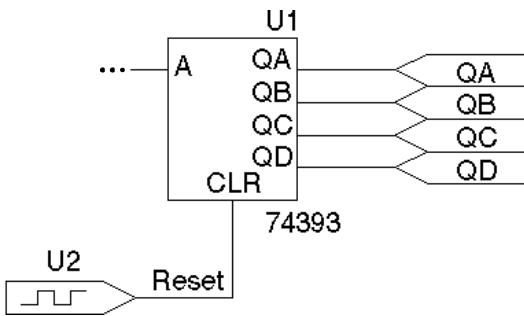


Figure 93 *FILESTIM1 used on a schematic page.*

In this case, the FILESTIM1 part instance, U2, generates a reset signal to the CLR pin of the 74393 counter.

To set up the U2 stimulus

The following steps set up the U2 stimulus so that the 74393 counter is cleared after 40 nsec have elapsed in a transient analysis.

- 1 Create a stimulus file named RESET.STM that contains the following lines:

Reset

A blank line is required between the signal name list and the first transition.

```
0ns 1
40ns 0
```

The header line contains the names of all signals described in the file. In this case, there is only one: Reset.

The remaining lines are the state transitions output for the signals named in the header. In this case, the Reset signal remains at state 1 until 40nsec have elapsed, at which time it drops to state 0.

- 2 Associate this file with the digital stimulus instance, U2, by setting U2's FILENAME property to RESET.STM.
- 3 Define the signal named Reset in RESET.STM as the output of U2 by setting U2's SIGNAME property to Reset. Since the labeled wire connecting U2 with the 74393 counter is also named Reset, it is also acceptable to leave SIGNAME undefined.
- 4 From the PSpice menu, choose Edit Simulation Settings.
- 5 Click the Include Files tab and configure RESET.STM as an include file. (Use the Browse button if necessary to locate the file.)

Defining simulation time

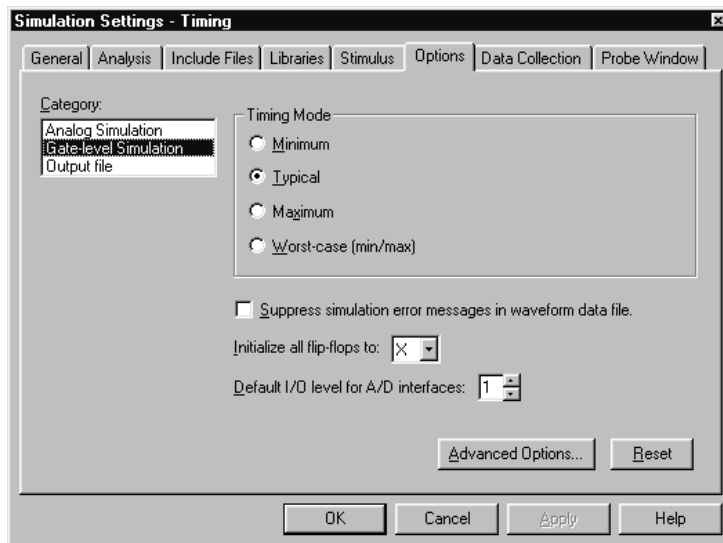
To set up the transient analysis

- 1 From Capture's PSpice menu, choose New Simulation Profile.
- 2 Enter a name for the new simulation profile.
- 3 Click OK.
- 4 In the Analysis Type list box on the Analysis tab, select Time Domain (Transient).

- 5 In the Run to Time text box, type the duration of the transient analysis.
- 6 Click OK.

Adjusting simulation parameters

Use the Options tab of the Simulation Settings dialog box to adjust the simulation behavior of your circuit's digital devices.



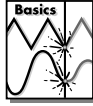
To access the digital settings in the Options tab

- 1 From Capture's PSpice menu, choose Edit Simulation Settings.
- 2 Click the Options tab.
- 3 In the Category list box, select Gate-level simulation.

Each of the dialog box settings is described in the following sections.

For additional options, see [Output control options on page 14-440](#).

Note Propagation delays are not supported in PSpice A/D Basics.



Selecting propagation delays

All digital devices—including primitives and library models—perform simulations using either minimum, typical, maximum or worst-case (min / max) timing characteristics. You can set the delay circuit-wide or on individual device instances.

Circuit-wide propagation delays

You can set these to minimum, typical, maximum or variable within the min / max range for digital worst-case timing simulation on the Options tab of the Simulation Settings dialog box.

To specify the delay level circuit-wide

- 1 From Capture's PSpice menu, choose Edit Simulation Settings.
- 2 Click the Options tab.
- 3 In the Category list box, select Gate-level simulation.

Part instance propagation delays

You can set the propagation delay mode on an individual device, thereby overriding the circuit-wide delay mode.

To override the circuit-wide default on an individual part

- 1 Set the part's MNTYMXDLY property from 1 to 4 where
 - 1 = minimum
 - 2 = typical
 - 3 = maximum
 - 4 = worst-case (min / max)

By default, MNTYMXDLY is set to 0, which tells PSpice A / D to use the circuit-wide value defined in the Options tab.

Initializing flip-flops

To initialize all flip-flops and latches

Select one of the three Flip-flop Initialization choices on the Options tab:

- If set to X, all flip-flops and latches produce an X (unknown state) until explicitly set or cleared, or until a known state is clocked in.
- If set to 0, all such devices are cleared.
- If set to 1, all such devices are preset.

Refer to the online *OrCAD PSpice A/D Reference Manual* for more information about flip-flops and latches.

Note *The X initialization is the safest setting, since many devices do not power up to a known state. However, the 0 and 1 settings are useful in situations where the initial state of the flip-flop is unimportant to the function of the circuit, such as a toggle flip-flop in a frequency divider.*

Starting the simulation

To start the simulation

From the PSpice menu, choose Run.

After PSpice A / D completes the simulation, the graphical waveform analyzer starts automatically.



Analyzing results

In effect, the waveform viewer in PSpice A/D is a software oscilloscope. Running PSpice A/D corresponds to building or changing a breadboard, and the waveform viewer corresponds to looking at the breadboard with an oscilloscope.

For a full discussion of how the waveform viewer is used to analyze results, see [Chapter 17, Analyzing waveforms](#).

For detailed information on how to add digital traces, see [Digital trace expressions on page 17-530](#).

PSpice A / D includes a graphical waveform analyzer for simulation results. You can observe and interactively manipulate the waveform data produced by circuit simulation.

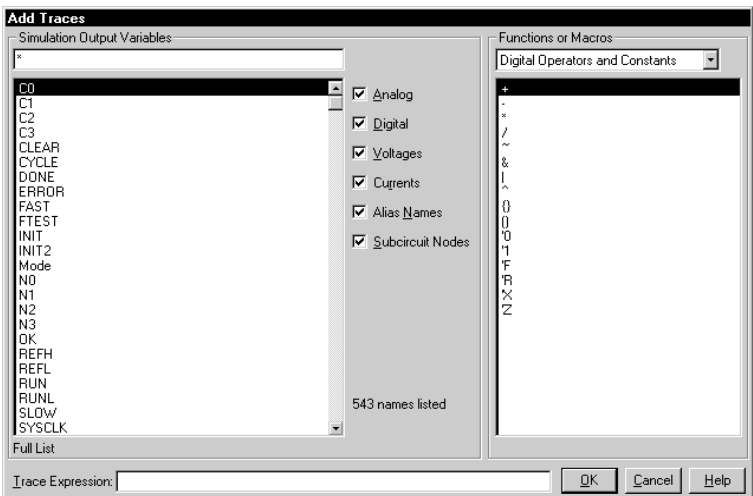
For mixed analog / digital simulations, the waveform analyzer can display analog and digital waveforms simultaneously with a common time base.

PSpice A / D generates two forms of output: the simulation output file and the waveform data file. The calculations and results reported in the simulation output file are like an audit trail of the simulation. However, the graphical analysis of information stored in the data file is a more informative and flexible method for evaluating simulation results.

To display waveforms



1 From the Trace menu, choose Add Trace.



2 Select traces for display:

- In the Simulation Output Variables list, click any waveforms you want to display. Each appears in the Trace Expressions box at the bottom.

- Construct expressions by selecting operators, functions and /or macros from the Functions or Macros list, and output variables in the Simulation Output Variables list.
- You can also type trace expression directly into the Trace Expression text box. A typical set of entries might be:

IN1 IN2 Q1 Q2

Use spaces or commas to separate the output variables you place in the Trace Expressions list.

3 Click OK.

Waveforms for the selected output variables appear.

Adding digital signals to a plot

When defining digital trace expressions, you can include any combination of digital signals, buses, signal constants, bus constants, digital operators, macros and the Time sweep variable.

The following rules apply:

- An arithmetic or logical operation between two bus operands results in a bus value that is wide enough to contain the result.
- An arithmetic or logical operation between a bus operand and a signal operand results in a bus value.

The syntax for expressing a digital output variable or expression is:

digital_output_variable[; *display_name*]

or

digital_expression[; *display_name*]

Table 9

This placeholder...	Means this...
<i>digital_output_variable</i>	output variable from the Simulation Output Variable list (Digital check box selected)
<i>digital_expression</i>	expression using digital output variables and operators
<i>display_name</i> (optional)	text string (name) to label the signal on the plot, instead of using the default output variable notation

To add a digital trace expression

- 1 In the Add Traces dialog box, make sure you select the Digital check box.
- 2 Do one of the following:
 - In the Simulation Output Variables list, click the signal you want to display.
 - In the Trace Expression text box, create a digital expression by either typing the expression, or by selecting digital output variables from the Simulation Output Variables list and digital operators from the Digital Operators and Functions list.
- 3 If you want to label a signal with a name that is different from the output variable:
 - a Click in the Trace Expression text box after the last character in the signal name.
 - b Type *;display_name* where *display_name* is the name of the label.

Example: U2 : Y ; OUT1
where U2:Y is the output variable. On the plot, the signal is labeled OUT1.

Adding buses to a waveform plot

You can evaluate and display a set of up to 32 signals as a bus *even if the selected signals were not originally a bus*. This is done by following the same procedure already given for adding digital signals to the plot. However, when adding a bus, be sure to enclose the list of signals in braces: { }.

```
{ Q3 Q2 Q1 Q2 }
```

The complete syntax is as follows:

```
{signal_list}[, [display_name][, radix]]
```

or

```
{bus_prefix[msb:lsb]}[, [display_name][, radix]]
```

Table 10

This placeholder...	Means this...
<i>signal_list</i>	comma- or space-separated list of up to 32 digital node names, in sequence from high order to low order
<i>bus_prefix[msb:lsb]</i>	alternate way to express up to 32 signals in the bus
<i>display_name</i> (optional)	text string (name) to label the bus on the plot, instead of using the default output variable notation
<i>radix</i> (optional)	numbering system in which to display bus values

To change the radix without changing the display name, be sure to include two consecutive semicolons.

Example :

```
{A3,A2,A1,A0};;radix
```

Valid entries for *radix* are shown in the following table.

Table 11

For this numbering system...	Use this notation...
Binary (base 2)	B
Decimal (base 10)	D
Hexadecimal (base 16)	H or X
Octal (base 8)	O (the letter)

Examples:

`{Q2, Q1, Q0};A;0` specifies a 3-bit bus whose most significant bit is Q2. PSpice A/D labels the plot A, and values appear in octal notation.

`{a3, a2, a1, a0};;d` specifies a 4-bit bus. On the plot, values appear in decimal notation. Since no display name is specified, PSpice A/D uses the signal list as a label.

`{a[3:0]}` is equivalent to
`{a3, a2, a1, a0}`

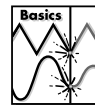
To add a bus expression

- 1 In the Add Traces dialog box, in the Functions and Macros list, choose Digital Operators and Constants.
- 2 Click the { } entry.
- 3 In the Simulation Output Variables list, select the signals in high-order to low-order sequence.
- 4 If you want to label the bus with a name that is different from the default:
 - a Click in the Trace Expression text box after the last character in the bus name.
 - b Type `;display_name` where *display_name* is the name of the label.
- 5 If you want to set the radix to something different from the default:
 - a Click in the Trace Expression text box after the last character in the expression.
 - b Type one of the following where *radix* is a value from the table on page [14-433](#):
 - If you specified a *display_name*, then type `;radix`.
 - If you did not specify a *display_name*, then type `;;radix` (two semicolons preceding the radix value).

Tracking timing violations and hazards

When there are problems with your design, such as setup / hold violations, pulse-width violations, or worst-case timing hazards, PSpice A / D saves messages to the simulation output file or data file. You can select messages and have the associated waveforms and detailed message text automatically appear.

PSpice A / D can also detect persistent hazards that may have a potential effect on a primary circuit output or on the internal state of the design.



Note Timing violations and hazards are not supported in PSpice A/D Basics.

The messaging feature is discussed further in [Tracking digital simulation messages](#) on page **17-517** of [Chapter 17, Analyzing waveforms](#).

Persistent hazards

Digital problems are usually either timing violations or timing hazards. Timing violations include SETUP, HOLD and minimum pulse WIDTH violations of component specifications. This type of violation may produce a change in the state behavior of the design, and potentially in the answer. However, the effects of many of these errors are short-lived and do not influence the final circuit results.

For example, consider an asynchronous data change on the input to flip-flop FF1 in Figure 94 below. The data change is too close to the clock edge e1, resulting in a SETUP violation. In a hardware implementation, the output of FF1 may or may not change. However, some designs are not sensitive to this individual missed data because the next clock edge (e2 in this example) latches the data. The designer must judge the significance of timing errors, accounting for the overall behavior of the design.

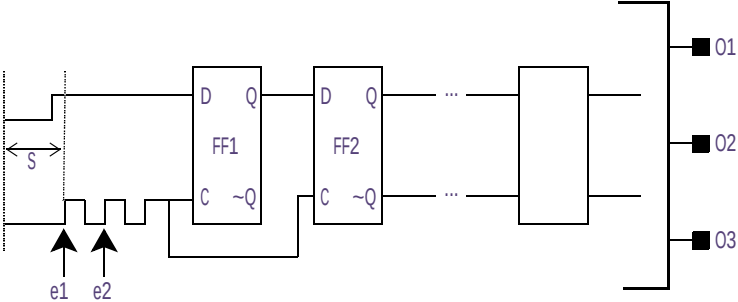


Figure 94 *Circuit with a timing error*

Timing hazards are most easily identified by simulating a design in worst-case timing mode, usually close to its critical timing limits. Under such conditions, PSpice A / D reports conditions such as **AMBIGUITY CONVERGENCE** hazards. Again, these may or may not pose a problem to the operation of the design.

However, there are identifiable cases that cause major problems. An example of a major problem is shown in Figure 95 below. Due to the simultaneous arrival of two timing ambiguities (having unrelated origins, therefore nothing in common) at the inputs to gate G1, PSpice A / D reports the occurrence as an **AMBIGUITY CONVERGENCE** hazard. This means that the output of G1 may glitch.

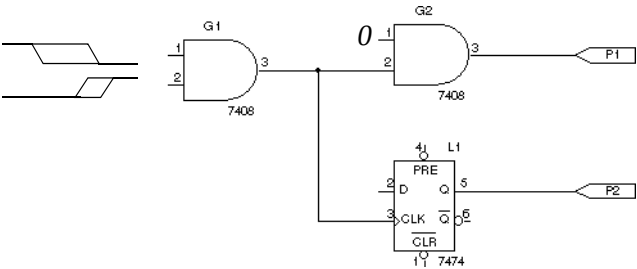


Figure 95 *Circuit with a timing ambiguity hazard*

Note that the output fans out to two devices, G2 and L1. The effects of a glitch on G1 in this case do not reach the

circuit output P1, because that path is not sensitized (since the other input to G2 is held LO and thus blocks the symptom). However, because G1's output is also used to clock latch L1, the effects of a glitch could result in visibly incorrect behavior on output P2. This is an example of a persistent hazard.

A persistent hazard is a timing violation or hazard that has a potential effect on a primary (external) circuit output or on the internal state (stored state or memory elements) of the design. For the design to be considered reliable, you must correct such timing hazards.

PSpice A/D fully distinguishes between state uncertainty and time uncertainty. When a hazard occurs, PSpice A/D propagates hazard origin information along with the machine state through all digital devices. When a hazard propagates to a state-storage device primitive (JKFF, DFF, SRFF, DLTCH, RAM), PSpice A/D reports a PERSISTENT HAZARD.

Simulation condition messages

PSpice A/D produces warning messages in various situations, such as those that originate from the digital CONSTRAINT devices monitoring timing relationships of digital nodes. These messages are directed to the simulation output file and / or to the waveform data file. Options are available for controlling where and how many of these messages are generated, as summarized later in this section.

Table 12 below summarizes the simulation message types, with a brief description of their meaning. Currently, the messages supported are specific to digital device timing violations and hazards.

Table 12 Simulation condition messages—timing violations

Message type	Severity level	Meaning
SETUP	WARNING	Minimum time required for a data signal to be stable <i>prior</i> to the assertion of a clock was not met.
HOLD	WARNING	Minimum time required for a data signal to be stable <i>after</i> the assertion of a clock was not met.
RELEASE	WARNING	Minimum time required for a signal that has gone inactive (usually a control such as CLEAR) to remain inactive before the asserting clock edge was not met.
WIDTH	WARNING	Minimum pulse width specification for a signal was not satisfied; that is, a pulse that was too narrow was observed on the node.
FREQUENCY	WARNING	Minimum or maximum frequency specification for a signal was not satisfied. Minimum frequency violations indicate that the period of the measured signal is too long, while maximum frequency violations describe signals changing too rapidly.
GENERAL	INFO	Boolean expression described within the GENERAL constraint checker was evaluated and produced a <i>true</i> result.

Table 13 *Simulation condition messages—hazards*

Message type	Severity level	Meaning
AMBIGUITY CONVERGENCE	WARNING	Convergence of conflicting rising and falling states (timing ambiguities) arrived at the inputs of a primitive and produced a pulse (glitch) on the output. See Chapter 16, Digital worst-case timing analysis for more information.
CUMULATIVE AMBIGUITY	WARNING	Signal ambiguities are additive, increased by propagation through each level of logic in the circuit. The ambiguities associated with both edges of a pulse increased to the point where they overlapped, which PSpice A/D reports as a cumulative ambiguity hazard. See Chapter 16, Digital worst-case timing analysis for more information.
SUPPRESSED GLITCH	WARNING	Pulse applied to the input of a primitive that is shorter than the active propagation delay was ignored by PSpice A/D; significance depends on the nature of the circuit. There might be a problem either with the stimulus, or with the path delay configuration of the circuit. See Chapter 16, Digital worst-case timing analysis for more information.
NET-STATE CONFLICT	WARNING	Two or more outputs attempted to drive a net to different states, which PSpice A/D reports as an X (unknown) state. This usually results from improper selection of a bus driver's enable inputs.
ZERO-DELAY- OSCILLATION	FATAL	Output of a primitive changed more than 50 times within a single digital time step. PSpice A/D aborted the run.
DIGITAL INPUT VOLTAGE	SERIOUS	Voltage on a digital pin was out of range, which means PSpice A/D used the state with a voltage range closest to the input voltage and continued the simulation.
PERSISTENT HAZARD	SERIOUS	Effects of any of the aforementioned logic hazards were able to propagate to either an external port or to any storage device in the circuit. See Persistent hazards on page 14-435 for more information.

Output control options

Four control options are available for managing the generation of simulation condition messages. These are described in [Table 14](#).

To access these commands, select the Options tab in the Simulation Settings dialog box. You can set NOOUTMSG and NOPRBMMSG by selecting the Output file Category. You can set DIGERRDEFAULT and DIGERRLIMIT by selecting the Gate-level simulation Category and clicking Advanced Options.

Table 14 Simulation message output control options

This option...	Means this...
NOOUTMSG	Suppresses the recording of simulation condition messages in the simulation output file.
NOPRBMMSG	Suppresses the recording of simulation condition messages in the waveform data file.
DIGERRDEFAULT=<n>	Establishes a default limit, <i>n</i> , to the number of condition messages that may be generated by any digital device that has a constraint checker primitive without a local default. If global or local defaults are unspecified, there is no limit.
DIGERRLIMIT=<n>	Establishes an upper limit, <i>n</i> , for the total number of condition messages that may be generated by any digital device. If this limit is exceeded, PSpice A / D aborts the run. By default, the total number of messages is 20.

Severity levels

PSpice A / D assigns one of four severity levels to the messages:

- FATAL
- SERIOUS
- WARNING
- INFO (informational)

FATAL conditions cause PSpice A / D to cancel the simulation. Under all other severity levels, PSpice A / D continues to run. The severity levels are used to filter the classes of messages that are displayed when loading a data file.

Mixed analog / digital simulation

15

Chapter overview

This chapter describes how PSpice A / D runs mixed analog / digital simulations and includes the following sections:

- [Interconnecting analog and digital parts on page 15-444](#)
- [Interface subcircuit selection by PSpice A/D on page 15-445](#)
- [Specifying digital power supplies on page 15-449](#)
- [Interface generation and node names on page 15-454](#)

Interconnecting analog and digital parts

Prior to simulation, netlisting translates the part instances and nets defined in your schematic into parts connected by nodes. The netlist contains a flat view of the circuit (no hierarchy). PSpice A/D extracts the definitions for all parts modeled as subcircuits, viewing parts as a collection of primitive parts and node connections.

The digital primitives that make up a digital part determine the way that PSpice A/D processes an analog/digital interface to that part. Specifically, the I/O model for each digital primitive connected at the interface gives PSpice A/D the necessary information.

PSpice A/D recognizes three types of nodes: analog nodes, digital nodes, and interface nodes. The node type is determined by the types of parts connected to it. If all of the parts connected to a node are analog, then it is an analog node. If all of the parts are digital, then it is a digital node. If there is a combination of analog and digital parts, then it is an interface node.

PSpice A/D automatically breaks interface nodes into one purely analog and one or more digital nodes by inserting one or more analog/digital interface subcircuits.

PSpice A/D also automatically connects a power supply to the interface subcircuit to complete the generation of the interface.

To view simulation results at an analog/digital interface in your schematic using the graphical waveform analyzer:

- Place a marker on the appropriate interface net. The additional nodes created by PSpice A/D remain transparent.
- View results in PSpice A/D by selecting traces from the output variable list (from the Trace menu, choose Add Trace). If you use this approach, note the names PSpice A/D generates for the new nodes.

To find out more, see [Interface generation and node names on page 15-454](#).

Interface subcircuit selection by PSpice A/D

Analog-to-digital (AtoD) and digital-to-analog (DtoA) interface subcircuits handle the translation between analog voltages / impedances and digital states, or vice-versa. The main component of an interface subcircuit is either a PSpice A / D N part (digital input: digital-to-analog) or a PSpice A / D O part (digital output: analog-to-digital).

PSpice A / D N and O parts are neatly packaged into interface subcircuits in the model library. The standard model library shipped with your OrCAD software installation includes interface subcircuits for each of the supported logic families: TTL, CD4000 series CMOS and high-speed CMOS (HC / HCT), ECL 10K, and ECL 100K. This frees you from ever having to define them yourself when using parts in the standard library.

Every digital primitive comprising the subcircuit description of a digital part has an I / O model describing its loading and driving characteristics. The name of the interface subcircuit actually inserted by PSpice A / D is specified by the I / O model of the digital primitive at the interface. The I / O model has parameters for up to four analog-to-digital (AtoD) and four digital-to-analog (DtoA) subcircuit names.

You can choose among four interface levels of subcircuit models, depending on the simulation accuracy you need. In some cases you may need more accurate simulations of the input / output stages of a digital part, while in other cases, a simpler, smaller model is enough.

Digital parts provided in the standard libraries only use interface levels 1 and 2. With the exception of the HC / HCT series (described below), levels 3 and 4 reference the same subcircuits as levels 1 and 2. [Table 15](#) below summarizes the four interface levels.

That's the letter O, not the numeral zero.

If you are creating custom digital parts in technologies other than those provided in the standard model library, you may need to create your own interface subcircuits.

The difference between levels 1 and 2 only occurs in the AtoD interfaces, described below. In all cases, the level 1 DtoA interface is the same as the level 2 DtoA interface.

Table 15 Interface subcircuit models

Level	Subcircuits	Definition
1	AtoD1 / DtoA1	AtoD generates intermediate R, F, and X levels
2	AtoD2 / DtoA2	AtoD does not generate intermediate R, F, and X levels
3	AtoD3 / DtoA3	(same as level 1)
4	AtoD4 / DtoA4	(same as level 2)

The elaborate model is noticeably slower than the simple model, so you should only use it if you are using a power supply level other than 5.0 volts.

The OrCAD libraries provide two different DtoA models in the HC / HCT series: the *simple* model and the *elaborate* model. You can use the simple model by specifying level 1 or 2, the elaborate model by specifying level 3 or 4.

The HC / HCT level 1 and 2 DtoA models produce accurate I-V curves given a fixed power supply of 5.0 volts and a temperature of 25°C. The level 3 and 4 DtoA models produce accurate I-V curves over the acceptable range of power supply voltages (2-6 volts), and they include temperature derating.

Level 1 interface

The level 1 AtoD interface generates intermediate logic levels (R, F, X) between the voltage ranges VILMAX and VIHMIN (specific voltages depend on the technology you are using). A steadily rising voltage on the input of the AtoD will transition from 0 to R at VILMAX and from R to 1 at VIHMIN. The F level is output for steadily falling voltages in a similar manner. The X level is produced if the input voltage starts in the threshold region or doubles back into a previously crossed threshold.

Level 1 (the default) strictly maps logic levels onto the changing input voltage. The exact switching voltage is assumed to be anywhere between VILMAX and VIHMIN due to temperature or power supply variations. Thus, it provides more accurate, less optimistic results.

This behavior may not be appropriate when the input rise and fall times are long, or when the input voltage never leaves the threshold region. If this is the case, you may want to use the level 2 interface.

Level 2 interface

The level 2 AtoD interface transitions directly from 0 to 1 and 1 to 0 without passing through intermediate R, F, or X levels. An exact switching voltage is assumed (again, the specific voltage depends on the technology you are using). It provides a more optimistic, and therefore less accurate, response than level 1. Level 2's behavior is appropriate when the input voltage oscillates around the threshold voltage.

You can avoid simulations that get bogged down with the greater detail of R, F, and X states around these oscillations. You may want to specify level 2 on only those parts for which this behavior is critical to a successful simulation. This is described in [Setting the default A/D interface](#) below.

Setting the default A/D interface

For mixed-signal simulation, you can select the AtoD and DtoA interface level circuit-wide and on individual part instances.

- To select the default interface level circuit-wide, select one of the four Default A / D interfaces in the Digital Setup dialog. Part instances whose `IO_LEVEL` property is set to 0 will use this value.
- You can override the circuit-wide default on an individual part by specifying an `IO_LEVEL` property from 1 to 4, where:
 - 1: AtoD1 and DtoA1 (default)
 - 2: AtoD2 and DtoA2
 - 3: AtoD3 and DtoA3
 - 4: AtoD4 and DtoA4

For example, you can tell the simulator to use the level 2 interface subcircuits for a 7400 part by setting the `IO_LEVEL` property to 2. All other part instances continue to use the circuit-wide setting. By default, `IO_LEVEL` is set to 0, which tells the simulator to use the circuit-wide level defined in the digital portion (DC Sweep analysis) of the Simulation Settings dialog box.

Specifying digital power supplies

Digital power supplies are used to power interface subcircuits that are automatically created by PSpice A/D when simulating analog/digital interfaces. They are specified as follows:

- PSpice A/D can instantiate them automatically.
- You can create your own digital power supplies and place them in your design.

When using parts from the standard libraries in your design, you can usually have PSpice A/D automatically create the necessary digital power supply.

Because digital power supplies are used only by analog/digital interface subcircuits, digital power supplies are not needed for digital-only designs. OrCAD recommends avoiding placing a power supply to a digital-only design because it may increase simulation time and memory usage.

If you use custom digital parts created in technologies other than those provided in the standard model library, you may need to create your own digital power supplies.

Default power supply selection by PSpice A/D

When PSpice A/D encounters an analog/digital interface, it creates the appropriate interface subcircuit and power supply according to the I/O model referenced by the digital part. The I/O model is specific to the digital part's logic family. The power supply provides reference or drive voltage for the analog side of the interface.

By default, PSpice A/D inserts one power supply subcircuit for every logic family in which a digital primitive is involved with an analog/digital interface. These power supply subcircuits create the digital power and ground nodes that are the defaults for all parts in that family. If multiple digital primitives from the same logic family are involved with analog/digital interfaces, one instance of the power supply subcircuit is created with all primitives connected to the power supply nodes.

Table 16 summarizes the default node names and values. For instance, TTL power supplies have a default value of 5.0 volts at analog / digital interfaces.

Table 16 *Default digital power/ground pin connections*

Logic family	Digital power / ground pin properties	Default digital power / ground nodes
TTL	PSPICEDEFAULTNET (PWR)	\$G_DPWR (5.0 volts)
	PSPICEDEFAULTNET (GND)	\$G_DGND (0 volts)
CD4000	PSPICEDEFAULTNET (VDD)	\$G_CD4000_VDD (5 volts)
	PSPICEDEFAULTNET (VSS)	\$G_CD4000_VSS (0 volts)
ECL 10K	PSPICEDEFAULTNET (VEE)	\$G_ECL_10K_VEE (-5.2 volts)
	PSPICEDEFAULTNET (VCC1)	\$G_ECL_10K_VCC1 (0 volts)
	PSPICEDEFAULTNET (VCC2)	\$G_ECL_10K_VCC2 (0 volts)
ECL 100K	PSPICEDEFAULTNET (VEE)	\$G_ECL_100K_VEE (-4.5 volts)
	PSPICEDEFAULTNET (VCC1)	\$G_ECL_100K_VCC1 (0 volts)
	PSPICEDEFAULTNET (VCC2)	\$G_ECL_100K_VCC2 (0 volts)

The default I/O models and power supply subcircuits are found in DIG_IO.LIB. The four default power supplies provided in the model library are DIGIFPWR (TTL), CD4000_PWR (CD4000 series CMOS), ECL_10K_PWR (ECL 10K), and ECL_100K_PWR (ECL 100K).

The PSPICEDEFAULTNET pin properties have the same default values as the digital power and ground nodes created by the default power supply. These node assignments are passed from the part instance to the digital primitives describing its behavior, connecting any digital primitive affected by an analog connection to the correct power supply.

Creating custom digital power supplies

When creating custom power supplies, you can refer to the power supply definitions in DIG_IO.LIB for examples of power supply subcircuit definitions.

Each digital part model has optional digital power and ground nodes that you can use to specify custom power supplies. To do this, use one of the digital power supplies listed in **Table 17** below in your design and redefine the digital power supply nodes.

Table 17 *Digital power supply parts in SPECIAL.OLB*

Part type (PSpice A/D X model)	Part name
CD4000 power supply	CD4000_PWR
TTL power supply	DIGIFPWR
ECL 10K power supply	ECL_10K_PWR
ECL 100K power supply	ECL_100K_PWR

The properties relevant to creating custom power supplies are shown in [Table 18](#).

Table 18 *Digital power supply properties*

Part name	Property	Description
CD4000_PWR	VOLTAGE	CD4000 series CMOS power supply voltage
	PSPICEDEFAULTNET	CD4000 series CMOS hidden power supply pins for VDD and VSS
DIGIFPWR	VOLTAGE	TTL power supply voltage
	PSPICEDEFAULTNET	TTL hidden power (PWR) and ground (GND) pins
ECL_10K_PWR	VEE	ECL power supply voltages
ECL_100K_PWR	VCC1	
	VCC2	ECL hidden power supply pins for VEE, VCC1 and VCC2
	PSPICEDEFAULTNET	

Note This procedure applies to all logic families.

To create a custom digital power supply

- 1 Place the appropriate power supply part listed in [Table 17](#) in your design (by logic family).
- 2 Rename the power supply power and ground pins (PSPICEDEFAULTNET properties).
- 3 Reset the power supply power and ground voltages as required.
- 4 For any digital part instance that uses the power supply, set its appropriate PSPICEDEFAULTNET pin properties to the power and ground pins created by the secondary power supply.

Overriding CD4000 power supply voltage throughout a design

Designs using CD4000 parts often require power supply voltages other than the default 5.0 volts supplied by the standard CD4000_PWR power supply part. If needed, you can override the power supply voltage for all CD4000 parts in a design.

The default power supply nodes used by CD4000 parts are named \$G_CD4000_VDD and \$G_CD4000_VSS as created by the power supply subcircuit CD4000_PWR. This supply defaults to 5.0 volts. You can override the voltage across these two nodes by defining values for the parameters named CD4000_VDD and CD4000_VSS that are referenced by the CD4000_PWR subcircuit definition.

To change the CD4000_PWR power supply to 12 volts, referenced to ground:

- 1 Place an instance of the PARAM pseudopart from SPECIAL.OLB.
- 2 Create a new PARAM property as follows:

`CD4000_VDD = 12.0V`

DC4000_VSS is left at its default of 0 volts.

If the reference voltage also needs to be reset, the same method can be used to define the CD4000_VSS parameter

by setting this property of the same PARAM instance. For example, if you want the supplies to go between -5 volts and +5 volts (a difference of 10 volts), set CD4000_VSS to -5V and CD4000_VDD to +10V; as a result, CD4000_VDD is 10 volts above CD4000_VSS, or +5 volts.

Creating a secondary CD4000, TTL, or ECL power supply

Designs using CD4000, TTL, or ECL parts may require power supply voltages in addition to the default 5.0 volts supplied by the standard CD4000_PWR power supply part.

To create a secondary power supply for any one of the CD4000, TTL, or ECL technologies, you must place the appropriate power supply part and create user-defined nodes with a new voltage value.

Designs with TTL and ECL parts rarely require secondary power supplies. If needed, however, you can use this procedure to add a secondary power supply for TTL and ECL parts.

To create and use a secondary CD4000 power supply with nodes MY_VDD and MY_VSS and a voltage of 3.5 volts:

- 1 Place the CD4000_PWR power supply and modify the appropriate pin properties as follows:

```
VOLTAGE = 3.5V
PSPICEDEFAULTNET = MY_VDD
PSPICEDEFAULTNET = MY_VSS
```

- 2 Select a CD4000 part in the schematic to which the new power supply should apply, then change the appropriate pin properties as follows:

```
PSPICEDEFAULTNET = MY_VDD
PSPICEDEFAULTNET = MY_VSS
```

Interface generation and node names

The majority of the interface generation process involves PSpice A/D determining whether analog and digital primitives are connected, and if so, inserting an interface subcircuit for each digital connection. This turns the interface node into a purely analog node, which now connects to the analog terminal of the interface subcircuit. To complete the original connection, PSpice A/D creates a new digital node between the digital terminal of the interface subcircuit and the digital primitive.

These node names are used in the output variables in the list of viewable traces when you choose Add Trace from the Trace menu.

Because PSpice A/D must create new digital nodes, it must give them unique names. Name generation follows these rules:

- The analog node retains the name of the original interface node—either the labeled wire name in the design, or the node name automatically generated for an unlabeled wire.
- Each new digital node name consists of the labeled wire name in the design or the node name automatically generated for an unlabeled wire, appended with \$AtoD or \$DtoA. If the node is attached to more than one digital part, the second digital node is appended with \$AtoD2 or \$DtoA2, and so on.

Figure 96 below shows a fragment of a mixed analog/digital circuit before and after the interface subcircuits have been added. The wires labeled 1 and 2 in the schematic representation are the interface nets connecting analog and digital parts. These translate to interface nodes, which are processed by PSpice A/D to create the circuit fragment shown in the PSpice A/D representation.

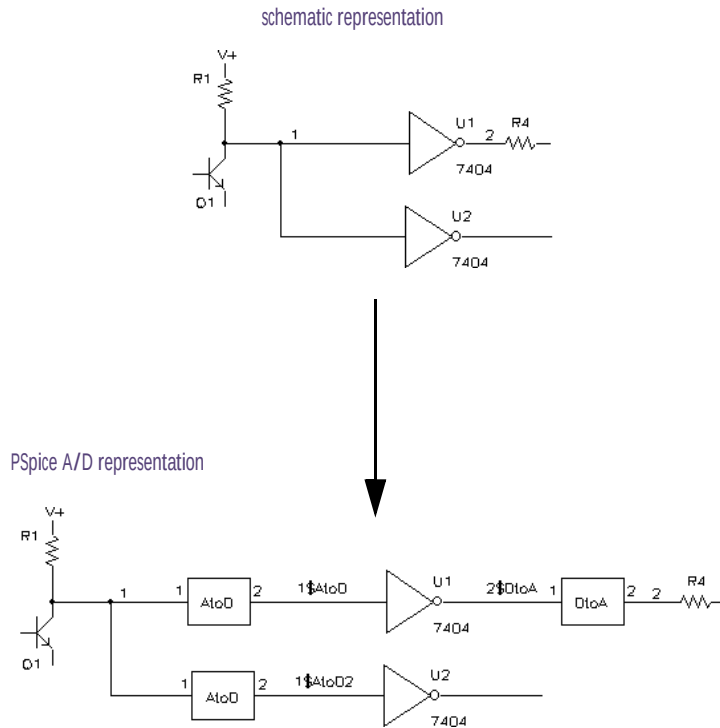


Figure 96 *Mixed analog/digital circuit before and after interface generation.*

After interface generation, node 1 is a purely analog node, connecting the resistor, transistor, and the analog inputs of both AtoD subcircuits. Node 2 is also a purely analog node, connecting the resistor and the analog output of the DtoA interface. You can see that PSpice A/D inserted two new digital nodes, 1\$AtoD and 1\$AtoD2, which connect the outputs of the AtoD interfaces to the inverter inputs. It also created one digital node, 2\$DtoA, to connect the output of U1 to the digital input of the DtoA interface.

The interface subcircuits PSpice A/D automatically generates are listed in the simulation output file under the section named Generated AtoD and DtoA Interfaces. For the example in Figure 96, this section would appear in the simulation output file as shown in Figure 97 below.

```

**** Generated AtoD and DtoA Interfaces ****
*
* Analog/Digital interface for node 1
*
* Moving X1.U1:.A from analog node 1 to new digital node *
1$AtoD
X$1_AtoD1 1 1$AtoD $G_DPWR $G_DGND AtoD_STD
+      PARAMS: CAPACITANCE= 0
* Moving X2.U1:.A from analog node 1 to new digital node *
1$AtoD2
X$1_AtoD2 1 1$AtoD $G_DPWR $G_DGND AtoD_STD
+      PARAMS: CAPACITANCE= 0
*
* Analog/Digital interface for node 2
*
** Moving X1.U1.Y from analog node 2 to new digital node *
2$DtoA
X$2_DtoA1 2$DtoA 2 $G_DPWR $G_DGND DtoA_STD
+      PARAMS: DRVL=0 DRVH=0 CAPACITANCE=0
*
* Analog/Digital interface power supply subcircuit
*
X$DIGIFPWR 0 DIGIFPWR

.END ;(end of AtoD and DtoA interfaces)

```

Figure 97 *Simulation output for mixed analog/digital circuit.*

The lines that begin with “Moving...from analog node” indicate the new digital node names that were generated. Below each of these are the interface subcircuit calls inserted by PSpice A/D.

In this example, the subcircuits named AtoD_STD and DtoA_STD are obtained from the I/O model that is referenced by the inverter primitive inside the subcircuit describing the 7404 part. The CAPACITANCE, DRVL (low-level driving resistance), and DRVH (high-level driving resistance) subcircuit parameter values come from the same I/O model.

After the interface subcircuit calls, PSpice A/D inserts one or more interface power supply subcircuits. The subcircuit name is specified in the I/O model for the digital primitive at the interface. In this example, PSpice A/D inserted DIGIFPWR, which is the power supply subcircuit used by all TTL models in the model library.

DIGIFPWR creates the global nodes \$G_DPWR and \$G_DGND, which are the default nodes used by each TTL part.

Digital worst-case timing analysis

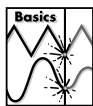
16

Chapter overview

This chapter deals with worst-case timing analysis and includes the following sections:

- [Digital worst-case timing on page 16-458](#)
- [Starting worst-case timing analysis on page 16-459](#)
- [Simulator representation of timing ambiguity on page 16-459](#)
- [Propagation of timing ambiguity on page 16-461](#)
- [Identification of timing hazards on page 16-462](#)
- [Convergence hazard on page 16-462](#)
- [Critical hazard on page 16-463](#)
- [Cumulative ambiguity hazard on page 16-464](#)
- [Reconvergence hazard on page 16-466](#)
- [Glitch suppression due to inertial delay on page 16-468](#)
- [Methodology on page 16-469](#)

Note Digital worst-case timing is not supported in PSpice A/D Basics.



Compared to analog worst-case analysis

Digital worst-case timing simulation is different from analog worst-case analysis in several ways. Analog worst-case analysis is implemented as a sensitivity analysis for each parameter which has a tolerance, followed by a projected worst-case simulation with each parameter set to its minimum or maximum value. This type of analysis is general since any type of variation caused by any type of parameter tolerance can be studied. But it is time consuming since a separate simulation is required for each parameter. This does not always produce true worst-case results, since the algorithm assumes that the sensitivity is monotonic over the tolerance range.

The techniques used for digital worst-case timing simulation are not compatible with analog worst-case analysis. It is therefore not possible to do combined analog/ digital worst-case analysis and simulation and get the correct results. PSpice A/D allows digital worst-case simulation of mixed-signal and all-digital circuits; any analog sections are simulated with nominal values.

Systems containing embedded analog-within-digital sections do not give accurate worst-case results; they may be optimistic or pessimistic. This is because analog simulation can not model a signal that will change voltage at an unknown point within some time interval.

Digital worst-case timing

Manufacturers of electronic components generally specify component parameters (such as propagation delays in the case of logic devices) as having tolerances. These are expressed as either an operating range, or as a spread around a typical operating point. The designer then has some indication of how much deviation from typical one might expect for any of these particular component delay values.

Realizing that any two (or more) instances of a particular type of component may have propagation delay values anywhere within the published range, designers are faced with the problem of ensuring that their products are fully functional when they are built with combinations of components having delay specifications that fall (perhaps randomly) anywhere within this range.

Historically, this has been done by making simulation runs using minimum (MIN), typical (TYP), and maximum (MAX) delays, and verifying that the product design is functional at these extremes. But, while this is useful to some extent, it does not uncover circuit design problems that occur only with certain combinations of slow and fast parts. True worst-case simulation, as provided by PSpice A / D, does just that.

Other tools called timing verifiers are sometimes used in the design process to identify problems that are indigenous to circuit definition. They yield analyses that are inherently pattern-independent and often pessimistic in that they tend to find more problems than will truly exist. In fact, they do not consider the actual usage of the circuit under an applied stimulus.

PSpice A / D does not provide this type of static timing verification. Worst-case timing simulation, as provided by PSpice A / D, is a pattern-dependent mechanism that allows a designer to locate timing problems subject to the constraints of a specific applied stimulus.

Starting worst-case timing analysis

To set up a worst-case timing analysis:

- 1 In the Simulation Settings dialog box, click the Options tab.
- 2 Under Category, select Gate-level Simulation.
- 3 In the Timing Mode frame, check Worst-case (min / max)
- 4 In the Initialize all flip-flops drop-down list, select X.
- 5 Set the Default I/O level for A/D interfaces to 1.
- 6 Click OK.
- 7 Start the simulation as described in [Starting a simulation on page 8-299](#).

See [Setting up analyses on page 8-289](#) for a description of the Simulation Settings dialog box.

Simulator representation of timing ambiguity

PSpice A/D uses the five-valued state representation {0,1,R,F,X}, where R and F represent rising and falling transitions, respectively. Any R or F transitions can be thought of as ambiguity regions. Although the starting and final states are known (example: R is a $0 \rightarrow 1$ transition), the exact time of the transition is not known, except to say that it occurs somewhere within the *ambiguity region*.

Timing ambiguities propagate through digital devices via whatever paths are sensitized to the specific transitions involved. This is normal logic behavior. The delay values (MIN, TYP, or MAX) skew the propagation of such signals by whatever amount of propagation delay is associated with each primitive instance.

The ambiguity region is the time interval between the earliest and the latest time that a transition could occur.

When worst-case (MIN / MAX) timing operation is selected, both the MIN and the MAX delay values are used to compute the duration of the timing ambiguity result that represents a primitive's output change.

For example, consider the model of a BUF device in the following figure.

```
U5 BUF $G_DPWR $G_DGND IN1 OUT1 ; BUFFER model
+ T_BUF IO_STD

.MODEL T_BUF UGATE (          ; BUF timing model
+ TPLHMN=15ns TPLHTY=25ns TPLHMX=40ns
+ TPHLMN=12ns TPHLTY=20ns TPHLMX=35ns)
```

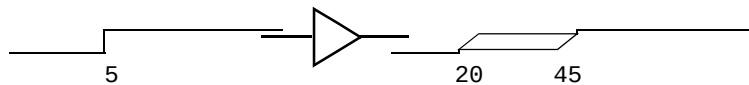


Figure 98 Timing ambiguity example one.

The application of the instantaneous 0-1 transition at 5nsec in this example produces a corresponding output result. Given the delay specifications in the timing model, the output edge occurs at a MIN of 15nsec later and a MAX of 40nsec later. The region of ambiguity for the output response is from 20 to 45nsec (from TPLHMN and TPLHMX values). Similar calculations apply to a 1-0 transition at the input, using TPHLMN and TPHLMX values.

Propagation of timing ambiguity

As signals propagate through the circuit, ambiguity is contributed by each primitive having a nonzero MIN / MAX delay spread. Consider the following example that uses the delay values of the previous BUF model.

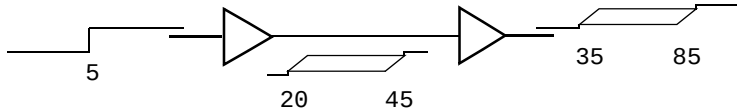


Figure 99 Timing ambiguity example two.

This accumulation of ambiguity may have adverse effects on proper circuit operation. In the following example, consider ambiguity on the data input to a flip-flop.

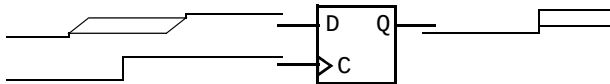


Figure 100 Timing ambiguity example three.

The simulator must predict an X output, because it is not known with any certainty when the data input actually made the 0-1 transition. If the cumulative ambiguity present in the data signal had been less, the 1 state would be latched up correctly.

Figure 101 illustrates the case of unambiguous data change (settled before the clock could transition) being latched up by a clock signal with some ambiguity. The Q output will change, but the time of its transition is a function of both the clock's ambiguity and that contributed by the flip-flop MIN / MAX delays.

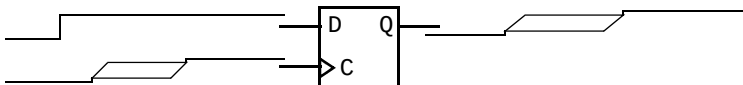


Figure 101 Timing ambiguity example four

Identification of timing hazards

Timing hazard is the term applied to situations where the response of a device cannot be properly predicted because of uncertainty in the arrival times of signals applied to its inputs.

For example, Figure 102 below shows the following signal transitions (0-1, 1-0) being applied to the AND gate.

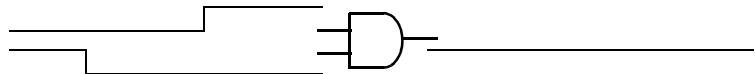


Figure 102 *Timing hazard example.*

The state of the output does not (and should not) change, since at no time do both input states qualify the gate, and the arrival times of the transitions are known.

Convergence hazard

In cases where there are ambiguities associated with the signal transitions 0-R-1 and 1-F-0—which have a certain amount of overlap—it is no longer certain which of the transitions happens first.

The output could pulse (0-1-0) at some point because the input states may qualify the gate. On the other hand, the output could remain stable at the 0 state. This is called a *convergence hazard* because the reason for the glitch occurrence is the convergence of the conflicting ambiguities at two primitive inputs.

Gate primitives (including LOGICEXP primitives) that are presented with simultaneous opposing R and F levels may produce a pulse of the form 0-R-0 or 1-F-1.

For example, a two-input AND gate with the inputs shown in Figure 103 below, produces the output shown.

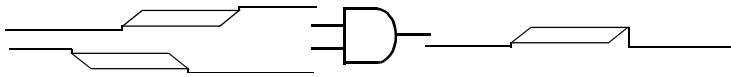


Figure 103 Convergence hazard example.

This output (0-R-0) should be interpreted as *a possible single pulse, no longer than the duration of the R level*. The actual device's output may or may not change, depending on the transition times of the inputs.

Note that other types of primitives, such as flip-flops, may produce an X instead of an R-0 or F-1 in response to a convergence hazard.

Critical hazard

It is important to note that the glitch predicted could propagate through the circuit and may cause incorrect operation. If the glitch from a timing hazard becomes latched up in an internal state (such as flip-flop or ram), or if it causes an incorrect state to be latched up, it is called a *critical hazard* because it definitely causes incorrect operation.

Otherwise, the hazard may pose no problem. Figure 104 below shows the same case as above, driving the data input to a latch.

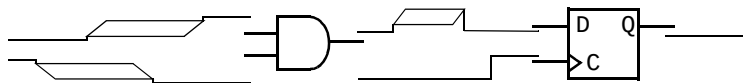


Figure 104 Critical hazard example.

As long as the glitch always occurs well before the leading edge of the clock input, it will not cause a problem.

See [Glitch suppression due to inertial delay on page 16-468](#).

Cumulative ambiguity hazard

In worst-case mode, simple signal propagation through the network will result in a buildup of ambiguity along the paths between synchronization points. The cumulative ambiguity is illustrated in Figure 105.

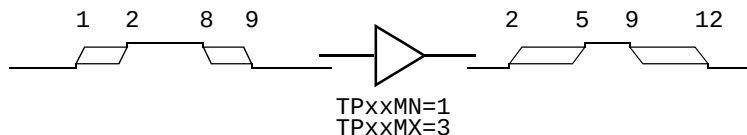


Figure 105 Cumulative ambiguity hazard example one.

The rising and falling transitions applied to the input of the buffer have a 1nsec ambiguity. The delay specifications of the buffer indicate that an additional 2nsec of ambiguity is added to each edge as they propagate through the device. Notice that the duration of the stable state 1 has diminished due to the accumulation of ambiguity.

Figure 106 shows the effects of additional cumulative ambiguity.

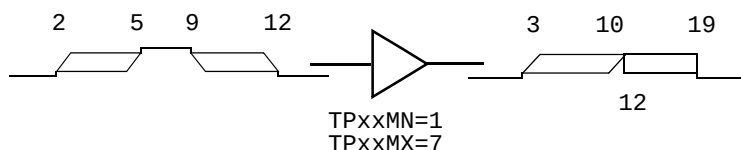


Figure 106 Cumulative ambiguity hazard example two.

The X result is predicted here because the ambiguity of the rising edge propagating through the device has increased to the point where it will overlap the later falling edge ambiguity. Specifically, the rising edge should occur between 3nsec and 12nsec; but, the subsequent falling edge applied to the input predicts that the output starts to fall at 10nsec. This situation is called a *cumulative ambiguity hazard*.

Another cause of cumulative ambiguity hazard involves circuits with asynchronous feedback. The simulation of such circuits under worst-case timing constraints yields an overly pessimistic result due to the unbounded accumulation of ambiguity in the feedback path. A simple example of this effect is shown in Figure 107.

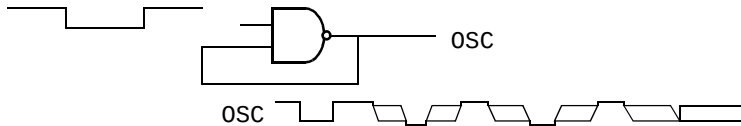


Figure 107 Cumulative ambiguity hazard example three.

Due to the accumulation of ambiguity in the loop, the output signal will eventually become X, because the ambiguities of the rising and falling edges overlap. However, in the hardware implementation of this circuit, a continuous phase shift with respect to absolute time is what will actually occur (assuming normal deviations of the rise and fall delays from the nominal values).

If this signal were used to clock another circuit, it would become the reference and the effects of the phase shift could be ignored. You can do this by setting the NAND gate's model parameter, `MNTYMXDLY=2` to utilize typical delay values for that one gate only (all other devices continue to operate in worst-case mode).

Reconvergence hazard

PSpice A / D recognizes situations where signals having a common origin reconverge on the inputs of a single device. In Figure 108, the relative timing relationship between the two paths (U2, U3) is important.

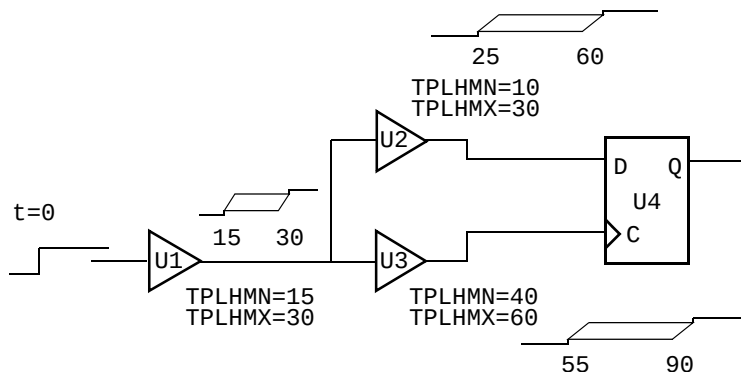


Figure 108 Reconvergence hazard example one.

Given the delay values shown, it is impossible for the clock to change before the data input, since the MAX delay of the U2 path is smaller than the MIN delay of the U3 path. In other words, the overlap of the two ambiguity regions could not actually occur.

PSpice A / D recognizes this type of situation and does not produce the overly pessimistic result of latching an X state into the Q-output of U4. This factors out the 15 nsec of common ambiguity attributed to U1 from the U2 and U3 signals (see Figure 109).

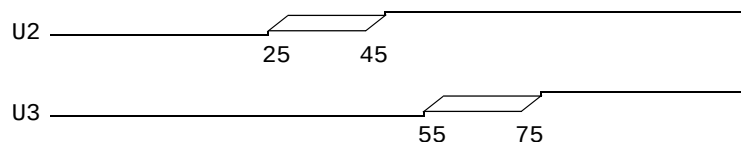


Figure 109 Reconvergence hazard example two.

The result in Figure 109 does not represent what is actually propagated at U2 and U3, but is a computation to determine that U2 must be stable at the earliest time U3 might change. This is why an X level should not be latched.

In the event that discounting the common ambiguity does not preclude latching the X (or, in the case of simple gates, predicting a glitch), the situation is called a *reconvergence hazard*. This is the same as a convergence hazard with the conflicting signal ambiguities having a common origin.

To use digital worst-case simulation effectively, find the areas of the circuit where signal timing is most critical and use constraint checkers where appropriate. These devices identify specific timing violations, taking into account the actual signal ambiguities (resulting from the elements' MIN / MAX delay characteristics). The most common areas of concern include:

- data / clock signal relationships
- clock pulse-widths
- bus arbitration timing

Signal ambiguities that converge (or reconverge) on wired nets or buses with multiple drivers may also produce hazards in a manner similar to the behavior of logic gates. In such cases, PSpice A / D factors out any common ambiguity before reporting the existence of a hazard condition.

The use of constraint checkers to validate signal behavior and interaction in these areas of your design identifies timing problems early in the design process. Otherwise, a timing-related failure is only identifiable when the circuit does not produce the expected simulation results.

See the online *OrCAD PSpice A/D Reference Manual* for more information about digital primitives.

See [Methodology on page 16-469](#) for information on digital worst-case timing simulation methodology.

Glitch suppression due to inertial delay

Signal propagation through digital primitives is performed by the simulator subject to constraints such as the primitive's function, delay parameter values, and the frequency of the applied stimulus. These constraints are applied both in the context of a normal, well-behaved stimulus, and a stimulus that represents timing hazards.

Timing hazards may not necessarily result in the prediction of an X or glitch output from a primitive; these are due to the delay characteristics of the primitive, which PSpice A/D models using the concept of inertial delay.

A device presented with a combination of rising and falling input transitions (assuming no other dominant inputs) produces a glitch due to the uncertainty of the arrival times of the transitions (see Figure 110).

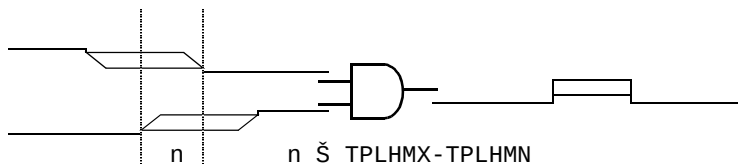


Figure 110 Glitch suppression example one.

However, when the duration of the conflicting input stimulus is less than the inertial delay of the device, the X result is automatically suppressed by the simulator because it would be overly pessimistic (see Figure 111).

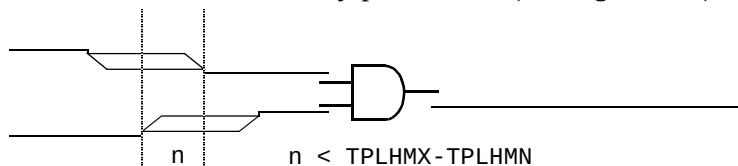


Figure 111 Glitch suppression example two.

In the analysis of reconvergent fanout cases (where common ambiguity is recognized), it is possible that conflicting signal ambiguities may still overlap at the inputs to a primitive, even after factoring out the

commonality. In such cases, where the amount of overlap is less than the inertial delay of the device, the prediction of a glitch is also suppressed by the simulator (see Figure 112).

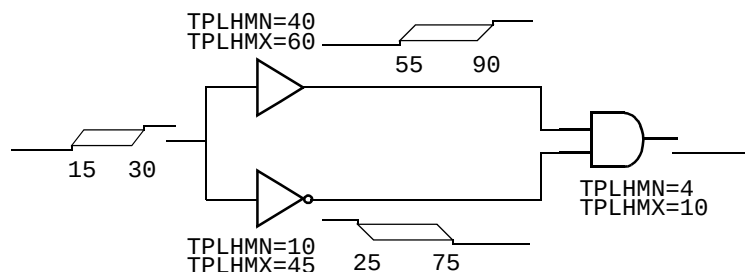


Figure 112 Glitch suppression example three.

In this case, factoring out the 15nsec common ambiguity still results in a 5nsec overlap of conflicting states. The glitch is suppressed, however, because 5nsec is less than $TPLHMX - TPLHMN$ (the computed inertial delay value of the AND gate, 6nsec).

Note Glitch suppression can be overridden by setting the pulse-width rejection threshold parameter ($TPWRT$) in the device's I/O Model.

Methodology

Combining component tolerances and the circuit design's functional response to a specific stimulus presents a challenge. You must make sure that all the finished circuits will operate properly. Well-designed systems have a high degree of immunity from the effects of varying combinations of individual component tolerances.

Digital worst-case timing simulation can help identify design problems, depending upon the nature of the stimulus applied to the design. You can use the simulation of signal propagation through the network to observe the timing relationships among various devices and make adjustments to the design.

Digital worst-case timing simulation does not yield such results without an applied stimulus; it is not a static timing analysis tool. The level of confidence that you

This is not intended to be a comprehensive discussion of the application of digital worst-case timing simulation in the design process. Rather, it is a suggested starting point for understanding the results of your simulation.

For example, if you were designing a digital ADDER circuit, you would probably want to ensure that no timing race conditions existed in the carry logic.

establish for your design's timing-dependent characteristics is directly a function of the applied stimulus.

Generally, the most productive way to define a stimulus is to use functional testing: a stimulus designed to operate the design in a normal manner, exercising all of the important features in combination with a practical set of data.

Your timing simulation methodology should include these key steps:

- Accurate specification of device delay characteristics.
- Functional specification of circuit behavior, including all “don't care” states or conditions.
- A set of stimuli designed to verify the operation of all functions of the design.

One common design verification strategy is stepwise identification of the sections of the design that are to be exercised by particular subsets of the stimulus, followed by verification of the response against the functional specification.

Complete this phase using normal (not worst-case) simulation, with typical delays selected for the elements. The crucial metric here is the state response of the design. Note that (with rare exception) this response consists of defined states and does not include X's.

The second phase of design verification is to use digital worst-case simulation, reapplying the functionally correct stimulus, and comparing the resulting state response to that obtained during normal simulation. Investigate differences at primary observation points (such as circuit outputs and internal state variables)—particularly those due to X states (such as critical hazards)—to determine their cause.

Starting at those points, use the waveform analyzer and the circuit schematic to trace back through the network. Continue until you find the reason for the hazard.

For example, in the case of a convergence or reconvergence hazard, look for conflicting rise/fall inputs. In the case of cumulative ambiguity, look for successive ambiguity regions merging within two edges forming a pulse.

After you identify the appropriate paths and know the relative timing of the paths, you can do either of the following:

- Modify the stimulus (in the case of a simple convergence hazard) to rearrange the relative timing of the signals involved.
- Change one or both of the path delays to rearrange the relative timing, by adding or removing logic, or by substituting component types with components that have different delay characteristics.

In the case of the cumulative ambiguity hazard, the most likely solution is to shorten the path involved. You can do this in either of two ways:

- Add a synchronization point to the logic, such as a flip-flop—or gating the questionable signal with a clock (having well-controlled ambiguity)—before its ambiguity can grow to unmanageable duration.
- Substitute faster components in the path, so that the buildup of ambiguity happens more slowly.

Modifying the stimulus is not generally effective for reconvergent hazards, because the problem is between the source of the reconvergent fanout and the location of the hazard. In this case, discounting the common ambiguity did not preclude the hazard.

Part four

Viewing results

Part four describes the ways to view simulation results.

- **Chapter 17, [Analyzing waveforms](#)**, describes how to perform graphical waveform analysis of simulation results.
- **Chapter 18, [Other output options](#)**, describes the special symbols you can place on your schematic to generate additional information to the PSpice output file, PSpice window, and to digital test vector files.

Analyzing waveforms

17

Chapter overview

This chapter describes how to perform graphical waveform analysis of simulation results in PSpice A/D. This chapter includes the following:

- [Overview of waveform analysis on page 17-476](#)
- [Setting up waveform analysis on page 17-480](#)
- [Viewing waveforms on page 17-483](#)
- [Analog example on page 17-497](#)
- [Mixed analog/digital tutorial on page 17-500](#)
- [User interface features for waveform analysis on page 17-505](#)
- [Tracking digital simulation messages on page 17-517](#)
- [Trace expressions on page 17-519](#)

Overview of waveform analysis

You can use the waveform analysis features of PSpice A/D to visually analyze and interactively manipulate the waveform data produced by circuit simulation.

PSpice A/D uses high-resolution graphics so you can view the results of a simulation both on the screen and in printed form. On the screen, waveforms appear as plots displayed in Probe windows within the PSpice A/D workspace.

In effect, waveform analysis is a software oscilloscope. Performing a PSpice A/D simulation corresponds to building or changing a breadboard, and performing waveform analysis corresponds to looking at the breadboard with an oscilloscope.

With waveform analysis you can:

- View simulation results in multiple Probe windows
- Compare simulation results from multiple circuit designs in a single Probe window
- Display simple voltages, currents, and noise data
- Display complex arithmetic expressions that use the basic measurements
- Display Fourier transforms of voltages and currents, or of arithmetic expressions involving voltages and currents
- For mixed analog / digital simulations, display analog and digital waveforms simultaneously with a common time base
- Add text labels and other annotation symbols for clarification

PSpice A/D generates two forms of output: the simulation output file and the waveform data file. The calculations and results reported in the simulation output file act as an audit trail of the simulation. However, the graphical analysis of information in the waveform data

file is the most informative and flexible method for evaluating simulation results.

Elements of a plot

A single plot consists of the analog (lower) area and the digital (upper) area.

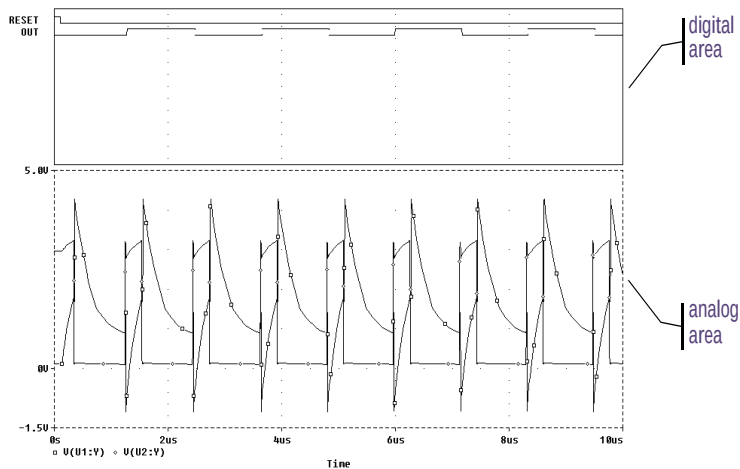


Figure 113 Analog and digital areas of a plot.

You can display multiple plots at a time. If you display only analog waveforms, the entire plot will be an analog area. Likewise, if you display only digital waveforms, the entire plot will be a digital area.

Elements of a Probe window

A Probe window is a separately managed waveform display area. A Probe window can include multiple analog and digital plots. Figure 114 shows two plots displayed together.

Because a Probe window is a window object, you can minimize and maximize windows, or move and scale the windows, within the PSpice A/D workspace. A toolbar can be displayed in the Probe window and applies to the active Probe window.

From the View menu, choose Toolbar to display orhide the toolbar.

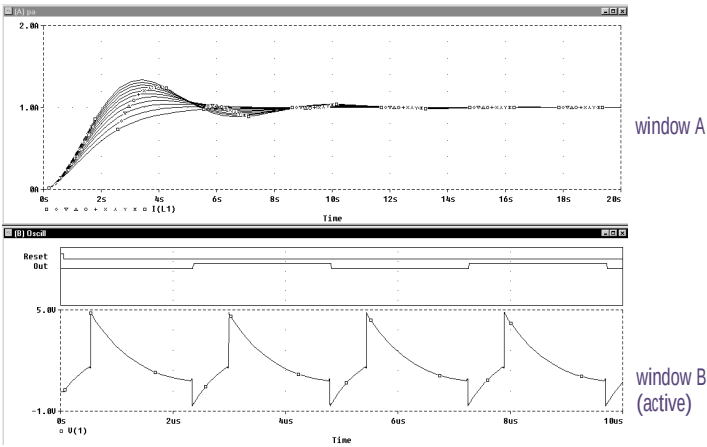


Figure 114 Two Probe windows.

You can display information from one or more waveform data files in one Probe window. After the first file is loaded, load other files into the same Probe window by appending them in PSpice A/D.

Managing multiple Probe windows

You can open any number of Probe windows. Each Probe window is a tab on the worksheet displayed in the middle of the workspace.

The same waveform data file can be displayed in more than one Probe window. You can tile the windows to compare data.

Only one Probe window is active at any given time, identified by a highlighted title bar or a topmost tab. Menu, keyboard, and mouse operations affect only the active Probe window. You can switch to another Probe window by clicking another tab or title bar.

Printing multiple windows

You can print all or selected Probe windows, with up to nine windows on a single page. When you choose Print from the File menu, a list of all open Probe windows appears. Each Probe window is identified by the unique identifier in parentheses in its title bar.

The arrangement of Probe windows on the page can be customized using the Page Setup dialog box. You can print in either portrait (vertical) or landscape (horizontal) orientation. You can also use Print Preview to view all of the Probe windows as they will appear when printed.

Setting up waveform analysis

Setting up colors

You can configure Probe display and print colors in:

- The configuration file, PSPICE.INI
- The Probe Options dialog box

For information on how to use the available colors and color order in a Probe window, see [Configuring trace color schemes on page 17-482](#).

Editing display and print colors in the PSPICE.INI file

In the PSPICE.INI file, you can control the following print and display color settings for Probe windows:

- The colors used to display traces
- The colors used for the Probe window foreground and background
- The order colors are used to display traces
- The number of colors used to display traces

Colors for all items are specified as `<item name>=<color>`. The item names and what they represent are listed in Table 19.

Here are the color names you can specify:

brightcyan	brightblue
brightgreen	brightred
brightmagenta	
brightwhite	
brightyellow	darkblue
darkcyan	darkgray
darkmagenta	darkgreen
darkpink	darkred
lightgreen	
lightblue	lightgray
magenta	mustard
pink	purple
brown	blue
white	black
	yellow

To edit display and print colors in the PSPICE.INI file

Note After editing PSPICE.INI, you must restart PSpice A/D before your changes will take effect.

- 1 In a standard text editor (such as Notepad), open PSPICE.INI. (This file is normally located in the C:\Windows directory.)
- 2 Scroll to the [PROBE DISPLAY COLORS] or [PROBE PRINTER COLORS] section of the file.
- 3 Add or modify a color entry. See [Table 19 on page 17-481](#) for a description of color entries and their default values. Valid item names include:
 - BACKGROUND
 - FOREGROUND

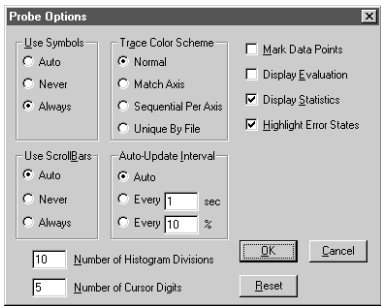
- TRACE_1 through TRACE_12
- 4 If you added or deleted trace number entries, set NUMTRACECOLORS=n to the new number of traces, where n is between 1 and 12. This item represents the number of trace colors displayed on the screen or printed before the color order repeats.
 - 5 Save the file.

Table 19 *Default waveform viewing colors.*

Item Name	Description	Default
BACKGROUND	specifies the color of window background	BLACK
FOREGROUND	specifies the default color for items not explicitly specified	WHITE
TRACE_1	specifies the first color used for trace display	BRIGHTGREEN
TRACE_2	specifies the second color used for trace display	BRIGHTRED
TRACE_3	specifies the third color used for trace display	BRIGHTBLUE
TRACE_4	specifies the fourth color used for trace display	BRIGHTYELLOW
TRACE_5	specifies the fifth color used for trace display	BRIGHTMAGENTA
TRACE_6	specifies the sixth color used for trace display	BRIGHTCYAN

When you want to copy Probe plots to the clipboard and then paste them into a black and white document, choose the Change All Colors to Black option under Foreground in the Copy to Clipboard—Color Filter dialog box (from the Window menu, choose Copy to Clipboard).

For information on what the default available colors and color order are and how to change them, see [Editing display and print colors in the PSPICE.INI file on page 17-480](#).



PSpice A/D saves the selected color scheme for future waveform analyses.

Configuring trace color schemes

In the Probe Options dialog box, you can set options for how the available colors and the color order specified in the PSPICE.INI file are used to display the traces in a Probe window. You can use:

- a different color for each trace
- the same color for all the traces that belong to the same y-axis
- the available colors in sequence for each y-axis
- the same color for all the traces that belong to the same waveform data file

To configure trace color schemes in the Probe Options dialog box

- 1 From the Tools menu, choose Options to display the Probe Options dialog box.
- 2 Under Trace Color Scheme, choose one of the following options:

Table 20

Choose this option...	To do this...
Normal	Use a different color for each trace (for up to 12 traces, depending on the number of colors set in the PSPICE.INI file).
Match Axis	Use the same color for all the traces that belong to the same y-axis. The title of the axis (by default, 1, 2, etc.) is the same color as its traces.
Sequential Per Axis	Use the available colors in sequence for each y-axis.
Unique by File	Use the same color for all the traces in one Probe window that belong to the same waveform data file.

- 3 Click OK.

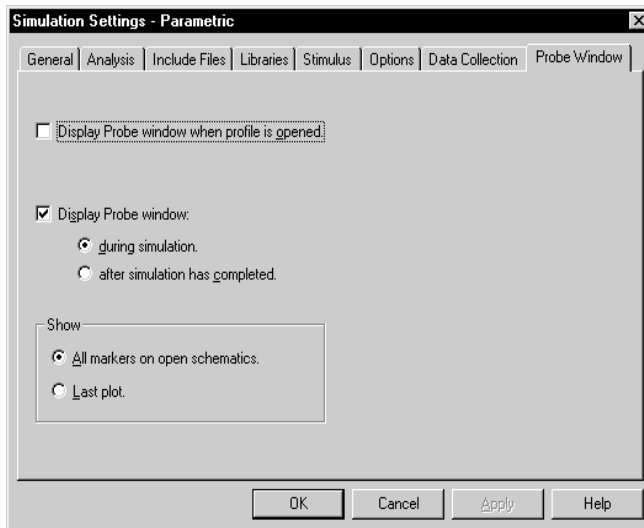
Viewing waveforms

If you are using Capture, you can either view waveforms automatically after you run a simulation, or you can monitor the progress of the simulation as it is running.

Setting up waveform display from Capture

You can configure the way you want to view the waveforms in PSpice A/D by defining display settings in the Probe Window tab in the Simulation Settings dialog box.

You do not need to exit PSpice if you are finished examining the simulation results for one circuit and want to begin a new simulation from within Capture. However, PSpice A/D unloads the old waveform data file for a circuit each time that you run a new simulation of the circuit. After the simulation is complete, the new or updated waveform data file is loaded for viewing.



The display settings in the Probe Window tab are explained in the following table.

Table 21

This setting...	Enables this type of waveform display...
Display Probe window when profile is opened.	Waveforms are displayed only when a .DAT file is opened from within PSpice A / D.
Display Probe window... during simulation.	Waveforms are displayed as the simulation progresses (“marching waveforms”).
Display Probe window... after simulation has completed.	Waveforms are displayed only after the full simulation has completed and all data has been calculated.
Show... all markers on open schematics.	Waveforms are displayed for those nets that have markers attached in the schematic.
Show... last plot.	Waveforms are displayed according to the last display configuration that was used in the Probe window.

Viewing waveforms while simulating

While a simulation is in progress, you can monitor the results for the data section being written by PSpice A / D. This function is only available when the Display Probe window during simulation option is enabled in the Probe Window tab of the Simulation Settings dialog box.

To monitor results during a simulation

- 1 From Capture’s PSpice menu, choose Edit Simulation Settings to display the Simulation Settings dialog box.
- 2 Click the Probe Window tab.
- 3 Select Display Probe window and then click during simulation.
- 4 Click OK to close the Simulation Settings dialog box.

If you open a new Probe window (from the Window menu, choose New Window) while monitoring the data, the new window also starts in monitor mode because it is associated with the same waveform data file.

- 5 From the PSpice menu, choose Run to start the simulation.
One Probe window is displayed in monitor mode.
- 6 Do one of the following to select the waveforms to be monitored:
 - From PSpice's Trace menu, choose Add, and enter one or more trace expressions.
 - From Capture's PSpice menu, point to Markers, then choose and place one or more markers.

The Probe window monitors the waveforms for as long as the most recent data section is being written. After that data section is finished, the window changes to manual mode. To see the full set of runs, you must update the display by using the Add Trace command under the Trace menu.

During a multi-run simulation (such as Monte Carlo, parametric, or temperature), PSpice displays only the data for the most recent run in the Probe window.



or press **Insert**

For more information, see [Using schematic page markers to add traces on page 17-487](#).

Configuring update intervals

You can define the frequency at which PSpice updates the waveform display as follows:

- At fixed time intervals (every n sec)
- According to the percentage of simulation completed (every n %), where n is user-defined

The default setting (Auto) updates traces each time PSpice gets new data from a simulation.

To change the update interval

- 1 From the Tools menu, choose Options.
- 2 In the Auto-Update Interval frame, choose the interval type (sec or %), then type the interval in the text box.

Interacting with waveform analysis during simulation

The functions that change the x-axis domain (that set a new x-axis variable) can not be accessed while the simulation is running. If you have enabled the display of

waveforms during simulation and wish to reconfigure the x-axis settings (as explained below), you must wait until the simulation run has finished.

The following table shows how to enable the functions that change the x-axis domain.

Table 22

	Enable this function...	By doing this...
	Fast Fourier transforms	1 From the Plot menu, choose Axis Settings. 2 In the Processing Options frame, select Fourier.

Pausing a simulation and viewing waveforms

You can pause a simulation to analyze waveforms before the simulation is finished. After you pause the simulation, you can either resume the simulation or end it.

To pause a simulation

- 1

From PSpice’s Simulation menu, choose Pause.
- 2

In the Probe window, view the waveforms generated before you paused the simulation.

- 3 Do one of the following:
 - From the Simulation menu, choose Run to resume the simulation.
 - From the Simulation menu, choose Stop to stop the simulation.

Using schematic page markers to add traces

You can place markers on a schematic page to identify the points where you want to see waveform results displayed. You can place markers:

- Before simulation, to limit results written to the waveform data file and automatically display those traces in PSpice.
- During or after simulation, with PSpice A/D running, to automatically display traces in the active Probe window.

The color of the marker you place is the same as its corresponding waveform analysis trace. If you change the color of the trace, the color of the marker on the schematic page changes accordingly.

The Markers submenu also provides options for controlling the display of marked results in PSpice, after initial marker placement, and during or after simulation.

To place markers on a schematic page

- 1 From Capture's PSpice menu, point to Markers, then choose the marker type you want to place. (Some of the markers are from the Advanced submenu.)

See [Trace expressions on page 17-519](#) for ways to add traces within PSpice A/D.



Table 23

Waveform	Markers menu command	Advanced submenu command
voltage	Voltage Level	not required
voltage differential	Voltage Differential	not required
current	Current Into Pin	not required
digital signal	Voltage Level	not required
dB*	Advanced	db Magnitude of Voltage db Magnitude of Current
phase*	Advanced	Phase of Voltage Phase of Current
group delay*	Advanced	Group Delay of Voltage Group Delay of Current
real*	Advanced	Real Part of Voltage Real Part of Current
imaginary*	Advanced	Imaginary Part of Voltage Imaginary Part of Current

* You can use these markers instead of the built-in functions provided in output variable expressions (see [Table 12 on page 17-528](#)). However, these markers are only available after defining a simulation profile for an AC Sweep / Noise analysis.

The color of the marker is the same as its corresponding waveform analysis trace. If you change the color of the trace, the color of the marker changes accordingly.

- 2 Point to the wires or pins you wish to mark and click to place the chosen markers.
- 3 Right-click and select End Mode to stop placing markers.
- 4 If you have not simulated the circuit yet, from the PSpice menu, choose Run.

To hide or delete marked results

- 1 From Capture’s PSpice menu, point to Markers, then choose one of the following:

Table 24

Choose this option...	To do this...
Hide All	Hide traces in the waveform analysis display for all markers placed on any page or level of the schematic.
Delete All	Remove all markers from the schematic and all corresponding traces from the waveform analysis display.

Limiting waveform data file size

When PSpice A/D performs a simulation, it creates a waveform data file. The size of this file for a transient analysis is roughly equal to:

$(\# \text{ transistors}) \cdot (\# \text{ simulation time points}) \cdot 24 \text{ bytes}$

The size for other analysis types is about 2.5 times smaller. For long runs, especially transient runs, this can generate waveform data files that are several megabytes in size. Even if this does not cause a problem with disk space, large waveform data files take longer to read in and take longer to display traces on the screen.

You can limit waveform data file size by:

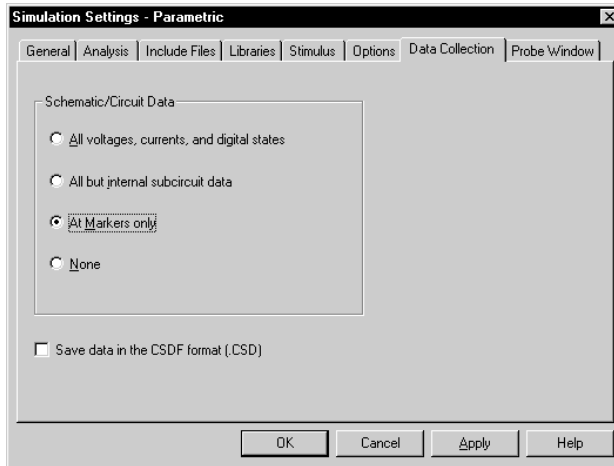
- placing markers on your schematic before simulation and having PSpice A/D restrict the saved data to these markers only
- excluding data for internal subcircuits
- suppressing simulation output

Limiting file size using markers

One reason that waveform data files are large is that, by default, PSpice A/D stores *all net voltages and device currents* for each step (for example, time or frequency points). However, if you have placed markers on your schematic prior to simulation, PSpice A/D saves only the results for the marked wires and pins.

To limit file size using markers

- 1 From Capture's PSpice menu, choose Edit Simulation Settings to display the Simulation Settings dialog box.



- 2 Click the Data Collection tab.
- 3 In the Schematic/ Circuit Data frame, choose At Markers only and click OK.
- 4 From the PSpice menu, point to Markers, then choose the marker type you want to place.
- 5 Point to the wires or pins you wish to mark and click to place the chosen markers.
- 6 Right-click and select End Mode to stop placing markers.
- 7 From the PSpice menu, choose Run to start the simulation.

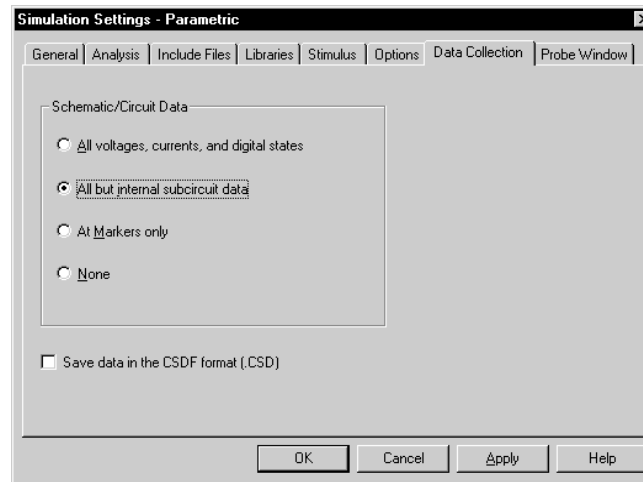
The color of the marker on the schematic page is the same as its corresponding waveform analysis trace. If you change the color of the trace, the color of the marker changes accordingly.

Limiting file size by excluding internal subcircuit data

By default, PSpice A/D saves data for all internal nodes and devices in subcircuit models in a design. You can exclude data for internal subcircuit nodes and devices.

To limit file size by excluding data for internal subcircuits

- 1 From PSpice's Simulation menu, choose Edit Simulation Settings to display the Simulation Settings dialog box.



- 2 Click the Data Collection tab.
- 3 In the Schematic/Circuit Data frame, choose All but internal subcircuit data, then click OK.
- 4 From the PSpice menu, choose Run to start the simulation.

Limiting file size by suppressing the first part of simulation output

Suppressing part of the data run also limits the size of the PSpice A/D output file.

Long transient simulations create large waveform data files because PSpice A/D stores many data points. You can suppress a part of the data from a transient run by setting the simulation analysis to start the output at a time later than 0. This does not affect the transient calculations

themselves—these always start at time 0. This delay only suppresses the output for the first part of the simulation.

To limit file size by suppressing the first part of transient simulation output

- 1 From Capture's PSpice menu, choose Edit Simulation Settings to display the Simulation Settings dialog box.
- 2 Click the Analysis tab.
- 3 From the Analysis type list, select the Time Domain (Transient) option.
- 4 In the Start saving data after text box, type a delay time.
- 5 Click OK to close the Simulation Settings dialog box.
- 6 From the PSpice menu, choose Run to start the simulation.

The simulation begins, but no data is stored until after the delay has elapsed.

Using simulation data from multiple files

You can load simulation data from multiple files into the same Probe window by appending waveform data files.

When more than one waveform data file is loaded, you can add traces using all loaded data, data from only one file, or individual data sections from one or more files.

Appending waveform data files

To append a waveform data file

- 1 In PSpice A/D, from the File menu, choose Append Waveform (.DAT).
- 2 Select a *.DAT file to append, and click OK.



- 3 If the file has multiple sections of data for the selected analysis type, the Available Sections dialog box appears. Do one of the following:
 - Click the sections you want to use.
 - Click the All button to use all sections.
- 4 Click OK.

Adding traces from specific loaded waveform data files

If two or more waveform data files have identical simulation output variables, trace expressions that include those variables generate traces for each file. However, you can specify which waveform data file to use in the trace expression. You can also determine which waveform data file was used to generate a specific trace.



or press **Insert**

The Simulation Output Variables list in the Add Traces dialog box contains the output variables for all loaded waveform data files.

Example: To plot the V(1) output for data section 1 from the second data file loaded, type the following trace expression:

```
V(1)@1@f2
```

You can also use the name of the loaded data file to specify it. For example, to plot the V(1) output for all data sections of a loaded data file, MYFILE.DAT, type the following trace expression:

```
V(1)@"MYFILE.DAT"
```

 **V(Out)**

trace symbols

To add a trace from a specific loaded waveform data file

- 1 In PSpice A/D, from the Trace menu, choose Add Trace to display the Add Traces dialog box.
- 2 In the Trace Expression text box, type an expression using the following syntax:

```
trace_expression@fn
```

where *n* is the numerical order (from left to right) of the waveform data file as it appears in the PSpice title bar, or

```
trace_expression@s@fn
```

where *s* is a specific data section of a specific waveform data file.

- 3 Click OK.

To identify the source file for an individual trace

- 1 In the trace legend, double-click the symbol for the trace you want to identify (Figure 115).

The Section Information dialog box appears, containing the trace name and—if there is more than

Figure 115 Trace legend symbols.

one waveform data file loaded in the plot—the full path for the file from which the trace was generated.

Also listed is information about the simulation that generated the waveform data file and the number of data points used (Figure 116).

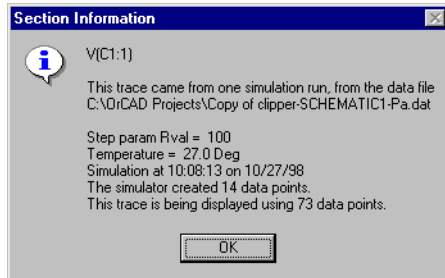


Figure 116 *Section information message box.*

Saving simulation results in ASCII format

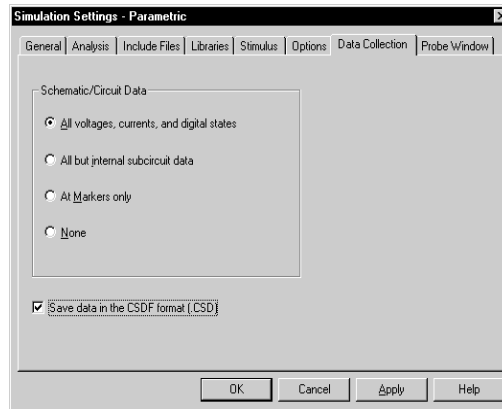
The default waveform data file format is binary. However, you can save the waveform data file in the Common Simulation Data Format (CSDF) instead.

Warning: Data files saved in the CSDF format are two or more times the size of binary files.

When you first open a CSDF data file, PSpice A/D converts it back to the .DAT format. This conversion takes two or more times as long as opening a .DAT file. PSpice A/D saves the new .DAT file for future use.

To save simulation results in ASCII format

- 1 From PSpice's Simulation menu, choose Edit Profile to display the Simulation Settings dialog box.



- 2 Click the Data Collection tab.
- 3 Select Save data in the CSDF format (.CSD).
- 4 Click OK.

PSpice A / D writes simulation results to the waveform data file in ASCII format (as *.CSD instead of *.DAT), following the CSDF convention.

The example project EXAMPLE.OPJ is provided with your OrCAD programs.

When shipped, EXAMPLE.OPJ is set up with multiple analyses. For this example, the AC sweep, DC sweep, Monte Carlo/ worst-case, and small-signal transfer function analyses have been disabled. The specification for each of these disabled analyses remains intact. To run them from Capture in the future, from the PSpice menu, choose Edit Simulation Settings and enable the analyses.

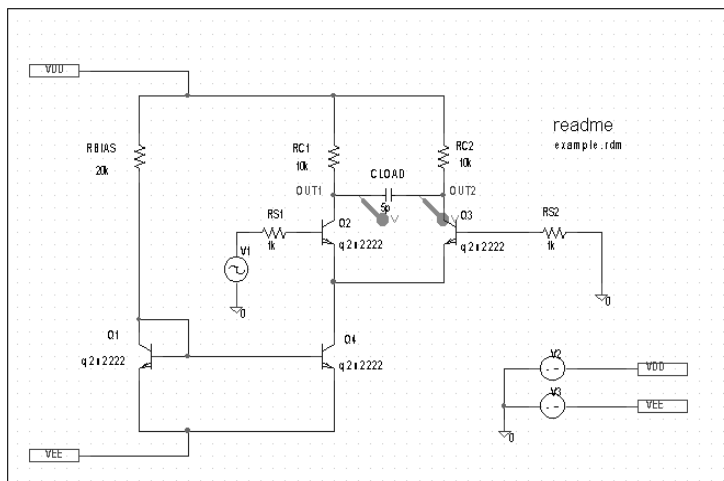


Figure 117 Example schematic *EXAMPLE.OPJ*.

Running the simulation

The simulation is run with the Bias Point Detail, Temperature, and Transient analyses enabled. The temperature analysis is set to 35 degrees. The transient analysis is setup as follows:

Print Step	20ns
Final Time	1000ns
Enable Fourier	selected
Center Frequency	1Meg
Output Vars	V(OUT2)

To start the simulation

- 1 From Capture's File menu, point to Open and choose Project.
- 2 Open the following project in your OrCAD program installation directory:

Note When you run a Fourier analysis using PSpice A/D as specified in this example, PSpice A/D writes the results to the PSpice output file (*.OUT). You can also use Probe windows to display the Fourier transform of any trace expression by using the FFT capability in PSpice. To find out more, refer to PSpice A/D online Help.

\PSPICE\SAMPLES\ANASIM\EXAMPLE\EXAMPLE.OPJ

If PSpice is set to show traces for all markers on startup, you will see the V(OUT1) and V(OUT2) traces when the Probe window displays. To clear these traces from the plot, from the Trace menu, choose Delete All Traces.

- 3 From the PSpice menu, choose Run to start the simulation.

PSpice A/D generates a binary waveform data file containing the results of the simulation. A new Probe window appears with the waveform data file EXAMPLE.DAT already loaded (Figure 118).

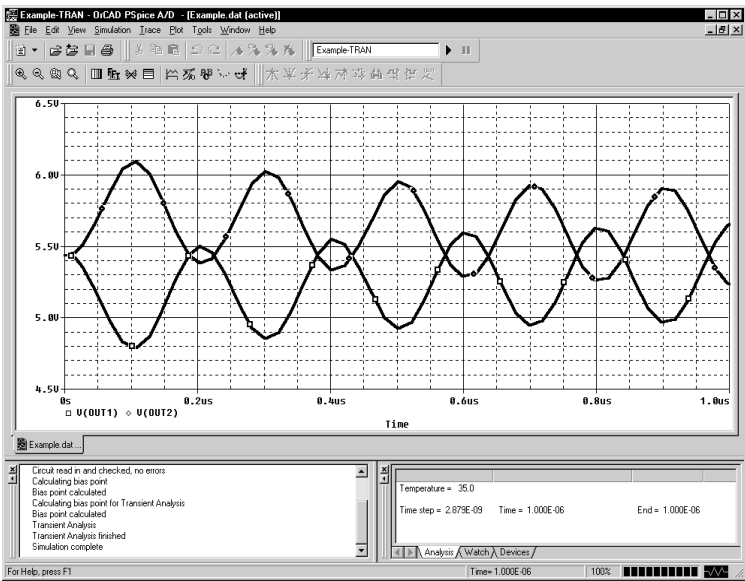


Figure 118 Waveform display for EXAMPLE.DAT.

Because this sample project was set up as a transient analysis type, the data currently loaded are the results of the transient analysis.

Note *In this sample, the voltage markers for OUT1 and OUT2 are already placed in the design. If the markers are not placed prior to simulating, you can display the waveforms later, as explained below in Displaying voltages on nets.*

Displaying voltages on nets

After selected an analysis, voltages on nets and currents into device pins can be displayed in the Probe windows using either schematic markers or output variables (as will be demonstrated in this example).

To display the voltages at the OUT1 and OUT2 nets using output variables

- 1 From the Trace menu, choose Add Trace to display the Add Traces dialog box.

press **Insert**



The Simulation Output Variables frame displays a list of valid output variables.

- 2 Click V(OUT1) and V(OUT2), then click OK. The Probe window should look similar to Figure 118.

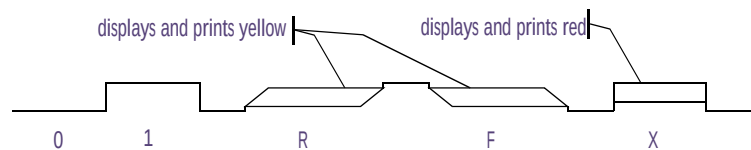
Mixed analog / digital tutorial

In this tutorial, you will use PSpice A/D to simulate a simple, mixed analog / digital circuit. You will then analyze the output by:

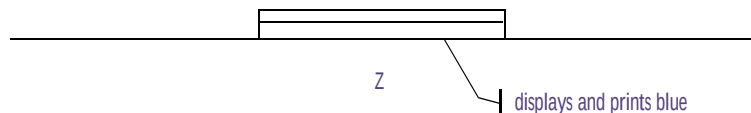
- simultaneously displaying analog and digital traces along a common time axis, and
- displaying digital data values and features unique to mixed analog / digital circuit analysis, such as identification of digital nets inserted by PSpice A/D.

About digital states

All digital states are supported in PSpice A/D. Logic levels appear as shown below.



Nets with the Z strength (at any level) are displayed as a triple line as shown below.



Running the simulation

To run the simulation



- 1 From Capture's PSpice menu, choose Run.

Because the oscillator circuit used here has been run with only a transient analysis, PSpice automatically selects the transient analysis data section from the waveform data file. This means that the Available Selection dialog box is skipped and a Probe window appears immediately.

Analyzing simulation results



or press **Insert**

You can also use aliases to refer to nets. For example, V(U2:A) refers to the same net as V(1).

To view the clock input to the inverter (voltage at net 1)

- 1 From PSpice's Trace menu, choose Add Trace to display the Add Traces dialog box.
- 2 In the Simulation Output Variables list, click V(1) to plot the voltage at net 1.
- 3 Click OK.

To add a second y-axis to avoid analog trace overlap

- 1 From the Plot menu, choose Axis Settings to display the Axis Settings dialog box.

The X Axis tab is active by default.

- a In the Data Range frame, choose User Defined and set the range from 0us to 10us, if this is not already set.
- b In the Scale frame, select Linear, if this is not already set.

- 2 Click the Y Axis tab.

- a In the Data Range frame, choose User Defined and set the range from -5 to 5. This will change the range for the current y-axis.

In the Probe window, double-click the y-axis.

In the Y Axis Settings dialog box, you can change the settings for another y-axis by selecting it from the Y axis Number box.

b Click OK.

- From the Plot menu, choose Add Y Axis.

press **Ctrl** + **Y**

The Probe window display should now look like Figure 120 below.

Note that the V(1) label at the bottom of the plot is preceded by a boxed 1. This indicates that the far-left y-axis applies to the V(1) waveform.

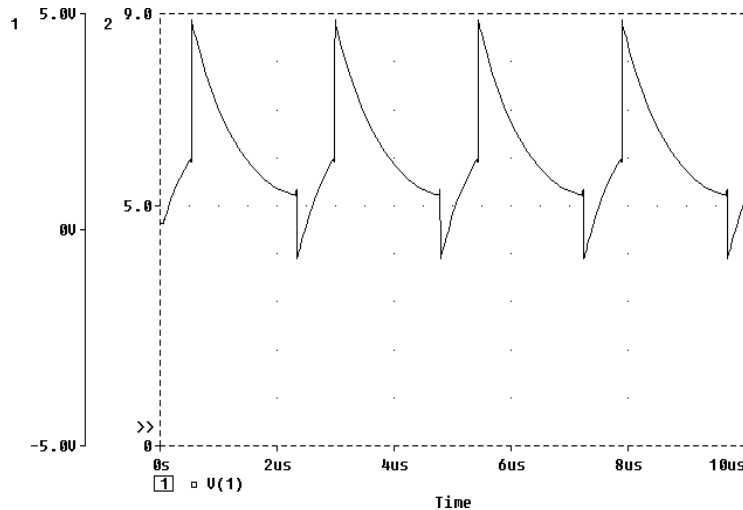


Figure 120 Voltage at net 1 with y-axis added.

To view traces for V(3), RESET, and OUT

- From the Trace menu, choose Add Trace to display the Add Traces dialog box.
- In the Simulation Output Variables list, click V(3), RESET, and OUT.

The trace names appear in the Trace Expression text box.

- Click OK to plot the traces.

The plot displays a digital area above the analog area as shown in Figure 121 below.

You can add up to 75 digital traces to the digital portion of the plot. If you add more traces than can be displayed, PSpice A/D scrolls the traces upwards so you can see the last trace added. A + character in front of the highest or lowest trace name indicates that there are more traces above or below the marked traces.

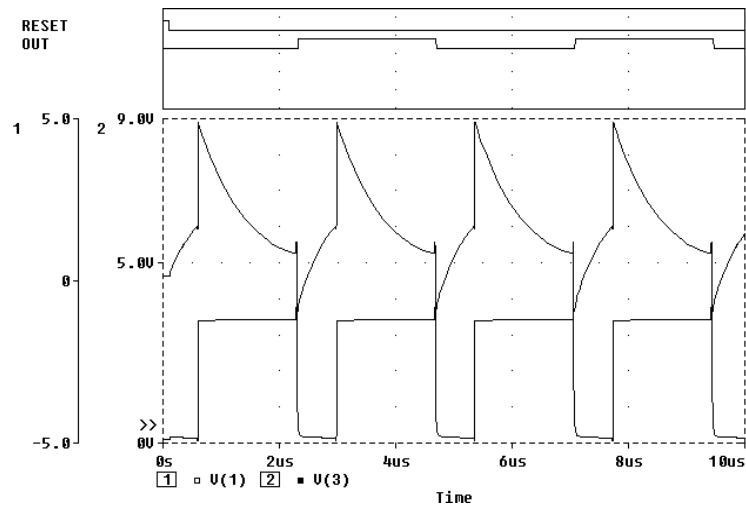


Figure 121 Mixed analog/digital oscillator results,

User interface features for waveform analysis

PSpice A/D provides direct manipulation techniques and shortcuts for analyzing waveform data. These techniques are described below.

Zoom regions

PSpice provides a direct manipulation method for marking the zoom region in either the digital or the analog area of the plot.

To zoom in or out

- 1 Do one of the following on the toolbar:
 - Click the View In toolbar button to zoom in by a factor of 2 around the point you specify.
 - Click the View Out toolbar button to zoom out by a factor of 2 around the point you specify.

Shortcut keys

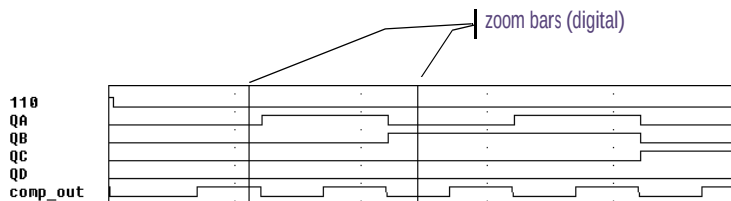
Many of the menu commands in PSpice A/D have equivalent keyboard shortcuts. For instance, after placing a selection rectangle in the analog portion of the plot, you can type **Ctrl** + **A** instead of choosing Area from the View menu. For a list of shortcut keys, search on Keyboard Shortcuts in PSpice A/D Help.



To zoom in the digital area using the mouse

- 1 In the digital area, drag the mouse pointer left or right to produce two vertical bars.

Click the mouse anywhere on the plot to remove the vertical bars without zooming.



- 2 From the View menu, point to Zoom, then choose Area.

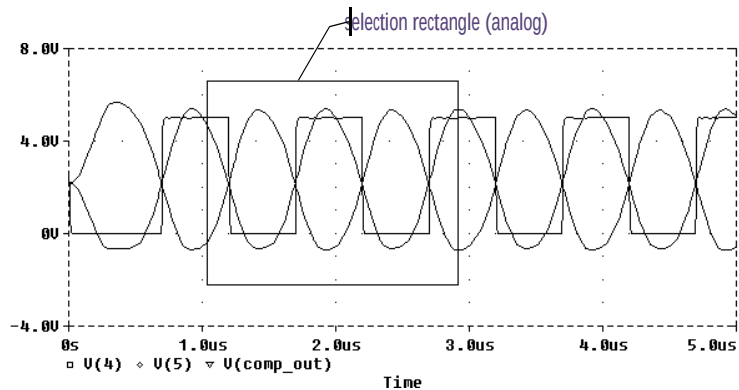
PSpice changes the plot display to the area in between the selection bars. If the plot includes an analog area, it is zoomed in as well.



Click anywhere on the plot to remove the selection rectangle without zooming.

To zoom in the analog area using the mouse

- 1 Drag the mouse pointer to make a selection rectangle as shown below.



- 2 From the View menu, point to Zoom, then choose Area.

PSpice A/D changes the plot to display the region within the selection rectangle. The digital portion of the display, if present, is also zoomed.

Scrolling traces

By default, when a plot is zoomed or when a digital plot contains more traces than can be displayed in the visible area, standard scroll bars appear to the right or at the bottom of the plot area as necessary. These can be used to pan through the data. You can configure scroll bars so they are always present or are never displayed.

To configure scroll bars

- 1 In PSpice A/D, from the Tools menu, choose Options.
- 2 In the Use Scroll Bars frame, choose one of the scroll bars options, as described below.

Table 1

Choose this option...	To do this...
Auto	Have scroll bars appear when a plot is zoomed or when additional traces are displayed in the plot but are not visible (default).
Never	Never display scroll bars. This mode provides maximum plot size and is useful on VGA and other low resolution displays.
Always	Display scroll bars at all times. However, they are disabled if the corresponding axis is full scale.

Sizing digital plots

Sizing bars can be used to change the digital plot size instead of choosing Digital Size from the Plot menu. The digital trace name sizing bar is at the left vertical boundary of the digital plot. If an analog plot area is displayed simultaneously with the digital plot, there is an additional plot sizing bar at the bottom horizontal boundary of the digital plot.

To set the digital plot size using the mouse

- 1 Display at least one digital trace and one analog trace in the Probe window for which you want to set the digital size.
- 2 To change the bottom position of the digital Probe window, do the following:
 - a Place the mouse pointer between the analog and digital parts of the plot.
 - b Click the plot separator.
 - c Drag the plot separator until you have the digital size you want.
- 3 To change the left side of the digital Probe window, do the following:
 - a Place the mouse pointer at the left edge of the digital Probe window you want to resize.
 - b Click the left edge.
 - c Drag the left edge of the digital Probe window to adjust the space available for displaying digital trace names.

To set the digital plot size using menu options

- 1 Display at least one digital trace in the plot for which you want to set the digital size.
- 2 From the Plot menu, choose Digital Size.
- 3 In the Digital Size dialog box, set the following:
 - Percentage of Plot to be Digital
 - Length of Digital Trace Name
- 4 Click OK.

Modifying trace expressions and labels

You can modify trace expressions, text labels, and ellipse labels that are currently displayed within the Probe window, thus eliminating the need to delete and recreate any of these objects.

To modify trace expressions

- 1 Click the trace name to select it (selection is indicated by a color change).
- 2 From the Edit menu, choose Modify Object.
- 3 In the Modify Trace dialog box, edit the trace expression just as you would when adding a trace.

To modify text and ellipse labels

- 1 Click the text or ellipse to select it (selection is indicated by a color change).
- 2 From the Edit menu, choose Modify Object.
- 3 Edit the label by doing one of the following:
 - In the Ellipse Label dialog box, change the inclination angle.
 - In the Text Label dialog box, change the text label.

To place a label, click Plot, point to Label and then choose the desired type of object you want to place.

For information about adding labels (including text, line, poly-line, arrow, box, circle, ellipse, and mark), refer to the online Help in PSpice A/D.

You can also double-click the trace name to modify the trace expression.

For more information on adding traces, see [Adding traces from specific loaded waveform data files on page 17-494](#) and [To add traces using output variables on page 17-519](#).

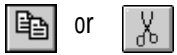
You can also double-click a text or ellipse label to modify it.

Moving and copying trace names and expressions

Trace names and expressions can be selected and moved or copied, either within the same Probe window or to another Probe window.

To copy or move trace names and expressions

- 1 Click one or more (**Shift**+click) trace names. Selected trace names are highlighted.
- 2 From the Edit menu, choose Copy or Cut to save the trace names and expressions to the clipboard. Cut removes trace names and traces from the Probe window.
- 3 In the Probe window where traces are to be added, do one of the following:
 - To add trace names to the end of the currently displayed set, choose Paste from the Edit menu.
 - To add traces before a currently displayed trace name, select the trace name and then choose Paste from the Edit menu.



or



or press **Ctrl**+**V**

When adding a trace to a Probe window, you can make the trace display name different from the trace expression:

- 1 From the Trace menu, choose Add Trace.
- 2 In the Trace Expression text box, enter a trace expression using the syntax:
`trace_expression[:display_name]`
- 3 Click OK.

Here are some considerations when copying or moving trace names and expressions into a different Probe window:

- If the new Probe window is reading the same waveform data file, the copied or moved trace names and expressions display traces that are identical to the original selection set.
- If the new Probe window is reading a *different* waveform data file, the copied or moved names and expressions display different traces generated from the new data.

For example, suppose two waveform data files, MYSIM.DAT and YOURSIM.DAT each contain a V(2) waveform. Suppose also that two Probe windows are currently displayed where window A is loaded with MYSIM.DAT, and window B is loaded with YOURSIM.DAT.

When V(2) is copied from window A to window B, the trace looks different because it is determined by data from YOURSIM.DAT instead of MYSIM.DAT.

Copying and moving labels

Labels can be selected and moved or copied, either within the same Probe window or to another Probe window.

To copy labels

- 1 Select one or more (**Shift**+click) labels, or select multiple labels by drawing a selection rectangle. Selected labels are highlighted.
- 2 From the Edit menu, choose Copy or Cut to save the labels to the clipboard.
Cut removes labels from the Probe window.
- 3 Switch to the Probe window where labels are to be added, and from the Edit menu, choose Paste.
- 4 Click on the new location to place the labels.

For information about adding labels (including text, line, poly-line, arrow, box, circle, ellipse, and mark), refer to the online Help in PSpice A/D.



press **Ctrl**+**V**



To move labels

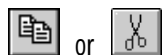
- 1 Select one or more (**Shift**+click) labels, or select multiple labels by drawing a selection rectangle. Selected labels are highlighted.
- 2 Move the labels by dragging them to a new location.

Tabulating trace data values

You can generate a table of data points reflecting one or more traces in the Probe window and use this information in a document or spreadsheet.

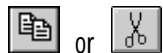
To view the trace data values table

- 1 Select one or more (**Shift**+click) trace names. Selected trace names are highlighted.
- 2 From the Edit menu, choose Copy or Cut to save the trace data point values to the Clipboard.
Cut removes traces from the Probe window.
- 3 In Clipboard Viewer, from the Display menu, choose either Text or OEM Text.



or

Saving the data directly to a file from Clipboard Viewer can create superfluous data at the beginning of the file.



or

To export the data points to other applications

- 1 Select one or more (**Shift**+click) trace names. Selected trace names are highlighted.
- 2 From the Edit menu, choose Copy or Cut to save the trace data point values to the Clipboard.
Cut removes traces from the Probe window.
- 3 Paste the data from the Clipboard into a text editor, a spreadsheet program, or a technical computing program (such as Mathcad).
- 4 Save the file.

Using cursors

When one or more traces are displayed, you can use cursors to display the exact coordinates of two points on the same trace, or points on two different traces. In addition, differences are shown between the corresponding coordinate values for the two cursors.

Displaying cursors

To display both cursors

- 1 From the Trace menu, point to Cursor, then choose Display.

The Probe Cursor window appears, showing the current position of the cursor on the x-axis and y-axis. As you move the cursors, the values in the cursor box change.

In the analog area of the plot (if any), both cursors are initially placed on the trace listed first in the trace legend. The corresponding trace symbol is outlined with a dashed line.

In the digital area of the plot (if any), both cursors are initially placed on the trace named first along the y-axis. The corresponding trace name is outlined with a dashed line.

press **Ctrl** + **⇧ Shift** + **C**



You can move the cursor box any where over the Probe window by dragging the box to another location.

For more information about the cursor commands, refer to the online Help in PSpice A/D.

For a family of curves (such as from a nested DC sweep), you can use the mouse or the arrow keys to move the cursor to one of the other curves in the family. You can also click the desired curve.

Moving cursors

To move cursors along a trace using menu commands

- 1 From the Trace menu, point to Cursor, then choose Peak, Trough, Slope, Min, Max, Point, or Search.

To move cursors along a trace using the mouse

- 1 Use the right and left mouse buttons as described in [Table 2](#) below.

Table 2 *Mouse actions for cursor control*

Click this...	To do this with the cursors...
cursor assignment	
Left-click the analog trace symbol or digital trace name.	Associate the first cursor with the selected trace.
Right-click the analog trace symbol or digital trace name.	Associate the second cursor with the selected trace.
cursor movement	
Left-click in the display area.	Move the first cursor to the closest trace segment at the X position.
Right-click in the display area.	Move the second cursor to the closest trace segment at the X position.

To move cursors along a trace using the keyboard

- 1 Use key combinations as described in [Table 3](#) below.

Table 3 *Key combinations for cursor control*

Us this key combination...	To do this with the cursors...
Ctrl +  and Ctrl + 	Change the trace associated with the first cursor.
Shift + Ctrl +  and Shift + Ctrl + 	Change the trace associated with the second cursor.
 and 	Move the first cursor along the trace.
Shift +  and Shift + 	Move the second cursor along the trace.

Table 3 *Key combinations for cursor control (continued)*

Us this key combination...	To do this with the cursors...
Home	Move the first cursor to the beginning of the trace.
Shift + Home	Move the second cursor to the beginning of the trace.
End	Move the first cursor to the end of the trace.
Shift + End	Move the second cursor to the end of the trace.

Example: using cursors

Figure 122 shows both cursors positioned on the Out signal in the digital area of a plot, and both cursors on the V(1) waveform in the analog area of the plot.

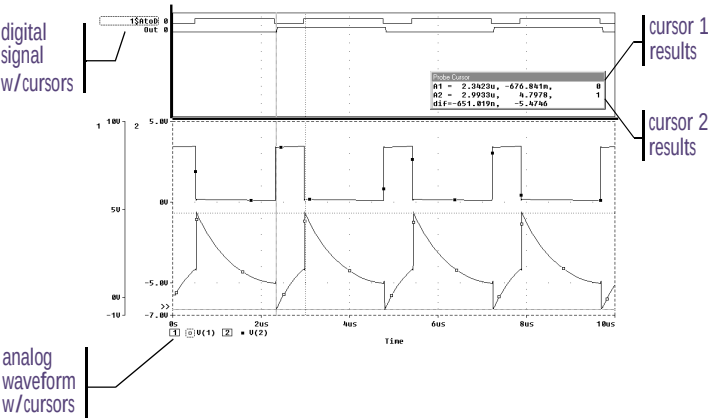


Figure 122 *Cursors positioned on a trough and peak of V(1)*

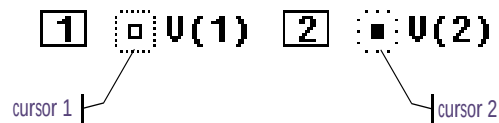
Cursor 1 is positioned on the first trough (dip) of the V(1) waveform. Cursor 2 is positioned on the second peak of the same waveform. In the Probe Cursor window, cursor 1 and cursor 2 coordinates are displayed (A1 and A2, respectively) with their difference shown in the bottom line (dif). The logic state of the Out signal is also displayed to the right of the cursor coordinates.

To position a cursor on the next trough of a waveform, from the Trace menu, point to Cursor, then choose Trough.

To position a cursor on the next peak of a waveform, from the Trace menu, point to Cursor, then choose Peak.

For more information about cursors, refer to the online Help in PSpice A/D.

The mouse buttons are also used to associate each cursor with a different trace by clicking appropriately on either the analog trace symbol in the legend or on the digital trace name (see [Table 2 on page 17-514](#)). These are outlined in the pattern corresponding to the associated cursor's crosshair pattern. Given the example in Figure 122, right-clicking the V(2) symbol will associate cursor 2 with the V(2) waveform. The analog legend now appears as shown below.



The Probe Cursor window also updates the A2 coordinates to reflect the X and Y values corresponding to the V(2) waveform.

Tracking digital simulation messages

PSpice A/D provides explanatory messages for errors that occur during a digital simulation with their corresponding waveforms. You can view messages from:

- the Simulation Message Summary dialog box, or
- the waveform display.

See [Simulation condition messages on page 14-437](#) for information on the message types that can be displayed by PSpice A/D.

Message tracking from the message summary

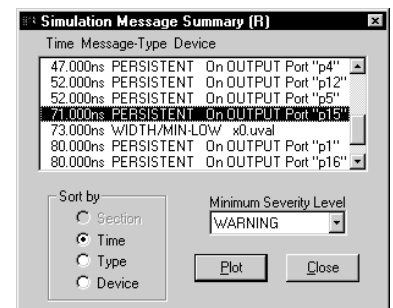
A message summary is available for simulations where diagnostics have been logged to the waveform data file. You can display the message summary:

- When loading a waveform data file (click OK when the Simulation Errors dialog box appears).
- Anytime by choosing Simulation Messages from the View menu.

The Simulation Message Summary dialog box

The Simulation Message Summary dialog box lists message header information. You can filter the messages displayed in the list by selecting a severity level from the Minimum Severity Level drop-down menu. Messages are categorized (in decreasing order of severity) as FATAL, SERIOUS, WARNING, or INFO (informational).

When you select a severity level, the Message Summary displays only those messages with the chosen severity or *higher*. By default, the minimum severity level displayed is SERIOUS.



Example: If you select WARNING as the minimum severity level, the Simulation Message Summary dialog box will display WARNING, SERIOUS, and FATAL messages.

To display waveforms associated with messages

- 1 In the Simulation Message Summary dialog box, double-click a message.

For most message conditions, a Probe window appears that contains the waveforms associated with the simulation condition, along with detailed message text.

Persistent hazards

If a PERSISTENT HAZARD message is displayed, two plots appear (see Figure 123), containing the following:

- the waveforms that initially caused the timing violation or hazard (lower plot)
- the primary outputs or internal state devices to which the condition has propagated (upper plot)

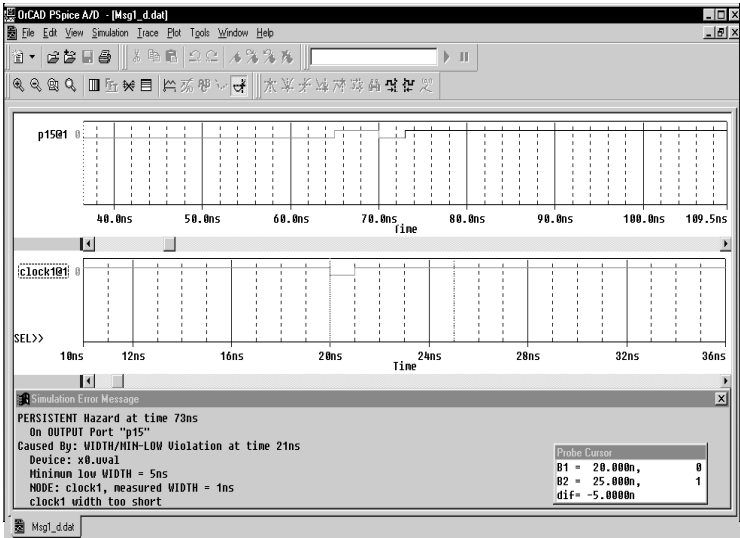


Figure 123 Waveform display for a persistent hazard.

Message tracking from the waveform

Trace segments with associated diagnostics are displayed in the foreground color specified in your PSPICE.INI file. This color is different from those used for standard state transitions.

To display explanatory message text

- 1 Double-click within the tagged region of a trace.

Trace expressions

Traces are referred to by output variable names. Output variables are similar to the PSpice A/D output variables specified in the Simulation Settings dialog box for noise, Monte Carlo, worst-case, transfer function, and Fourier analyses. However, there are additional alias forms that are valid for trace expressions. Both forms are discussed here.

To add traces using output variables

- 1 From the Trace menu, choose Add Trace to display the Add Traces dialog box.
- 2 Construct a trace expression using any combination of these controls:
 - In the Simulation Output Variables frame, click output variables.
 - In the Functions or Macros frame, select operators, functions, constants, or macros.
 - In the Trace Expression text box, type in or edit output variables, operators, functions, constants, or macros.
- 3 If you want to change the name of the trace expression as it displays in the Probe window, use the following syntax:

You can display a subset of the available simulation output variables by selecting or clearing the variable type check boxes in the Simulation Output Variables frame. Variable types not generated by the circuit simulation are dimmed.

For more information about trace expressions, see [Analog trace expressions on page 17-527](#) and [Digital trace expressions on page 17-530](#).

trace expression; display name

4 Click OK.

Basic output variable form

This form is representative of those used for specifying some PSpice A / D analyses.

`<output>[AC suffix](<name>[,name])`

Table 4

This placeholder...	Means this...
<code><output></code>	type of output quantity: V for voltage or I for current (digital values do not require a prefix)
<code>[AC suffix]*</code>	quantity to be reported for an AC analysis. For a list of valid AC suffixes, see Table 8 on page 17-524
<code><name>[,name]</code>	specifies either the <i>net</i> or (+ <i>net</i>, - <i>net</i>) pair for which the voltage is to be reported, or the <i>device</i> for which a current is reported, where: <i>net</i> specifies either the <i>net</i> or <i>pin id</i> (<code><fully qualified device name>:<pin name></code>) <i>device name</i> specifies the fully qualified device name; for a list of device types, see Table 9 on page 17-524 and Table 10 on page 17-525

A fully qualified device name consists of the full hierarchical path followed by the device's reference designator. For information about syntax, see the voltage output variable naming rules on page [8-292](#).

Output variable form for device terminals

This form can only be specified for trace expressions. The primary difference between this and the basic form is that the terminal symbol appears before the *net* or *device name* specification (whereas the basic form treats this as the *pin name* within the *pin id*).

`<output>[terminal]*[AC suffix](<name>[,name])`

Table 5

This placeholder...	Means this...
<code><output></code>	type of output quantity: V for voltage, I for current, or N for noise (digital values do not require a prefix)
<code>[terminal]*</code>	one or more terminals for devices with more than two terminals; for a list of terminal IDs, see Table 10 on page 17-525
<code>[AC suffix]*</code>	quantity to be reported for an AC analysis; for a list of valid AC suffixes, see Table 8 on page 17-524
<code><name>[,<name>])</code>	<i>net</i> , <i>net</i> pair, or fully qualified <i>device name</i> ; for a list of device types, see Table 9 on page 17-524 and Table 10 on page 17-525

[Table 6 on page 17-521](#) summarizes the valid output formats. [Table 7 on page 17-523](#) provides examples of equivalent output variables. Note that some of the output variable formats are unique to trace expressions.

Table 6 Output variable formats

Format	Meaning
Voltage variables	
<code>V[ac](< +analog net > [,< -analog net >])</code>	Voltage between + and - <i>analog net ids</i>
<code>V<pin name>[ac](< device >)</code>	Voltage at <i>pin name</i> of a <i>device</i>

Table 6 Output variable formats (continued)

Format	Meaning
V< x >[ac](<i>< 3 or 4-terminal device ></i>)	Voltage at non-grounded terminal <i>x</i> of a <i>3 or 4-terminal device</i>
V< z >[ac](<i>< transmission line device ></i>)	Voltage at end <i>z</i> of a <i>transmission line device</i> (<i>z</i> is either <i>A</i> or <i>B</i>)
Current variables	
I[ac](<i>< device ></i>)	Current into a <i>device</i>
I< x >[ac](<i>< 3 or 4-terminal device ></i>)	Current into terminal <i>x</i> of a <i>3 or 4-terminal device</i>
I< z >[ac](<i>< transmission line device ></i>)	Current into end <i>z</i> of a <i>transmission line device</i> (<i>z</i> is either <i>A</i> or <i>B</i>)
Digital signal and bus variables	
< <i>digital net</i> >[;< <i>display name</i> >]	Digital state at <i>digital net</i> labeled as <i>display name</i>
{< <i>digital net</i> >*[;< <i>display name</i> >] [< <i>radix</i> >]}	Digital bus labeled as <i>display name</i> and of specified <i>radix</i>
Sweep variables	
< <i>DC sweep variable</i> >	name of any variable used in the DC sweep analysis
FREQUENCY	AC analysis sweep variable
TIME	transient analysis sweep variable

Table 6 *Output variable formats (continued)*

Format	Meaning
Noise variables	
V[db](ONoise)	total RMS-summed noise at output net
V[db](INoise)	total equivalent noise at input source
NTOT(ONoise)	sum of all noise contributors in the circuit
N< noise type >(< device name >)	contribution from <i>noise type</i> of device <i>name</i> to the total output noise*

* See [Table 11 on page 17-526](#) for a complete list of noise types by device type. For information about noise output variable equations, the units used to represent noise quantities in trace expressions, and a noise analysis example, see [Analyzing Noise in the Probe window on page 10-337](#).

Table 7 *Examples of output variable formats*

A basic form	An alias equivalent	Meaning
V(NET3,NET2)	(same)	voltage between analog nets labeled NET3 and NET2
V(C1:1)	V1(C1)	voltage at pin1 of capacitor C1
VP(Q2:B)	VBP(Q2)	phase of voltage at base of bipolar transistor Q2
V(T32:A)	VA(T32)	voltage at port A of transmission line T32
I(M1:D)	ID(M1)	current through drain of MOSFET device M1
QA	(same)	digital state at net QA
{IN1, IN2, IN3}; MYBUS;X	(same)	digital bus made of 3 digital nets (IN1, IN2, IN3) named MYBUS displayed in hexadecimal
VIN	(same)	voltage source named VIN
FREQUENCY	(same)	AC analysis sweep variable
NFID(M1)	(same)	flicker noise from MOSFET M1

Table 8 *Output variable AC suffixes*

Suffix	Meaning of output variables
none	magnitude
DB	magnitude in decibels
G	group delay ($-d\text{PHASE}/d\text{FREQUENCY}$)
I	imaginary part
M	magnitude
P	phase in degrees
R	real part

Table 9 *Device names for two-terminal device types*

Two-terminal device type*	Device type letter
capacitor	C
diode	D
voltage-controlled voltage source**	E
current-controlled current source**	F
voltage-controlled current source**	G
current-controlled voltage source**	H
independent current source	I
inductor	L
resistor	R
voltage-controlled switch**	S
independent voltage source	V
current-controlled switch**	W

* The pin name for two-terminal devices is either 1 or 2.

** The controlling inputs for these devices are not considered terminals.

Table 10 *Terminal IDs by three & four-terminal device type*

Three & four-terminal device type	Device type letter	Terminal IDs
GaAs MOSFET	B	D (drain) G (gate) S (source)
Junction FET	J	D (drain) G (gate) S (source)
MOSFET	M	D (drain) G (gate) S (source) B (bulk, substrate)
Bipolar transistor	Q	C (collector) B (base) E (emitter) S (substrate)
transmission line	T	A (<i>near</i> side) B (<i>far</i> side)
IGBT	Z	C (collector) G (gate) E (emitter)

Table 11 Noise types by device type

Device type	Noise types*	Meaning
B (GaAsFET)	FID	flicker noise
	RD	thermal noise associated with RD
	RG	thermal noise associated with RG
	RS	thermal noise associated with RS
	SID	shot noise
	TOT	total noise
D (diode)	FID	flicker noise
	RS	thermal noise associated with RS
	SID	shot noise
	TOT	total noise
Digital Input	RHI	thermal noise associated with RHI
	RLO	thermal noise associated with RLO
	TOT	total noise
Digital Output	TOT	total noise
J (JFET)	FID	flicker noise
	RD	thermal noise associated with RD
	RG	thermal noise associated with RG
	RS	thermal noise associated with RS
	SID	shot noise
	TOT	total noise
M (MOSFET)	FID	flicker noise
	RB	thermal noise associated with RB
	RD	thermal noise associated with RD
	RG	thermal noise associated with RG
	RS	thermal noise associated with RS
	SID	shot noise
	TOT	total noise

Table 11 **Noise types by device type (continued)**

Device type	Noise types*	Meaning
Q (BJT)	FIB	flicker noise
	RB	thermal noise associated with RB
	RC	thermal noise associated with RC
	RE	thermal noise associated with RE
	SIB	shot noise associated with base current
	SIC	shot noise associated with collector current
	TOT	total noise
R (resistor)	TOT	total noise
Iswitch	TOT	total noise
Vswitch	TOT	total noise

* These variables report the contribution of the specified device's noise to the total output noise in units of V^2/Hz . This means that the sum of all device noise contributions is equal to the total output noise in V^2/Hz , $\text{NTOT}(\text{ONoise})$.

Analog trace expressions

Trace expression aliases

Analog trace expressions vary from the output variables used in simulation analyses because analog net values can be specified by:

<output variable>[:display name]

as opposed to the *<output variable>* format used in analyses. With this format, the analog trace expression can be displayed in the analog legend with an optional alias.

Arithmetic functions

Arithmetic expressions of analog output variables use the same operators as those used in simulation analyses (by means of part property definitions in Capture). You can also include intrinsic functions in expressions. The intrinsic functions available for trace expressions are similar to those available for PSpice A / D math

expressions, but with some differences, as shown in [Table 12](#). A complete list of PSpice A/D arithmetic functions can be found in [Table 10 on page 3-111](#).

Table 12 Analog arithmetic functions for trace expressions

Probe function	Description	Available in PSpice A/D?
ABS(x)	x	YES
SGN(x)	+1 (if x>0), 0 (if x=0), -1 (if x<0)	YES
SQRT(x)	$x^{1/2}$	YES
EXP(x)	e^x	YES
LOG(x)	$\ln(x)$	YES
LOG10(x)	$\log(x)$	YES
M(x)	magnitude of x	YES
P(x)	phase of x (degrees)	YES
R(x)	real part of x	YES
IMG(x)	imaginary part of x	YES
G(x)	group delay of x (seconds)	NO
PWR(x,y)	x ^y	YES
SIN(x)	$\sin(x)$	YES
COS(x)	$\cos(x)$	YES
TAN(x)	$\tan(x)$	YES
ATAN(x) ARCTAN(x)	$\tan^{-1}(x)$	YES
d(x)	derivative of x with respect to the x-axis variable	YES*
s(x)	integral of x over the range of the x-axis variable	YES**
AVG(x)	running average of x over the range of the x-axis variable	NO
AVGX(x,d)	running average of x from X_axis_value(x)-d to X_axis_value(x)	NO
RMS(x)	running RMS average of x over the range of the x-axis variable	NO

Table 12 *Analog arithmetic functions for trace expressions*

Probe function	Description	Available in PSpice A/D?
DB(x)	magnitude in decibels of x	NO
MIN(x)	minimum of the real part of x	NO
MAX(x)	maximum of the real part of x	NO

* In PSpice A/D, this function is called DDT(x).

** In PSpice A/D, this function is called SDT(x).

Note *For AC analysis, PSpice A/D uses complex arithmetic to evaluate trace expressions. If the result of the expression is complex, then its magnitude is displayed.*

Rules for numeric values suffixes

Explicit numeric values are entered in trace expressions in the same form as in simulation analyses (by means of part properties in Capture), with the following exceptions:

- Suffixes M and MEG are replaced with m (milli, 1E-3) and M (mega, 1E+6), respectively.
- MIL and mil are not supported.
- With the exception of the m and M scale suffixes, PSpice is not case sensitive; therefore, upper and lower case characters are equivalent.

Unit suffixes are *only* used to label the axis; they never affect the numerical results. Therefore, it is always safe to leave off a unit suffix.

The units to use for trace expressions are shown in [Table 13](#).

Table 13 *Output units for trace expressions*

Symbol	Unit
V	volt
A	amps
W	watt

Example: V(5) and v(5) are equivalent in trace expressions.

Example: The quantities 2e-3, 2mV, and .002v all have the same numerical value. For axis labeling purposes, PSpice A/D recognizes that the second and third forms are in volts, whereas the first is dimensionless.

PSpice also knows that $W=V \cdot A$, $V=W/A$, and $A=W/V$. So, if you add this trace:

V(5)*ID(M13)

the axis values are labeled with W.

For a demonstration of analog trace presentation, see [Analog example on page 17-497](#).

Table 13 Output units for trace expressions (continued)

Symbol	Unit
d	degree (of phase)
s	second
Hz	hertz

For a procedural discussion of digital trace expressions, see [Analyzing results on page 14-430](#) in the *Digital simulation* chapter.

Example: You can request that four bus lines be displayed together as one hexadecimal digit. You can combine up to 32 digital signals into a bus.

Example: { Q2, Q1, Q0 } specifies a 3-bit bus whose high-order bit is the digital value at net Q2.

Exception: You can display your radix designation option with the digital trace expression by leaving the display name blank and using the following syntax:

digital trace expression;;radix

Digital trace expressions

Digital output variables in trace expressions vary from those used in simulation analyses as follows:

- Digital net values are specified by:
<digital net>[;display name]
as opposed to the *<digital net>* format used for analyses. With this format, the digital signal can be displayed on the digital plot with an optional alias.
- The output from several digital nets can be collected into a single output of higher radix known as a bus.
A bus is formed by enclosing a list of digital net names (separated by blanks or commas) within braces according to the format:
{<high-order net> [mid-order net] <low-order net>}*
The elements of the bus definition, taken left to right, specify the output values of the bus from high order to low order.

By definition, a *digital signal* is any digital net value or a logical expression involving digital nets. For the digital output variable formats described earlier, you can use a digital signal expression everywhere a net name is expected. You can also form buses into expressions using both logical and arithmetic operators.

As a result, the generalized form for defining a digital trace is:

<digital trace expression> [;display name [;radix]]

Table 14

This placeholder...	Means this...
<i>digital trace expression</i>	expression of digital buses or digital signals.
<i>display name</i>	name that will be displayed on the screen; if no display name is specified, the actual trace expression is used; if a display name is given, it is available for use in subsequent trace definitions.
<i>radix</i>	applies only to bus expressions and denotes the radix in which the bus value is to be displayed; the radix is specified as: H or X hexadecimal (default) D decimal O octal B binary

Table 15 presents the operators available for digital signal and bus expressions listed in order of precedence (high to low).

Table 15 *Digital logical and arithmetic operators*

Operator	Meaning
()	grouping
~	logical complement
*	multiplication (bus values only)
/	division (bus values only)
+	addition (bus values only)
-	subtraction (bus values only)
&	and
^	exclusive or
	or

An arithmetic or logical operation between two bus operands results in a bus value that is as wide as is

necessary to contain the result. Prior to the operation, if necessary, the shorter operand is extended to the width of the longer operand by zero-filling on the high-order end.

An arithmetic or logical operation between a bus operand and a signal operand results in a bus value. Prior to the operation, the signal is converted to a bus of width one, then extended if necessary.

You can use signal constants in signal expressions. Specify them as shown in [Table 16](#).

Table 16 Signal constants for digital trace expressions

Signal Constant	Meaning
'0	low
'1	high
'F	falling
'R	rising
'X	unknown
'Z	high impedance

You can use bus constants in bus expressions. Specify them as strings of the form:

r'ddd

Example notations for bus constants:

This notation...	Has this radix...
x'3FFFF	hexadecimal
h'5a	hexadecimal
d'79	decimal
o'177400	octal
b'100110	binary

Table 17

This placeholder...	Means this...
<i>r</i>	case-insensitive radix specifier (x, h, d, o, or b)
<i>ddd</i>	string of digits appropriate to the specified radix

For a discussion and demonstration of digital trace presentation, complete the [Mixed analog/digital tutorial on page 17-500](#).

Other output options

18

Chapter overview

This chapter describes how to output results in addition to those normally written to the data file or output file.

- [Viewing analog results in the PSpice window on page 18-534](#) explains how to monitor the numerical values for voltages or currents on up to three nets in your circuit as the simulation proceeds.
- [Writing additional results to the PSpice output file on page 18-535](#) explains how to generate additional line plots and tables of voltage and current values to the PSpice output file.
- [Creating test vector files on page 18-538](#) explains how to save digital output states to a file that you can use later as input to another circuit.

Viewing analog results in the PSpice window

Capture provides a special WATCH1 part that lets you monitor voltage values for up to three nets in your schematic as a DC sweep, AC sweep or transient analysis proceeds. Results are displayed in PSpice A/D.

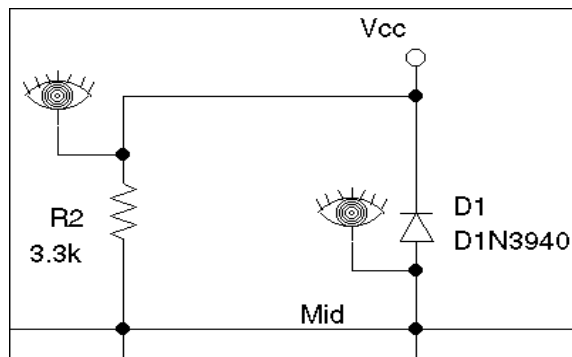
To display voltage values in the PSpice window



- 1 Place and connect a WATCH1 part (from the PSpice library SPECIAL.OLB) on an analog net.
- 2 Double-click the WATCH1 part instance to display the Parts spreadsheet.
- 3 In the ANALYSIS property column, type DC, AC, or TRAN (transient) for the type of analysis results you want to see.
- 4 Enter values in the LO and HI properties columns to define the lower and upper bounds, respectively, on the values you expect to see on this net.
- 5 Repeat steps [1](#) through [4](#) for up to two more WATCH1 instances.
- 6 Start the simulation.

If the results move outside of the specified bounds, PSpice A/D pauses the simulation so that you can investigate the behavior.

For example, in the schematic fragment shown below, WATCH1 parts are connected to the Mid and Vcc nets. After starting the simulation, PSpice A/D displays voltages on the Mid and Vcc nets.



Writing additional results to the PSpice output file

Capture provides special parts that let you save additional simulation results to the PSpice output file as either line-printer plots or tables.

To view the PSpice output file after having run a simulation:

- 1 From the Simulation menu, choose Examine Output.

Generating plots of voltage and current values

You can generate voltage and current line-printer plots for any DC sweep, AC sweep, or transient analysis.

To generate plots of voltage or current to the output file

- 1 Place and connect any of the following parts (from the PSpice library SPECIAL.OLB).

Table 18

Use this part...	To plot this...
VPLOT1	Voltage on the net that the part terminal is connected to.
VPLOT2	Voltage differential between the two nets that the part terminals are connected to.
IPLOT	Current through a net. (Insert this part in series, like a current meter.)



- 2 Double-click the part instance to display the Parts spreadsheet.
- 3 Click the property name for the analysis type that you want plotted: DC, AC, or TRAN.
- 4 In the columns for the analysis type that you want plotted (DC, AC or TRAN), type any non-blank value such as Y, YES or 1.

If you do not enable a format, PSpice A/D defaults to MAG.

- 5
- If you selected the AC analysis type, enable an output format:
- a

Click the property name for one of the following output formats: MAG (magnitude), PHASE, REAL, IMAG (imaginary), or DB.
- b

Type any non-blank value such as Y, YES or 1.
- c

Repeat the previous steps (a) and (b) for as many AC output formats as you want to see plotted.
- 6
- Repeat steps 2 through 5 for any additional analysis types you want plotted.

Note If you do not enable an analysis type, PSpice A/D reports the transient results.

Generating tables of voltage and current values

You can generate tables of voltage and current values on nets for any DC sweep, AC sweep, or transient analysis.

To generate tables of voltage or current to the output file

- 1
- Place and connect any of the following parts (from the PSpice library SPECIAL.OLB).

Table 19

Use this part...	To tabulate this...
 VPRINT1	Voltage on the net that the part terminal is connected to.
 VPRINT2	Voltage differential between the two nets that the part terminals are connected to.
 IPRINT	Current through a cut in the net. (Insert this part in series, like a current meter.)

- 2
- Double-click the part instance to display the Parts spreadsheet.

- 3 Click the property name for the analysis type that you want tabulated: DC, AC, or TRAN.
- 4 In the columns for the analysis type that you want plotted (DC, AC or TRAN), type any non-blank value such as Y, YES or 1.
- 5 If you selected the AC analysis type, enable an output format.
 - a Click the property name for one of the following output formats: MAG (magnitude), PHASE, REAL, IMAG (imaginary), or DB.
 - b Type any non-blank value such as Y, YES or 1.
 - c Repeat the previous steps (a) and (b) for as many AC output formats as you want to see tabulated.
- 6 Repeat steps 2 through 5 for any additional analysis types you want plotted.

If you do not enable a format, PSpice A/D defaults to MAG.

Note *If you do not enable an analysis type, PSpice A/D reports the transient results.*

Generating tables of digital state changes

You can generate a table of digital state changes during a transient analysis for any net.

To generate a table of digital state changes to the output file

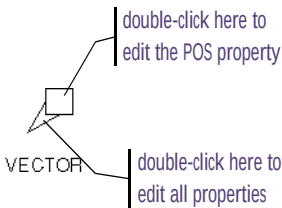
- 1 Place a PRINTDGTLCG part (from the PSpice library SPECIAL.OLB) and connect it to the net that you are interested in.



Creating test vector files

To find out about vector file syntax, refer to the online *OrCAD PSpice A/D Reference Manual*.

To find out about setting up digital stimuli, see [Defining a digital stimulus on page 14-413](#).



Note You can group separate signal values to form a hex or octal value by specifying the same POS property and defining RADIX as Hex or Octal. Define the bit position within the value using the BIT property.

Capture provides a special VECTOR part that lets you save digital simulation results to a vector file. Whenever any net with an attached VECTOR part changes state, PSpice A / D writes a line of *time-value* data to the vector file using the same format as the file stimulus device. This means that you can use the vector file to drive inputs for another simulation.

To generate a test vector file from your circuit

- 1 Place a VECTORn part (from the PSpice library SPECIAL.OLB) and connect it to a wire or bus at the output of a digital part instance.
- 2 Double-click the VECTORn part instance to display the Parts spreadsheet.
- 3 Set the part properties as described below.

Table 20

For this property...	Define this...
POS	Column position in the file. Valid values range from 1 to 255.
FILE	Name of the vector file. If left blank, PSpice A / D creates a file named SCHEMATIC_NAME.VEC.
RADIX	If the VECTOR part is attached to a bus, the numerical notation for a bus. Valid values are B[inary], O[ctal], and H[ex].
BIT	If the VECTOR part is attached to a wire, the bit position within a single hex or octal digit.
SIGNAMES	Names of the signals that appear in the header of the file. If left blank, PSpice A / D defaults to the following: • For a wire, the label (name) on the wire. • For a bus, a name derived from the position of each signal in the bus (from MSB to LSB).

- 4 Repeat steps **1** through **3** for as many test vectors as you want to create.

Setting initial state

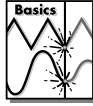
A

Appendix overview

This appendix includes the following sections:

- [Save and load bias point on page A-542](#)
- [Setpoints on page A-544](#)
- [Setting initial conditions on page A-546](#)

Note *Bias point is not supported in PSpice A/D Basics.*



If the circuit uses high gain components, or if the circuit's behavior is nonlinear around the bias point, this feature is not useful.

Save and load bias point

Save Bias Point and Load Bias Point are used to save and restore bias point calculations in successive PSpice A/D simulations. Saving and restoring bias point calculations can decrease simulation times when large circuits are run multiple times and can aid convergence.

Save/Load Bias Point affect the following types of analyses:

- transient
- DC
- AC

Save bias point

Save bias point is a simulation control function that allows you to save the bias point data from one simulation for use as initial conditions in subsequent simulations. Once bias point data is saved to a file, you can use the load bias point function to use the data for another simulation.

To use save bias point

- 1 In the Simulation Settings dialog box, click the Analysis tab.
- 2 Under Options, select Save Bias Point.
- 3 Complete the Save Bias Point dialog box.
- 4 Click OK.

See [Setting up analyses on page 8-289](#) for a description of the Analysis Setup dialog box.

Load bias point

Load bias point is a simulation control function that allows you to set the bias point as an initial condition. A common reason for giving PSpice A/D initial conditions is to select one out of two or more stable operating points (set or reset for a flip-flop, for example).

To use load bias point

- 1 Run a simulation using the Save Bias Point option in the Simulation Settings dialog box.
- 2 Before running another simulation, click the Analysis tab in the Simulation Settings dialog box.
- 3 Under Options, select Load Bias Point.
- 4 Specify a bias point file to load. Include the path if the file is not located in your working directory, or use the Browse button to find the file.
- 5 Click OK.

See [Setting up analyses on page 8-289](#) for a description of the Analysis Setup dialog box.

Setpoints

Pseudocomponents that specify initial conditions are called setpoints. These apply to the analog portion of your circuit.

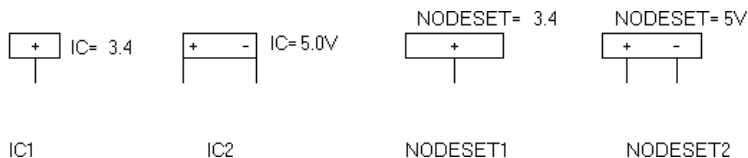


Figure A-1 Setpoints.

The example in Figure A-1 includes the following:

- IC1 a one-pin symbol that allows you to set the initial condition on a net for both small-signal and transient bias points
- IC2 a two-pin symbol that allows you to set initial condition between two nets

Using IC symbols sets the initial conditions for the bias point only. It does not affect the DC sweep. If your circuit design contains both an IC symbol and a NODESET symbol for the same net, the NODESET symbol is ignored.

To specify the initial condition, edit the value of the VALUE property to the desired initial condition.

PSpice A/D attaches a voltage source with a 0.0002 ohm series resistance to each net to which an IC symbol is connected. The voltages are clamped this way for the entire bias point calculation.

NODESET1 is a one-pin symbol which helps calculate the bias point by providing a initial guess for some net.

NODESET2 is a two-pin symbol which helps calculate the bias point between two nets. Some or all of the circuit's nets may be given an initial guess. NODESET symbols are effective for the bias point (both small-signal and transient bias points) and for the first step of the DC sweep. It has no effect during the rest of the DC sweep or during the transient analysis itself.

Unlike the IC pseudocomponents, NODESET provides only an initial guess for some net voltages. It does not clamp those nodes to the specified voltages. However, by providing an initial guess, NODESET symbols may be used to break the tie (in a flip-flop, for instance) and make it come up in a desired state. To guess at the bias point, enter the initial guess in the Value text box for the VALUE property. PSpice A/D attaches a voltage source with a 0.0002 ohm series resistance to each net to which an IC symbol is connected.

These pseudocomponents are netlisted as PSpice A/D .IC and .NODESET commands. Refer to these commands in the online *OrCAD PSpice A/D Reference Manual* for more information. Setpoints can be created for inductor currents and capacitor voltages using the IC property described in [Setting initial conditions on page A-546](#).

Setting initial conditions

The IC property allows initial conditions to be set on capacitors and inductors. These conditions are applied during all bias point calculations. However, if you select the Skip Initial Transient Solution check box in the Transient Analysis Setup dialog box, the bias point calculation is skipped and the simulation proceeds directly with transient analysis at TIME=0. Devices with the IC property defined start with the specified voltage or current value; however, all other such devices have an initial voltage or current of 0.

Note *Skipping the bias point calculation can make the transient analysis subject to convergence problems.*

See [Setpoints on page A-544](#) for more information about IC1 and IC2.

Applying an IC property for a capacitor has the same effect as applying one of the pseudocomponents IC1 or IC2 across its nodes. PSpice A/D attaches a voltage source with a 0.002 ohm series resistance in parallel with the capacitor. The IC property allows the user to associate the initial condition with a device, while the IC1 and IC2 pseudocomponents allow the association to be with a node or node pair.

In the case of initial currents through inductors, the association is only with a device, and so there are no corresponding pseudocomponents. The internal implementation is analogous to the capacitor. PSpice A/D attaches a current source with a 1 Gohm parallel resistance in series with the inductor.

Convergence and “time step too small errors”

B

Appendix overview

This appendix discusses common errors and convergence problems in PSpice.

- [Introduction on page B-548](#)
- [Bias point and DC sweep on page B-553](#)
- [Transient analysis on page B-556](#)
- [Diagnostics on page B-561](#)

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 AC and noise analyses are linear and do not use an iterative algorithm, so the following discussion does not apply to them. Digital devices are evaluated using boolean algebra; this discussion does not apply to them either.

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.

Newton-Raphson requirements

The Newton-Raphson algorithm is *guaranteed to converge to a solution*. However, this guarantee has some conditions:

- 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. Remember that the PSpice algorithms are used in computer hardware that

has finite precision and finite dynamic range that produce these limits:

- Voltages and currents in PSpice are limited to +/-1e10 volts and amps.
- Derivatives in PSpice are limited to 1e14.
- The arithmetic used in PSpice is double precision and has 15 digits of accuracy.

Is there a solution?

Yes, for any physically realistic circuit. However, it is not difficult to set up a circuit that has no solution within the limits of PSpice numerics. Consider, for example, a voltage source of one megavolt connected to a resistor of one micro-ohm. This circuit does not have a solution within the dynamic range of currents (+/- 1e10 amps).

Here is another example:

```
V1      1,      0      5v
D1      1,      0      DMOD
.MODEL          DMOD(IS=1e-16)
```

The problem here is that the diode model has no series resistance. 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:

$$I = 1e-16 * e^{200} = 7.22e70 \text{ amps}$$

This circuit also does not have a solution within the limits of the dynamic range of PSpice. In general, be careful of components without limits built into them. Extra care is needed when using the expressions for controlled sources (such as for behavioral modeling). It is easy to write expressions with very large values.

To find out more about the diode equations, refer to the *Analog Devices* chapter in the online *OrCAD PSpice A/D Reference Manual*.

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. This example tries to approximate an ideal switch using the diode model:

```
.MODEL DMOD(IS=1e-16 N=1e-6)
```

The current through this diode is:

$$I = 1e-16 * e^{V / (N * 0.025)} = 1e-16 * e^{V / 25e-9}$$

Avoid unrealistic model parameters. Behavioral modeling expressions need extra care.

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.

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?

Newton-Raphson is guaranteed to converge only if the analysis is started close to the answer. Also, there is no measurement that can tell how close is close enough.

PSpice 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 alternative 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 $\pm 1e10$. 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 “squelsches” 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.

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.

Skiping the bias point

The SKIPBP option for the transient analysis skips the bias point calculation. In this case the transient analysis has no known solution to start from and, therefore, is not assured of converging at the first time point. Because of this, its use is not recommended. Its inclusion in PSpice 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 10uf
```

“10” (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.

Note OrCAD recommends 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.

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 $R = 2\pi \cdot 100e3 \cdot .001 = 628$ 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
			E				
(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.

Index

A

ABM

- ABM part templates, 198
- ABM.OLB, 195
- basic controlled sources, 239
- cautions and recommendations for simulation, 232
- control system parts, 199
- custom parts, 239
- frequency domain device models, 227
- frequency domain parts, 227, 233
- instantaneous models, 222, 232
- overview, 194
- placing and specifying ABM parts, 196
- PSpice A/D-equivalent parts, 220–221
- signal names, 193
- simulation accuracy, 238
- syntax, 221
- triode modeling example, 217

AC stimulus property, 326

AC sweep analysis, 288, 323–324

- about, 324
- displaying simulation results, 79
- example, 77, 329
- introduction, 44

noise analysis, 288, 333

setup, 77, 324, 327

stimulus, 325

treatment of nonlinear devices, 331

ACMAG stimulus property, 326

ACPHASE stimulus property, 326

adding a stimulus, 73

AGND ground part, 124

ambiguity

cumulative hazard, 439

analog behavioral modeling, *see* ABM

analog parts

basic components (ABM), 199, 201

basic controlled sources (ABM), 239

behavioral, 106

bipolar transistors, 137, 297, 525, 527

breakout, 105

capacitors, 295

Chebyshev filters, 199, 203, 237, 393

Darlington model transistors, 137

diodes, 137, 295, 526

expression parts (ABM), 200, 214

frequency table parts (ABM), 220, 229, 237

GaAsFET, 296, 525–526

IGBT, 137, 297, 525

inductors, 295

integrators and differentiators (ABM), 199, 206
 JFET, 137, 296, 525–526
 Laplace transform (ABM), 200, 210, 220, 227, 233
 limiters (ABM), 199, 202
 math functions (ABM), 200, 213
 mathematical expressions (ABM), 220
 MOSFET, 137, 297, 525–526
 nonlinear magnetic core, 137
 opamp, 137
 passive, 104
 PSpice A / D-equivalent parts (ABM), 220
 resistors, 296, 527
 switch, 527
 table look-up (ABM), 199, 206, 220, 225
 transmission lines, 297, 525
 vendor-supplied, 101
 voltage comparator, 137
 voltage reference, 137
 voltage regulator, 137
 analyses
 AC sweep, 77, 288, 323–324
 bias point detail, 62, 288
 DC sensitivity, 288, 320
 DC sweep, 66, 288, 306
 execution order, 291
 Fourier, 288
 frequency response, 288
 Monte Carlo, 289, 381
 noise, 288, 333
 overview, 43
 parametric, 82, 288, 364
 performance analysis, 89
 sensitivity / worst-case, 289, 398
 setup, 289
 small-signal DC transfer, 288, 317
 temperature, 289, 373
 transient, 72, 288
 types, 288
 approximation, problems, 551
 AtoD interface, *see* mixed analog / digital circuits

B

basic components (ABM), 199, 201
 basic controlled sources (ABM), 239
 behavioral modeling expressions, 555
 behavioral parts, 106
 bias point
 convergence analysis, 557

 save / restore, 542
 skipping, 557
 bias point detail analysis, 288
 example, 62
 introduction, 43
 bipolar transistors, 137, 297, 525, 527
 problems, 560
 Bode plot, 44, 80

C

CAPACITANCE (I / O model parameter), 456
 capacitors, 295
 CD4000_PWR digital power part, 115
 CD4000_PWR parts (power supply), 451
 charge storage nets, 258
 Chebyshev filters, 199, 203, 237, 393
 circuit file (.CIR), 50
 simulating multiple circuits, 300
 color settings for waveform analysis, 480
 COMMANDn stimulus property (digital), 423
 comparator, 137
 CONSTRAINT primitive, 106, 278
 continuous equations
 problems, 550
 control system parts (ABM), 199
 controlled sources, 220, 239
 convergence analysis
 bias point, 557
 convergence hazard, 439
 convergence problems, 547
 approximations, 551
 behavioral modeling expressions, 555
 bias point, 553
 bipolar transistors, 560
 continuous equations, 550
 DC sweep, 553
 derivatives, 550
 diagnostics, 561
 dynamic range of time, 557
 inductors and transformers, 559
 Newton-Raphson requirements, 548
 parasitic capacitances, 559
 semiconductors, 553
 switches, 554
 transient analysis, 556
 Create Subcircuit command, 133, 157
 current source, controlled, 220, 239
 cursors, waveform analysis, 513

custom part creation for models, 175
 using the Model Editor, 142, 173

D

Darlington model transistors, 137

DC analyses

displaying simulation results, 68
see also DC sweep analysis, bias point detail
 analysis, small-signal DC transfer
 analysis, DC sensitivity analysis

DC sensitivity analysis, 288, 320

introduction, 43

DC stimulus property, 311

DC sweep analysis, 288, 306

about, 308

curve families, 313

example, 66

introduction, 43

nested, 311

setting up, 66

stimulus, 310

DELAY stimulus property (digital), 422

derivative

problems, 550

design

preparing for simulation, 49, 96

device noise, 334, 337

total, 337

diagnostic problems, 561

differentiators (ABM), 199, 206

DIG_GND stimulus property (digital), 423

DIG_PWR stimulus property (digital), 423

DIGCLOCK digital stimulus parts, 119, 413, 422

DIGDRVF (strengths), 264

DIGDRVZ (strengths), 264

DIGERRDEFAULT (simulation option), 440

DIGERRLIMIT (simulation option), 440

DIGIFPWR digital power part, 115

DIGIOLVL (simulation option), 249

digital models, 271

digital parts

\$G_DGND (reserved global net), 456

\$G_DPWR (reserved global net), 456

CONSTRAINT primitive, 106

DIGIFPWR (power supply), 456

logic propagation delay selection, 428

LOGICEXP primitive, 106

PINDLY primitive, 106

vendor-supplied, 101

digital primitives, 243, 272

input (N device), 266

output (O device), 266

propagation delays, *see* timing model

syntax, 246

timing model, *see* timing model

digital simulation

messages, 437

propagation delays, *see* timing model

states, 262, 411

strengths, 262

timing model, *see* timing model

vector file, 538

waveform display, 500, 527, 530

worst-case timing, 458

digital values, 411

digital worst-case timing, 458

compared to analog worst-case, 458

convergence hazard, 439

cumulative ambiguity hazard, 439

glitch suppression, 439

DIGMNTYMX (simulation option), 459

DIGMNTYSCALE (simulation option), 252

DIGOVRDRV (simulation option), 264

DIGPOWER (I/O model), 258

DIGSTIM digital stimulus part, 119, 414

DIGTYMXSCALE (simulation option), 252

diodes, 137, 295, 526

DRVH (I/O model parameter), 456

DRVH (I/O model), 258, 263

DRVL (I/O model parameter), 456

DRVL (I/O model), 258, 263

DRVZ (I/O model), 258

DtoA interface, *see* mixed analog / digital circuits

dynamic range of time, 557

E

ECL_100K_PWR digital power part, 115

ECL_10K_PWR digital power part, 115

EGND ground part, 124

examples and tutorials

AC sweep analysis, 77, 329

analog waveform analysis, 495

bias point detail analysis, 62

creating a digital model, 280

DC sweep analysis, 66

example circuit creation, 56

- frequency response vs. arbitrary parameter, 370
- mixed analog / digital waveform analysis, 500
- modeling a triode (ABM), 217
- Monte Carlo analysis, 385
- parametric analysis, 82
- performance analysis, 89, 366
- transient analysis, 72
- using the Model Editor, 146, 156
- worst-case analysis, 401

expression parts (ABM), 200, 214

expressions, 109–110

- see also* parameters
- ABM, 220
- functions, 111
- specifying, 109
- system variables, 113
- waveform analysis, 527

F

files

- generated by Capture, 50
- user-configurable, 51
- with simulation results, 54

FILESTIM digital stimulus part, 119, 424

flicker noise, 337

FORMAT stimulus property (digital), 423

Fourier analysis, 288

- introduction, 45

FREQUENCY output variable, 522

frequency response vs. arbitrary parameter, 370

frequency table parts (ABM), 220, 229, 237

functions

- PSpice A / D, 111
- waveform analysis, 528

G

GaAsFET, 296, 525–526

glitch suppression, 439

goal functions, 367

- in performance analysis, 368
- single data point, 368

grid spacing

- part graphics, 178
- part pins, 178

ground

- missing, 124
- missing DC path to, 125

- parts, 100

group delay, 524

H

histograms, 393

hysteresis curves, 359

I

I / O model, 245, 248, 257, 445

- and switching times (TSW), 259

DIGPOWER, 258

DRVH, 258

DRVL, 258

DRVZ, 258

INLD, 258

INR, 258

OUTLD, 258

- parameter summary, 260

TPWRT, 255, 258

TSTOREMN, 258

IAC stimulus part, 325

IC (property), 546

ICn initial conditions parts, 544

IDC stimulus part, 114, 310

IGBT, 137, 297, 525

imaginary part, 524

include files, 51

- configuring, 53, 162
- with model definitions, 163

inductors, 295

- problems, 559

inertial delay, 255

initial conditions, 542, 546

INLD (I / O model), 258

input noise, total, 337

INR (I / O model), 258

instance models

- and the Model Editor, 143, 154
- changing model references, 159
- editing, 145
- reusing, 160
- saving for global use instead
 - using the Model Editor, 155

integrators (ABM), 199, 206

interface subcircuits, 266, 444, 456

- and I / O models, 248, 445
- and power supplies, 444

CAPACITANCE, 266
 customized, 266
 DRVH, 266
 DRVL, 266
 IO_LEVEL, 246
 N device (digital input), 266
 O device (digital output), 266
 syntax, 266
 IO_LEVEL
 interface subcircuit parameter, 246
 part property, 189
 stimulus property (digital), 423
 IO_LEVEL property, 189
 IO_MODEL stimulus property (digital), 423
 IPLOT (write current plot part), 535
 IPRINT (write current table part), 536
 ISRC stimulus part, 114, 310, 325
 ISTIM stimulus part, 117

J

JFET, 137, 296, 525–526

L

Laplace transform parts (ABM), 200, 210, 220, 227, 233, 235
 libraries
 configuring, 162
 footprint, 53
 model, 130
 package, 53
 part (.OLB), 53
 searching for models, 163
 see also model libraries
 Library List, using the, 103
 limiters (ABM), 199, 202
 loading delay, 254
 LOGICEXP primitive, 106, 271

M

magnetic core, nonlinear, 137
 magnitude, 524
 markers, 490
 displaying traces, 68
 for limiting waveform data file size, 490
 for waveform display, 487

math function parts (ABM), 200, 213
 mathematical expressions (ABM), 220
 messages, simulation, 437
 mixed analog / digital circuits, 271, 288
 I/O models, 445
 interface subcircuits, 444
 power supplies, 444, 456
 waveform display, 500, 527, 530

MNTYMXDLY

 part property, 190
 timing model parameter, 246

Model Editor

 about, 50, 152
 analyzing model parameter effects, 139
 changing
 .MODEL definitions, 152
 .SUBCKT definitions, 153
 model names, 153
 creating parts for models, 142, 173
 custom, 175
 example, 156
 fitting models, 139
 starting stand-alone, 141
 starting from the schematic page editor, 143
 supported device types, 137
 testing and verifying models, 138
 tutorial, 146
 using data sheet information, 138
 viewing performance curves, 140
 ways to use, 136

model editor

 running from the
 schematic page editor, 153

model libraries, 51, 130

 adding to the configuration, 164
 analog list of, 121
 and duplicate model names, 164
 configuration, 131
 configured as include files, 163
 configuring, 53, 122, 162
 digital list of, 121
 directory search path, 167
 global vs. design, 131, 165
 how PSpice searches them, 163
 nested, 132
 OrCAD-provided, 132
 preparing for part creation, 172
 search order, 163, 166

MODEL property, 129, 180

models

- built-in, 42
 - changing associations to parts, 159
 - creating parts for
 - custom, 175
 - using the Model Editor, 142, 173
 - creating with the Model Editor, 152
 - defined as
 - parameter sets, 129
 - subcircuits, 129, 157
 - digital models, 271
 - global vs. design, 131
 - instance, 143, 154, 159–160
 - organization, 130
 - preparing for part creation, 172
 - saving as design
 - using the Model Editor, 143
 - saving as local
 - using the Model Editor, 153
 - testing/ verifying (Model Editor-created), 138
 - tools to create, 133
 - ways to create/ edit, 134
 - Monte Carlo analysis, 289, 381
 - collating functions, 379
 - histograms, 393
 - introduction, 47
 - model parameter values reports, 377
 - output control, 377
 - tutorial, 385
 - using the Model Editor, 156
 - waveform reports, 378
 - with temperature analysis, 380
 - MOSFET, 137, 297, 525–526
 - multiple y-axes, waveform analysis, 368, 502
- ## N
- netlist
 - failure to netlist, 98
 - file (.NET), 50
 - Newton-Raphson requirements, 548
 - nodes, interface, 444
 - NODESETn initial conditions parts, 544
 - noise analysis, 288, 333
 - about, 44, 334
 - device noise, 334
 - flicker noise, 337
 - noise equations, 337
 - setup, 333, 335
 - shot noise, 337
 - thermal noise, 337
 - total output and input noise, 334
 - units of measure, 338
 - viewing results, 338
 - viewing simulation results, 337, 526
 - waveform analysis output variables, 337, 526
 - noise units, 338
 - non-causality, 235
 - nonlinear
 - magnetic core, 137
 - nonlinear devices
 - in AC sweep analysis, 331
 - NOOUTMSG (simulation option), 440
 - NOPRBMSG (simulation option), 440
- ## O
- OFFTIME stimulus property (digital), 422
 - ONTIME stimulus property (digital), 422
 - opamp, 137
 - operators in expressions, 110
 - OPPVAL stimulus property (digital), 422
 - options
 - DIGERRDEFAULT, 440
 - DIGERRLIMIT, 440
 - DIGIOLVL, 249
 - DIGMNTYMX, 459
 - DIGMNTYSCALE, 252
 - DIGOVRDRV, 264
 - DIGTYMXSCALE, 252
 - NOOUTMSG, 440
 - NOPRBMSG, 440
 - RELTOL, 238
 - OUTLD (I/O model), 258
 - output control parts, 100, 535
 - output file (.OUT), 64
 - control parts, 535
 - messages, 437
 - tables and plots, 535
 - output noise, total, 337
 - output variables
 - arithmetic expressions, 527
 - digital signals and buses, 530
 - digital trace expression, 530
 - logic/ arithmetic operators, 531
 - noise (waveform analysis), 337, 526
 - PSpice A/D, 292
 - waveform analysis, 519, 530–531
 - waveform analysis functions, 528

P

- PARAM global parameter part, 107
- parameters, 107
- parametric analysis, 288, 364
 - analyzing waveform families, 85
 - example, 82
 - frequency response vs. arbitrary parameter, 370
 - introduction, 46
 - performance analysis, 366
 - setting up, 83
 - temperature analysis, 289, 373
- parasitic capacitance, 559
- part wizard
 - using custom parts, 175
- parts
 - creating for models
 - using the Model Editor, 142, 173
 - creating new stimulus parts, 352
 - editing graphics, 177
 - grid spacing
 - graphics, 178
 - pins, 178
 - ground, 100
 - non-simulation, 182
 - output control, 100
 - pins, 123, 178
 - preparing model libraries for part creation, 172
 - properties for simulation, 181
 - saving as global
 - using the Model Editor, 142, 173
 - simulation control, 100
 - simulation properties, 171
 - stimulus, 100
 - ways to create for models, 171
- AGND (ground), 124
- BBREAK (GaAsFET), 105
- behavioral, 106
- breakout, 105
- C (capacitor), 104
- CBREAK (capacitor), 105
- CD4000_PWR (digital power), 115
- creating for models
 - custom parts, 175
 - using the Model Editor, 173
- CVAR (capacitor), 104
- D (diode), 104
- DBREAK (diode), 105
- DIGCLOCK (digital stimulus), 119
- DIGIFPWR (digital power), 115
- DIGSTIM (digital stimulus), 119
- ECL_100K_PWR (digital power), 115
- ECL_10K_PWR (digital power), 115
- EGND (ground), 124
- FILESTIM (digital stimulus), 119
- finding, 102
- IAC (AC stimulus), 325
- ICn (initial conditions), 544
- IDC (DC stimulus), 114, 310
- IO_LEVEL property, 189
- ISRC (analog stimulus), 114, 310, 325
- ISTIM (transient stimulus), 117
- JBREAK (JFET), 105
- K_LINEAR (transformer), 104
- KBREAK (inductor coupling), 105
- KCOUPLEN (coupled transmission line), 104
- LBREAK (inductor), 105
- MBREAK (MOSFET), 105
- MNTYMXDLY property, 190
- MODEL property, 180
- NODESETn, 544
- PARAM (global parameter), 107
- passive, 104
- PSPICEDEFAULTNET properties, 191
- QBREAK (bipolar transistor), 105
- R (resistor), 104
- RBREAK (resistor), 105
- RVAR (resistor), 104
- SBREAK (voltage-controlled switch), 105
- STIMn (digital stimulus), 119
- T (ideal transmission line), 104
- TBREAK (transmission line), 105
- TEMPLATE property, 182
- TLOSSY (Lossy transmission line), 104
- TnCOUPLEDx (coupled transmission line), 104
- unmodeled, 120
- VAC (AC stimulus), 116, 325
- VDC (DC stimulus), 114, 116
- vendor-supplied, 101
- VEXP (transient stimulus), 116
- VPULSE (transient stimulus), 116
- VPWL (transient stimulus), 116
- VPWL_F_N_TIMES (transient stimulus), 117
- VPWL_F_RE_FOREVER (transient stimulus), 116
- VPWL_N_TIMES (transient stimulus), 117
- VPWL_RE_FOREVER (transient stimulus), 116
- VSFFM (transient stimulus), 117
- VSIN (transient stimulus), 117
- VSRC (analog stimulus), 114, 116, 325

- VSTIM (analog stimulus), 116
- VSTIM (transient stimulus), 117
- WBREAK (current-controlled switch), 105
- XFRM_LINEAR (transformer), 104
- XFRM_NONLINEAR (transformer), 105
- ZBREAK (IGBT), 105
- ABMn and ABMn/I (ABM), 200, 214
- ABS (ABM), 200, 213
- ARCTAN (ABM), 200, 213
- ATAN (ABM), 200, 213
- BANDPASS (ABM), 199, 204
- BANDREJ (ABM), 199, 205
- CONST (ABM), 199, 201
- COS (ABM), 200, 213
- DIFF (ABM), 199, 201
- DIFFER (ABM), 199, 206
- DIGIFPWR (power supply), 451
- E (ABM controlled analog source), 239
- ECL_100K_PWR (power supply), 451
- ECL_10K_PWR (power supply), 451
- EFREQ (ABM), 220, 229
- ELAPLACE (ABM), 220, 227
- EMULT (ABM), 220, 224
- ESUM (ABM), 220, 224
- ETABLE (ABM), 220, 225
- EVALUE (ABM), 220, 222–223
- EXP (ABM), 200, 213
- F (ABM controlled analog source), 239
- FTABLE (ABM), 199, 207
- G (ABM controlled analog source), 239
- GAIN (ABM), 199, 201
- GFREQ (ABM), 220, 229
- GLAPLACE (ABM), 220, 227
- GLIMIT (ABM), 199, 202
- GMULT (ABM), 220, 224
- GSUM (ABM), 220, 224
- GTABLE (ABM), 220, 225
- GVALUE (ABM), 220, 222–223
- H (ABM controlled analog source), 239
- HIPASS (ABM), 199, 204
- ICn (initial condition), 544
- INTEG (ABM), 199, 206
- LAPLACE (ABM), 200, 210
- LIMIT (ABM), 199, 202
- LOG (ABM), 200, 213
- LOG10 (ABM), 200, 213
- LOPASS (ABM), 199, 203
- MULT (ABM), 199, 201
- NODESETn (initial bias point), 544
- PWR (ABM), 200, 213
- PWRS (ABM), 200, 213
- SIN (ABM), 200, 213
- SOFTLIM (ABM), 199, 202
- SQRT (ABM), 200, 213
- SUM (ABM), 199, 201
- TABLE (ABM), 199, 206
- TAN (ABM), 200, 213
- performance analysis, 366
 - example, 89
 - goal functions, 367
- phase, 524
- PINDLY primitive, 106, 271
- plots
 - sizing, 508
- plots in waveform analysis, 477
- power supplies, 456
 - \$G_DGND, 456
 - \$G_DPWR, 456
 - A/D interfaces, 114
 - analog, 114
 - default digital power supply selection by PSpice A/D, 449
 - DIGIFPWR, 456
 - digital, custom CD4000, TTL, or ECL, 450, 453
- primitives, digital, 272
- PRNTDGTLCCHG (write digital state changes part), 537
- Probe windows
 - plot update methods, 509
 - plots, 477–478
 - printing Probe windows, 479
 - scrolling, 507
 - setting colors, 480
 - sizing plots, 508
 - trace data tables, 512
 - traces, displaying, 68
 - zoom regions, 505
- propagation delay, *see* timing model
- properties (part) for simulation, 181
- PSpice
 - default shortcut keys, 505
 - waveform analysis, 476
 - multiple y-axes, 368
- PSpice A/D
 - about, 42
 - default power supply selection, 449
 - expressions, 109
 - functions, 111
 - output file (.OUT), 64, 535
 - output variables, 292

- PSpice A/D-equivalent parts, 220–221
- simulation status window, 301, 534
- starting, 299
- using with other programs, 49
- viewing in-progress output values, 534
- waveform data file (.DAT), 54
- PSPICE.INI file, editing, 480
- PSPICEDEFAULTNET properties, 191

R

- real part, 524
- regulator, 137
- RELTOL (simulation option), 238
- resistors, 296, 527

S

- schematic page editor
 - starting other tools from
 - Model Editor, 143, 153–154
- scrolling, Probe windows, 507
- semiconductor
 - problems, 553
- shot noise, 337
- simulation
 - about, 42
 - analysis
 - execution order, 291
 - setup, 289
 - types, 288
 - batch jobs, 300
 - bias point, 542
 - failure to start, 98
 - initial conditions, 542, 546
 - messages, 437
 - output file (.OUT), 64
 - setup checklist, 96
 - starting, 299
 - status window, 301
 - troubleshooting checklist, 98
- simulation control parts, 100
 - ICn, 544
 - NODESETn (initial conditions), 544
 - PARAM, 107
- small-signal DC transfer analysis, 288, 317
 - introduction, 43
- STARTVAL stimulus property (digital), 422
- states, digital, 262, 411

- STIMn digital stimulus parts, 119, 422
- Stimulus Editor, 73, 346
 - about, 49
 - creating new stimulus parts, 352
 - defining analog stimuli, 117
 - defining digital inputs, 414
 - defining stimuli, 349
 - editing a stimulus, 353
 - manual stimulus configuration, 354
 - starting, 347
 - stimulus files, 346–347
- stimulus files, 51
 - configuring, 53, 162
- stimulus generation, 344
 - manually configuring, 354
- stimulus, adding, 73
 - AC sweep, 325
 - bus transitions (digital), 417
 - clock transitions (digital), 414
 - DC sweep, 310
 - for multiple analysis types, 118
 - loops (digital), 420
 - signal transitions (digital), 415
 - transient (analog / mixed-signal), 344
 - transient (digital), 413
- subcircuits, 129
 - analog / digital interface, 444
 - creating .SUBCKT definitions from designs, 133
 - creating .SUBCKT definitions from schematics, 157
 - tools to create, 133
 - ways to create / edit, 134
 - see also* models
- switch, 527
 - problems, 554
- system variables in expressions, 113

T

- table look-up parts (ABM), 199, 206, 220, 225
- temperature analysis, 289, 373
 - introduction, 46
 - with statistical analyses, 380
- TEMPLATE property, 182
 - and non-simulation parts, 182
 - examples, 185
 - naming conventions, 183
 - regular characters, 182
 - special characters, 184

- test vector file, 538
- thermal noise, 337
- TIME (Probe output variable), 522
- TIMESTEP stimulus property (digital), 423
- timing model, 245, 248, 251
 - hold times (TH), 251
 - inertial delay, 255
 - loading delay, 254
 - propagation delays, 251, 428
 - calculation, 254
 - DIGMNTYSCALE, 252
 - DIGTYMXSCALE, 252
 - MNTYMXDLY, 246
 - unspecified, 252
 - pulse widths (TW), 251
 - setup times (TSU), 251
 - switching times (TSW), 251
 - transport delay, 256
 - unspecified timing constraints, 253
- timing violations and hazards
 - convergence, 439
 - cumulative ambiguity, 439
 - persistent hazards, 435
- total noise, 334
 - circuit, 337
 - per device, 337
- TPWRT (I/O model), 255, 258
- traces
 - adding, 68
 - direct manipulation, 505
 - displaying, 68, 75
 - markers, 490
 - output variables, 519
 - placing a cursor on, 70
- transformer
 - problems, 559
- transient analysis, 288
 - example, 72
 - Fourier analysis, 288
 - hysteresis curves, 359
 - internal time steps, 358
 - introduction, 45
 - overview, 342
 - problems, 556
 - setting up, 74
 - Stimulus Editor, 346
 - stimulus generation, 344
 - switching circuits, 359
 - transient response, 356
- transistors, Darlington model, 137

- transmission lines, 525
- transport delay, 256
- triode, 217
- troubleshooting
 - checklist, 98
 - missing DC path to ground, 125
 - missing ground, 124
 - unconfigured libraries and files, 122
 - unmodeled parts, 120
 - unmodeled pins, 123
- TSTOREMN (I/O model), 258
- TTL, 456
- tutorials, *see* examples and tutorials

U

- unmodeled
 - parts, 120
 - pins, 123
- updating plots, 509

V

- VAC stimulus part, 116, 325
- variables in expressions, 113
- VDC (DC stimulus), 310
- VDC stimulus part, 114, 116
- VECTOR (write digital vector file part), 538
- vector file, 538
- vendor-supplied parts, 101
- VEXP stimulus part, 116
- voltage comparator, 137
- voltage reference, 137
- voltage regulator, 137
- voltage source, controlled, 220, 239
- VPLOTn (write voltage plot part), 535
- VPRINTn (write voltage table part), 536
- VPULSE stimulus part, 116
- VPWL stimulus part, 116
- VPWL_F_N_TIMES stimulus part, 117
- VPWL_F_RE_FOREVER stimulus part, 116
- VPWL_N_TIMES stimulus part, 117
- VPWL_RE_FOREVER stimulus part, 116
- VSFFM stimulus part, 117
- VSIN stimulus part, 117
- VSRC stimulus part, 114, 116, 119, 310, 325
- VSTIM stimulus part, 73, 116–117

W

- WATCH1 (view output variable part), 534
- waveform analysis, 476
 - about, 48
 - adding traces, 68
 - cursors, 513
 - digital display name, 531
 - digital signals and buses, 530
 - displaying simulation results, 68, 79
 - expressions, 527
 - functions, 528
 - hysteresis curves, 359
 - limiting waveform data file size, 490
 - logic/arithmetic operators, 531
 - messages, 437
 - multiple y-axes, 368, 502
 - output variables, 519, 530
 - for noise, 337, 526
 - performance analysis, 89, 366
 - placing a cursor on a trace, 70
 - plot, 477
 - printing Probe windows, 479
 - setting colors, 480
 - trace data tables, 512
 - traces, 490
 - traces, displaying, 505
 - traces, using output variables, 519
 - using markers, 487
 - waveform data file (.DAT), 54
 - waveform data file formats, 495
 - waveform families, 85, 313
- waveform data file formats, 495
- waveform families, displaying, 85
- wavform analysis
 - arithmetic expressions, 527
 - output variables, 530
- WIDTH stimulus property (digital), 423
- worst-case analysis, 289, 398
 - collating functions, 379
 - example, 401
 - hints, 405
 - introduction, 47
 - model parameter values reports, 377
 - output control, 377
 - overview, 398
 - waveform reports, 378
 - with temperature analysis, 380

Z

- zoom regions, Probe windows, 505

