HSPICE® Command Reference

Release U-2003.09-RA, September 2003

Comments?
E-mail your comments about Synopsys documentation to doc@synopsys.com

SYNOPSYS®

Copyright Notice and Proprietary Information

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

> "This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of _____ and its employees. This is copy number _____."

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Registered Trademarks (®)

Synopsys, AMPS, Arcadia, C Level Design, C2HDL, C2V, C2VHDL, Cadabra, Calaveras Algorithm, CATS, CoCentric, COSSAP, CSim, DelayMill, Design Compiler, DesignPower, DesignWare, Device Model Builder, EPIC, Formality, HSPICE, Hypermodel, I, iN-Phase, InSpecs, in-Sync, LEDA, MAST, Meta, Meta-Software, ModelAccess, ModelExpress, ModelTools, PathBlazer, PathMill, Photolynx, Physical Compiler, PowerArc, PowerMill, PrimeTime, RailMill, Raphael, RapidScript, Saber, SmartLogic, SNUG, SolvNet, Stream Driven Simulator, SiVL, Superlog, System Compiler, Testify, TetraMAX, TimeMill, TMA, Vera, and Virtual Stepper are registered trademarks of Synopsys, Inc.

Trademarks (™)

abraCAD, abraMAP, Active Parasitics, AFGen, Apollo, Apollo II, Apollo-DPII, Apollo-GA, ApolloGAII, Astro, Astro-Rail, Astro-Xtalk, Aurora, AvanTestchip, AvanWaves, BCView, Behavioral Compiler, BOA, BRT, Cedar, ChipPlanner, Circuit Analysis, Columbia, Columbia-CE, Comet 3D, Cosmos, CosmosEnterprise, CosmosLE, CosmosScope, CosmosSE, Cyclelink, Davinci, DC Expert, DC Expert *Plus*, DC Professional, DC Ultra, DC Ultra Plus, Design Advisor, Design Analyzer, DesignerHDL, DesignTime, DFM-Workbench, DFT Compiler, Direct RTL, Direct Silicon Access, DW8051, DWPCI, Dynamic-Macromodeling, Dynamic Model Switcher, ECL Compiler, ECO Compiler, EDAnavigator, Encore, Encore PQ, Evaccess, ExpressModel, Floorplan Manager, Formal Model Checker, FormalVera, FoundryModel, FPGA Compiler II, FPGA *Express*, Frame Compiler, Frameway, Galaxy, Gatran, HDL Advisor, HDL Compiler, Hercules, Hercules-Explorer, Hercules-II, Hierarchical Optimization Technology, High Performance Option, HotPlace, HSPICE-Link, iN-Tandem, Integrator, Interactive Waveform Viewer, iQBus, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, JVXtreme, Liberty, Libra-Passport, Library Compiler, Libra-Visa, LRC, Mars, Mars-Rail, Mars-Xtalk, Medici, Metacapture, Metacircuit, Metamanager, Metamixsim, Milkyway, ModelSource, Module Compiler, MS-3200, MS-3400, NanoSim, Nova Product Family, Nova-ExploreRTL, Nova-Trans, Nova-VeriLint, Nova-VHDLlint, OpenVera, Optimum Silicon, Orion_ec, Parasitic View, Passport, Planet, Planet-PL, Planet-RTL, Polaris, Polaris-CBS, Polaris-MT, Power Compiler, PowerCODE, PowerGate, ProFPGA, Progen, Prospector, Proteus OPC, Protocol Compiler, PSMGen, Raphael-NES, RoadRunner, RTL Analyzer, Saturn, ScanBand, Schematic Compiler, Scirocco, Scirocco-i, Shadow Debugger, Silicon Blueprint, Silicon Early Access, SinglePass-SoC, Smart Extraction, SmartLicense, SmartModel Library, Softwire, Source-Level Design, Star, Star-DC, Star-MS, Star-MTB, Star-Power, Star-Rail, Star-RC, Star-RCXT, Star-Sim, Star-Sim XT, Star-Time, Star-XP, SWIFT, Taurus, Taurus-Device, Taurus-Layout, Taurus-Lithography, Taurus-OPC, Taurus-Process, Taurus-Topography, Taurus-Visual, Taurus-Workbench, The Power in Semiconductors, TimeSlice, TimeTracker, Timing Annotator, TopoPlace, TopoRoute, Trace-On-Demand, True-Hspice, TSUPREM-4, TymeWare, VCS, VCS Express, VCSi, Venus, Verification Portal, VFormal, VHDL Compiler, VHDL System Simulator, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (℠)

DesignSphere, MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license.
AMBA is a trademark of ARM Limited. ARM is a registered trademark of ARM Limited.
All other product or company names may be trademarks of their respective owners.

Table of Contents

Preface

This preface includes the following sections:

About this Manual

The *HSPICE Command Reference* describes how to use HSPICE to maintain signal integrity in your chip design.

Audience

This manual is for circuit designers and engineers who use Synopsys HSPICE for circuit simulation and analysis.

Related Publications

For additional information about the *HSPICE Command Reference*, see

- Synopsys Online Documentation (SOLD), which is included with the software for CD users or is available to download through the Synopsys Electronic Software Transfer (EST) system
- Documentation on the Web, which is available through SolvNet at http://solvnet.synopsys.com
- The Synopsys MediaDocs Shop, from which you can order printed copies of Synopsys documents, at http://mediadocs.synopsys.com

You might also want to refer to the documentation for the following related Synopsys products:

- *HSPICE Simulation and Analysis User Guide*
- *HSPICE Signal Integrity Guide*
- *HSPICE Elements and Device Models Manual*
- *HSPICE MOSFET Models Manual*
- *HSPICE Command Reference*

What's New

This manual is only available online.

New Features

See the *Release Notes* for information about new features and last-minute changes.

Conventions

This manual uses certain style conventions to indicate functions, node/port names (in C code), examples, and file names.

Commands

SYNTAX:

command_name[*argument(s)*]

argument types: keyword | *value* | tag=*value* | tag=keyword

Command Argument	Definition
keyword	Keywords are identifiers that must be used as they appear. They are shown in base font.
value	Values are user-determined. They are shown in italic text to distinguish them from commands and keywords.
tag= *value* keyword	Tags can be followed by either a value or a keyword. Tags and keywords are in the base font. Argument values are in italics to distinguish them from commands, keywords, and tags.

Symbol	Definition
|	A pipe symbol (|) represents the word “or” and separates choices between two or more arguments.
...	An ellipsis (...) indicates that more than one argument can be specified. Ellipses are used only for multiple arguments with tags.
[ ]	Open and closed square brackets indicate that the enclosed argument is optional.
( )	Open and closed parenthesis indicate that there is a choice between the enclosed arguments (two or more). These are used only when a command has several groups of argument choices; multiple pipe symbols (|), in this case, would result in an ambiguous syntax.

SYNTAX:

report_node_i a[vg] | r[ms] | p[eak] | h[ist] *node_name(s)*

For this command, a[vg], r[ms], p[eak], and h[ist] are keywords–choose one of these. The *node_name(s)* are user-determined.

EXAMPLE 1:

`report_node_i a VDD GND`

In this example, a is a keyword, and *VDD* and *GND* are values.

SYNTAX:

report_node_ic [a[ll]] [q[uoted]] [for=epic | spice] [*time(s)*]

For this command, a[ll] and q[uoted] are optional keywords. The tag for= is part of an optional argument for which you can choose the keyword epic or spice. The *time(s)* are user-determined and, in this case, optional.

EXAMPLE 2:

```
report_node_ic all for=spice 1u
```

In this example, `all` is a keyword, `for=` is a tag, `spice` is a keyword, and `1u` is a value.

Menu Text, File Names, and Examples

Menu text appears in bold, as shown in the following example.

EXAMPLE 1:

To start setting up a run, select **File > Design Data Setup**.

File names are shown in the same font as the surrounding text, but in italics.

EXAMPLE 2:

This is an example of a *file_name.out* being shown in text.

Examples are shown in a courier font as they might appear on your screen. Each example is followed by an explanation. Anything shown in the example that appears in the explanation is shown in the same courier font used in the example.

EXAMPLE 3:

```
add_node_cap RS100 275
```

In the example, the capacitance value of node `RS100` is increased by `275` femtofarads.

Functions

SYNTAX:

int **fmDelayDrive**(*delay, port_id, w, l*)

For this command, int is the data type of the function and *delay, port_id, w, and l* are user-determined parameters.

DESCRIPTION:

Each function description includes a table that provides the data type and description for each function parameter.

Data Type	Parameter	Description
double	*delay*	Delay time in 0.01 ns.
char *	*port_name*	The name of the port for which the intrinsic delay time will be set.
double	*w*	Width specified in 0.01 µ.
double	*l*	Length specified in 0.01 µ.

Some functions (few) have an alternate syntax that includes 0.0 as a keyword. Because this parameter is not user-determined, it is not shown in italic text and must be typed as is.

Port/Node Names and Examples

Names of ports and nodes are always specified in double quotes. This is a standard of the C programming language.

EXAMPLE 1:

```
fmDisablePort("out");
```

For each example there is a corresponding explanation. Anything shown in the example that appears in the explanation is shown in the same courier font used in the example.

EXAMPLE 2:

```
fmDisablePortById(out_id);
```

This example disables the port with ID previously defined as `out_id` for the current model.

Customer Support

Customer support is available through SolvNet online customer support and through contacting the Synopsys Technical Support Center.

Accessing SolvNet

SolvNet includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet also gives you access to a wide range of Synopsys online services including software downloads, documentation on the Web, and “Enter a Call With the Support Center.”

To access SolvNet:

1. Go to the SolvNet Web page at http://solvnet.synopsys.com/.
2. If prompted, enter your user name and password. (If you do not have a Synopsys user name and password, click New Synopsys User Registration.)

If you need help using SolvNet, click SolvNet Help in the column on the left side of the SolvNet Web page.

Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

- Open a call to your local support center from the Web.
  - Go to http://solvnet.synopsys.com (Synopsys user name and password required) and click “Enter a Call With the Support Center.”
- Send an e-mail message to support_center@synopsys.com.
- Telephone your local support center.
  - Call (800) 245-8005 from within the continental United States.
  - Call (650) 584-4200 from Canada.
  - Find other local support center telephone numbers at http://www.synopsys.com/support/support_ctr.

1

Command Categories

This chapter lists all commands that you can use in HSPICE, arranged by the tasks that use each command.

Alter Blocks

Use these commands in your HSPICE netlist to run alternative simulations of your netlist, using different data.

.ALIAS
.ALTER
.DEL LIB
.TEMP

Analysis

Use these commands in your HSPICE netlist to start different types of HSPICE analysis, to save the simulation results into a file, and to load the results of a previous simulation into a new simulation.

.AC
.DC
.DISTO
.ENDDATA
.FFT
.FOUR
.LIN
.NET
.NOISE
.OP
.PZ
.SAMPLE
.SENS
.TEMP
.TF
.TRAN

Conditional Block

Use these commands in your HSPICE netlist to setup a conditional block. HSPICE does not execute the commands in the conditional block, unless the specified conditions are true.

.ELSE
.ELSEIF
.ENDIF
.IF

Encryption

Use these commands in your HSPICE netlist to mark the start and end of an encrypted section of a netlist.

.PROTECT
.UNPROTECT

Field Solver

Use these commands in your HSPICE netlist to define a field solver.

.FSOPTIONS
.LAYERSTACK
.MATERIAL
.SHAPE

Input/Output Buffer Information Specification (IBIS)

Use these commands in your HSPICE netlist for specifying input/output buffer information.

.EBD
.IBIS
.PKG

Library Management

Use these commands in your HSPICE netlist to manage libraries of circuit designs, and to call other files when simulating your netlist.

.DEL LIB
.ENDL
.INCLUDE
.LIB
.PROTECT
.UNPROTECT
.VEC

Model Definition

Use these commands in your HSPICE netlist to define models.

.MALIAS
.MODEL

Node Naming

Use these commands in your HSPICE netlist to name nodes in circuit designs.

.CONNECT
.GLOBAL

Output Porting

Use these commands in your HSPICE netlist to specify the output of a simulation, to a printer, plotter, or graph. You can also define the parameters to measure, and to report in the simulation output.

.BIASCHK
.DOUT
.GRAPH
.MEASURE
.PLOT
.PRINT
.PROBE
.STIM
.WIDTH

Setup

Use these commands in your HSPICE netlist to setup your netlist for simulation.

.DATA
.DCVOLT
.GLOBAL
.IC
.LOAD
.NODESET
.OPTION
.PARAM
.SAVE
.TITLE
.VEC

Simulation Runs

Use these commands in your HSPICE netlist to mark the start and end of individual simulation runs, and conditions that apply throughout an individual simulation run.

.END
.TEMP
.TITLE

Subcircuits

Use these commands in your HSPICE netlist to define subcircuits, and to add instances of subcircuits to your netlist.

.ENDS
.EOM
.INCLUDE
.MACRO
.MODEL
.SUBCKT

Command Categories:

2

Commands in HSPICE Netlists

This chapter contains an alphabetical listing of all commands that you can use in an HSPICE netlist. For a list of commands grouped according to tasks that use each command, see Chapter 1, “Command Categories.”

- .DEL LIB
- .DISTO
- .DOUT
- .EBD
- .ELSE
- .ELSEIF
- .END
- .ENDDATA
- .ENDIF
- .ENDL
- .ENDS
- .EOM
- .FFT
- .FOUR
- .FSOPTIONS
- .GLOBAL
- .GRAPH
- .IBIS
- .IC
- .IF
- .INCLUDE

- .LAYERSTACK
- .LIB
- .LIN
- .LOAD
- .MACRO
- .MALIAS
- .MATERIAL
- .MEASURE
- .MODEL
- .NET
- .NODESET
- .NOISE
- .OP
- .OPTION
- .PARAM
- .PKG
- .PLOT
- .PRINT
- .PROBE
- .PROTECT
- .PZ
- .SAMPLE

- .SAVE
- .SENS
- .SHAPE
- .STIM
- .SUBCKT
- .TEMP
- .TF
- .TITLE
- .TRAN
- .UNPROTECT
- .VEC
- .WIDTH

.AC

SYNTAX:

Single/Double Sweep

.AC *type np fstart fstop*

.AC *type np fstart fstop* <SWEEP *var* <START=>*start*
+ <STOP=>*stop* <STEP=>*incr*>

.AC *type np fstart fstop* <SWEEP *var type np start stop*>

.AC *type np fstart fstop*
+ <SWEEP *var* START="*param_expr1*"
+ STOP="*param_expr2*" STEP="*param_expr3*">

.AC *type np fstart fstop* <SWEEP *var start_expr*
+ *stop_expr step_expr*>

Sweep Using Parameters

.AC *type np fstart fstop* <SWEEP DATA = *datanm*>

.AC DATA = *datanm*

.AC DATA = *datanm* <SWEEP *var* <START=>start
<STOP=>stop
+ <STEP=>*incr*>

.AC DATA = *datanm* <SWEEP *var type np start stop*>

.AC DATA = *datanm* <SWEEP *var* START="*param_expr1*"
+ STOP="*param_expr2*" STEP="*param_expr3*">

.AC DATA = *datanm* <SWEEP *var start_expr stop_expr*
+ *step_expr*>

Optimization

.AC DATA = *datanm* OPTIMIZE = *opt_par_fun*
+ RESULTS = *measnames* MODEL = *optmod*

Random/Monte Carlo

.AC *type np fstart fstop* <SWEEP MONTE = *val*>

EXAMPLE 1:

```
.AC DEC 10 1K 100MEG
```

This example performs a frequency sweep, by 10 points per decade, from 1 kHz to 100 MHz.

EXAMPLE 2:

```
.AC LIN 100 1 100HZ
```

This example runs a 100-point frequency sweep from 1 Hz to 100 Hz.

EXAMPLE 3:

```
.AC DEC 10 1 10K SWEEP cload LIN 20 1pf 10pf
```

This example performs an AC analysis, for each value of cload. This results from a linear sweep of cload between 1 pF and 10 pF (20 points), sweeping the frequency by 10 points per decade, from 1 Hz to 10 kHz.

EXAMPLE 4:

```
.AC DEC 10 1 10K SWEEP rx POI 2 5k 15k
```

This example performs an AC analysis, for each value of rx, 5 k and 15 k, sweeping the frequency by 10 points per decade, from 1 Hz to 10 kHz.

EXAMPLE 5:

```
.AC DEC 10 1 10K SWEEP DATA = datanm
```

This example uses the **.DATA** statement to perform a series of AC analyses, modifying more than one parameter. The datanm file contains the parameters.

EXAMPLE 6:

```
.AC DEC 10 1 10K SWEEP MONTE = 30
```

This example illustrates a frequency sweep, and a Monte Carlo analysis, with 30 trials.

DESCRIPTION:

You can use the **.AC** statement in several different formats, depending on the application, as shown in the examples below. You can also use the **.AC** statement to perform data-driven analysis in HSPICE.

If the input file includes an **.AC** statement, HSPICE runs AC analysis for the circuit, over a selected frequency range, for each parameter in the second sweep.

For AC analysis, the data file must include at least one independent AC source element statement (for example, VI INPUT GND AC 1V). HSPICE checks for this condition, and reports a fatal error if you did not specify such AC sources.

Command Argument	**Definition**
DATA = *datanm*	Data name, referenced in the **.AC** statement
incr	Increment value of the voltage, current, element, or model parameter. If you use *type* variation, specify the *np* (number of points) instead of *incr*.
fstart	Starting frequency. If you use POI (list of points) type variation, use a list of frequency values, not fstart fstop.
fstop	Final frequency.
MONTE = val	Produces a number (*val*) of randomly-generated values. HSPICE uses these values to select parameters from a distribution, either *Gaussian*, *Uniform*, or *Random Limit*.

Command Argument	Definition
np	Number of points, or points per decade or octave, depending on which keyword precedes it.
start	Starting voltage or current, or any parameter value for an element or model.
stop	Final voltage or current, or any parameter value for an element or a model.
SWEEP	This keyword indicates that the **.AC** statement specifies a second sweep.
TEMP	This keyword indicates a temperature sweep
type	Can be any of the following keywords: • DEC – decade variation. • OCT – octave variation. • LIN – linear variation. • POI – list of points.
var	Name of an independent voltage or current source, element or model parameter, or the TEMP (temperature sweep) keyword. HSPICE supports source value sweep, referring to the source name (SPICE style). If you select a parameter sweep, a **.DATA** statement, and a temperature sweep, then you must choose a parameter name for the source value. You must also later refer to it in the **.AC** statement. The parameter name cannot start with V or I.

SEE ALSO:

.DC

.TRAN

.ALIAS

SYNTAX:

.ALIAS <*model_name1*> <*model_name2*>

EXAMPLE 1:

You delete a library named *poweramp*, that contains a model named *pa1*. Another library contains an equivalent model named *par1*. You can then alias the pa1 model name to the par1 model name:

```
.ALIAS pa1 par1
```

During simulation, when HSPICE encounters a model named *pa1* in your netlist, it initially cannot find this model, because you used a **.ALTER** statement to delete the library that contained this model. However, the **.ALIAS** statement indicates to use the *par1* model, in place of the old *pa1* model. HSPICE *does* find this new model in another library, so simulation continues.

You must specify an old model name and a new model name to use in its place. You cannot use **.ALIAS** without any model names:

```
.ALIAS
```

or with only *one* model name:

```
.ALIAS pa1
```

You also cannot alias a model name to *more than one* model name, because then the simulator would not know which of these new models to use in place of the deleted or renamed model:

```
.ALIAS pa1 par1 par2
```

For the same reason, you cannot alias a model name to a second model name, and then alias the second model name to a third model name:

```
.ALIAS pa1 par1
.ALIAS par1 par2
```

If your netlist does not contain a **.ALTER** command, and if the **.ALIAS** does not report a usage error, then the **.ALIAS** does not affect the simulation results.

EXAMPLE 2:

Your netlist might contain the statement:

```
.ALIAS myfet nfet
```

Without a **.ALTER** statement, HSPICE does not use *nfet* to replace *myfet* during simulation.

If your netlist contains one or more **.ALTER** commands, the first simulation uses the original *myfet* model. After the first simulation, if the netlist references *myfet* from a deleted library, **.ALIAS** substitutes *nfet* in place of the missing model.

- If HSPICE finds model definitions for both *myfet* and *nfet*, it reports an error and aborts.
- If HSPICE finds a model definition for *myfet*, but not for *nfet*, it reports a warning, and simulation continues, using the original *myfet* model.
- If HSPICE finds a model definition for *nfet*, but not for *myfet*, it reports a *replacement successful* message.

DESCRIPTION:

You can use **.ALTER** statements to rename a model, to rename a library containing a model, or to delete an entire library of models in HSPICE. If your netlist references the old model name, then after you use one of these types of **.ALTER** statements, HSPICE no longer finds this model.

For example, if you use **.DEL LIB** in the **.ALTER** block to delete a library, the **.ALTER** command deletes all models in this library. If your netlist references one or more models in the deleted library, then HSPICE no longer finds the models.

To resolve this issue, HSPICE provides a **.ALIAS** command, to let you alias the old model name to another model name that HSPICE can find in the existing model libraries.

SEE ALSO:

.MALIAS

.ALTER

SYNTAX:

.ALTER *<title_string>*

EXAMPLE:

```
.ALTER simulation_run2
```

DESCRIPTION:

You can use the **.ALTER** statement to rerun an HSPICE simulation, using different parameters and data.

Use parameter (variable) values for print and plot statements, before you alter them. The **.ALTER** block cannot include **.PRINT**, **.PLOT**, **.GRAPH** or any other input/output statements. You can include analysis statements (**.DC**, **.AC**, **.TRAN**, **.FOUR**, **.DISTO**, **.PZ**, and so on) in a **.ALTER** block in an input netlist file.

However, if you change only the analysis type, and you do not change the circuit itself, then simulation runs faster if you specify all analysis types in one block, instead of using separate **.ALTER** blocks for each analysis type.

The **.ALTER** sequence or block can contain:

- Element statements (except source elements)
- **.DATA** statements
- **.DEL LIB** statements
- **.INCLUDE** statements
- **.IC** (initial condition) and **.NODESET** statements
- **.LIB** statements
- **.MODEL** statements

- **.OP** statements
- **.OPTION** statements
- **.PARAM** statements
- **.TEMP** statements
- **.TF** statements
- **.TRAN**, **.DC**, and **.AC** statements
- **.ALIAS** statements

Command Argument	Definition
title_string	Any string up to 72 characters. HSPICE prints the appropriate title string for each **.ALTER** run, in each section heading of the output listing, and in the graph data (.tr#) files.

.BIASCHK

SYNTAX:

.BIASCHK *type* terminal1=*t1* terminal2=*t2* limit=*lim*
+ <noise=*ns*><name=*devname1*><name=*devname2*>...
+ <mname=*modelname1*><mname=*modelname2*> ...

* Check transistor mode of operation
.BIASCHK *type* region=<*region*>
+ <*name=devname1*> <*name=devname2*> ...
+ <*mname=modelname1*> <*mname=modelname2*>

EXAMPLE:

```
.BIASCHK NMOS terminal1=ng terminal2=nb limit=2v
+ noise=0.01v name=x1.x3.m1 mname=nch.1 name=m3
```

DESCRIPTION:

Breakdown can occur if a voltage bias between some terminals of an element is too large. The **.BIASCHK** statement monitors the voltage bias, using the limits and noise that you define. Bias monitoring checks the specified bias, during transient analysis, and reports:

- Element name
- Time
- Terminals
- Bias that exceeds the limit
- Number of times the bias exceeds the limit for an element

HSPICE saves the information as both a warning and a BIASCHK summary, in the **.lis* file.

You can use this command *only* for active elements and capacitors.

A **.BIASCHK** statement might check for voltages that exceed a specified limit, for MOS dielectric breakdown. BIASCHK can check voltages from the gate, to the source, drain, or bulk.

BIASCHK cannot detect the bias that exceeds the limit, if the bias is always the same value during transient analysis.

If a model name, referenced in an active element statement, contains a period (.), then **.BIASCHK** reports an error. This occurs because it is unclear whether a reference such as x.123 is a model name or a sub-circuit name (123 model in the x sub-circuit).

Instance (element) and model names can contain wildcards, either ? (stands for one character) or * (stands for 0 or more characters).

If you do not set *name* and *mname*, HSPICE checks all elements of this type for bias voltage (you must include type in the .biaschk card).

You can use a wild card, to describe name and mname, in the biaschk card.

- ? stands for one character.
- * stands for 0 or more characters.

Command Argument	Definition
type	Element type to check MOS (R, C ...) In the first syntax, the *type* can be DIODE, BIPOLAR, BJT, JFET, MOS, NMOS, PMOS, or C. In the second syntax, that specifies a region, *type* must be MOS.

Command Argument	Definition
terminal 1, 2	Terminals, between which HSPICE checks (that is, checks between *terminal1* and *terminal2*): • For MOS level 57: *nd, ng, ns, ne, np, n6* • For MOS level 58: *nd, ngf, ns, ngb* • For MOS level 59: *nd, ng, ns, ne, np* • For other MOS level: *nd, ng, ns, nb* • For capacitor: *n1, n2* • For diode: *np*, *nn* • For bipolar: *nc*, *nb*, *ne*, *ns* • For JFET: *nd*, *ng*, *ns*, *nb*
limit	Biaschk limit that you define. Reports an error, if the bias voltage (between appointed terminals, of appointed elements and models), is larger than the limit.
noise	Biaschk noise that you define. The default is 0.1v. Noise-filter some of the results (the local maximum bias voltage, that is larger than the limit). The next local max replaces the local max, if all of the following conditions are satisfied: local_max-local_min<noise>. next local_max-local_min<noise>. This local max is smaller than the next local max. For a parasitic diode, HSPICE ignores the smaller local max biased voltage, and does not output this voltage. To disable this feature, set the noise detection level to 0.
name	Element name to check.
mname	Model name. HSPICE checks elements of the model for bias.
region	Values can be **cutoff**, **linear**, or **saturation**. HSPICE monitors when the MOS device, defined in the .biaschk command, enters and leaves the specified region (such as **cutoff**).

.CONNECT

SYNTAX:

.CONNECT *node1 node2*

EXAMPLE 1:

```
...
.subckt eye_diagram node1 node2 ...
.connect node1 node2
...
.ends
```

This is now the same as the following:

```
...
.subckt eye_diagram node1 node1 ...
...
.ends
...
```

HSPICE reports the following error message:

```
**error**: subcircuit definition duplicates node
node1
```

To apply any HSPICE statement to *node2*, apply it to *node1* instead. Then, to change the netlist construction to recognize *node2*, use a **.ALTER** statement.

EXAMPLE 2:

```
*example for  .connect
vcc 0 cc 5v
r1  0 1  5k
r2  1 cc 5k
.tran 1n 10n
.print i(vcc) v(1)
.alter
.connect cc 1
.end
```

The first **.TRAN** simulation includes two resistors. Later simulations have only one resistor, because r2 is shorted by connecting cc with 1. v(1) does not print out, but v(cc) prints out instead.

Use multiple **.CONNECT** statements to connect several nodes together.

EXAMPLE 3:

```
.CONNECT node1 node2
.CONNECT node2 node3
```

This example connects both *node2* and *node3* to *node1*. All connected nodes must be in the same subcircuit, or all in the main circuit. The first HSPICE simulation evaluates only *node1*; *node2*, and *node3* are the same node as *node1*. Use **.ALTER** statements to simulate *node2* and *node3*.

If you set **.OPTION NODE**, then HSPICE prints out a node connection table.

DESCRIPTION:

The **.CONNECT** statement connects two nodes in your HSPICE netlist, so that simulation evaluates two nodes as only one node. Both nodes must be at the same level in the circuit design that you are simulating: you cannot connect nodes that belong to different subcircuits.

If you connect node2 to node1, HSPICE does not recognize *node2* at all.

Command Argument	Definition
node1	Name of the first of two nodes to connect to each other.
node2	Name of the second of two nodes to connect to each other. The first node replaces this node in the simulation.

.DATA

SYNTAX:

- Inline **.DATA** statement:

  .DATA *datanm pnam1 <pnam2 pnam3 ... pnamxxx >*
  + pval1<pval2 pval3 ... pvalxxx>
  + pval1' <pval2' pval3' ... pvalxxx'>
  .ENDDATA

- External File **.DATA** statement, for concatenated data files:

  .DATA *datanm* MER
  FILE = '*filename1*' *pname1 = colnum*
  + <pname2 = colnum ...>
  <FILE = '*filename2*' *pname1 = colnum*
  + <pname2 = colnum ...>>
  ...
  <OUT = '*fileout*'>
  .ENDDATA

- Column Laminated **.DATA** statement:

  .DATA *datanm* LAM
  FILE = '*filename1*' *pname1 = colnum*
  + <panme2 = colnum ...>
  <FILE = '*filename2*' *pname1 = colnum*
  + <pname2 = colnum ...>>
  ...
  <OUT = '*fileout*'>
  .ENDDATA

EXAMPLE 1:

```
* Inline .DATA statement
.TRAN       1n    100n           SWEEP DATA = devinf
.AC DEC     10    1hz    10khz   SWEEP DATA = devinf
.DC TEMP   -55    125    10      SWEEP DATA = devinf
.DATA      devinfwidth  length  thresh  cap
  +        50u    30u    1.2v    1.2pf
  +        25u    15u    1.0v    0.8pf
  +        5u     2u     0.7v    0.6pf
.ENDDATA
```

HSPICE performs the above analyses for each set of parameter values defined in the **.DATA** statement. For example, the program first uses the width = 50u, length = 30u, thresh = 1.2v, and cap = 1.2pf parameters to perform **.TRAN**, **.AC**, and **.DC** analyses.

HSPICE then repeats the analyses for width = 25u, length = 15u, thresh = 1.0v, and cap = 0.8pf, and again for the values on each subsequent line in the **.DATA** block.

EXAMPLE 2:

```
* .DATA as the inner sweep
M1 1 2 3 0 N   W = 50u   L = LN
VGS 2 0 0.0v
VBS 3 0 VBS
VDS 1 0 VDS
.PARAM VDS = 0 VBS = 0 L = 1.0u
.DC DATA = vdot
.DATA vdot
          VBS     VDS    L
           0      0.1    1.5u
           0      0.1    1.0u
           0      0.1    0.8u
          -1      0.1    1.0u
          -2      0.1    1.0u
          -3      0.1    1.0u
           0      1.0    1.0u
           0      5.0    1.0u
.ENDDATA
```

This example performs a DC sweep analysis for each set of VBS, VDS, and L parameters in the **.DATA** vdot block. That is, HSPICE runs eight DC analyses, one for each line of parameter values in the **.DATA** block.

EXAMPLE 3:

```
* .DATA as the outer sweep
.PARAM W1 = 50u W2 = 50u L = 1u CAP = 0
.TRAN 1n 100n SWEEP DATA = d1
.DATA d1
          W1      W2     L       CAP
          50u     40u    1.0u    1.2pf
          25u     20u    0.8u    0.9pf
.ENDDATA
```

In this example:

- The default start time for the **.TRAN** analysis is 0.
- The time increment is 1 ns.
- The stop time is 100 ns.

This results in transient analyses at every time value from 0 to 100 ns, in steps of 1 ns, using the first set of parameter values in the **.DATA** d1 block. Then HSPICE reads the next set of parameter values, and performs another 100 transient analyses. It sweeps time from 0 to 100 ns, in 1 ns steps. The outer sweep is time, and the inner sweep varies the parameter values. HSPICE performs two hundred analyses: 100 time increments, times 2 sets of parameter values.

EXAMPLE 4:

```
* External File .DATA, for concatenated data files
.DATA datanm MER
 + FILE = filename1 pname1 = colnum
 + <pname2 = colnum ...>
 + <FILE = filename2 pname1 = colnum
 + <pname2 = colnum ...>>
 + ...
 + <OUT = fileout>
.ENDDATA
```

EXAMPLE 5:

If you concatenate the three files (file1, file2, and file3).

```
file1    file2     file3
a a a    b b b     c c c
a a a    b b b     c c c
a a a
```

The data appears as follows:

```
a a a
a a a
a a a
b b b
b b b
c c c
c c c
```

The number of lines (rows) of data in each file does not need to be the same. The simulator assumes that the associated parameter of each column of the A file is the same as each column of the other files.

The **.DATA** statement for this example is:

```
* External File .DATA statement
.DATA inputdata MER
  FILE = ‘file1’ p1 = 1 p2 = 3 p3 = 4
  FILE = ‘file2’ p1 = 1
  FILE = ‘file3’
.ENDDATA
```

This listing concatenates *file1*, *file2*, and *file3*, to form the *inputdata* dataset. The data in *file1* is at the top of the file, followed by the data in *file2*, and *file3*. The *inputdata* in the **.DATA** statement references the dataname specified in either the **.DC**, **.AC**, or **.TRAN** analysis statements. The parameter fields specify the column that contains the parameters (you must already have defined the parameter names in **.PARAM** statements). For example, the values for the p1 parameter are in column 1 of *file1* and *file2*. The values for the p2 parameter are in column 3 of *file1*.

For data files with fewer columns than others, HSPICE assigns values of zero to the missing parameters.

EXAMPLE 6:

Three files (*D*, *E*, and *F*) contain the following columns of data:

```
File D        File E      File F

d1 d2 d3      e4 e5       f6
d1 d2 d3      e4 e5       f6
d1 d2 d3      e4 e5       f6
```

The laminated data appears as follows:

```
d1 d2 d3      e4 e5       f6
d1 d2 d3      e4 e5       f6
d1 d2 d3      e4 e5       f6
```

The number of columns of data does not need to be the same in the three files.

The number of lines (rows) of data in each file does not need to be the same. HSPICE interprets missing data points as zero.

The **.DATA** statement for this example is:

```
* Column-Laminated .DATA statement
.DATA dataname LAM
   FILE = ‘file1’ p1 = 1 p2 = 2 p3 = 3
   FILE = ‘file2’ p4 = 1 p5 = 2
   OUT = ‘fileout’
.ENDDATA
```

This listing laminates columns from *file1*, and *file2*, into the *fileout* output file. Columns one, two, and three of *file1*, and columns one and two of *file2*, are designated as the columns to place in the output file. You can specify up to 10 files per **.DATA** statement.

If you run HSPICE on a different machine than the one on which the input data files reside (such as when you work over a network), use full path names instead of aliases. Aliases might have different definitions on different machines.

DESCRIPTION:

Data-driven analysis syntax requires a **.DATA** statement, and an analysis statement that contains a DATA = dataname keyword.

You can use the **.DATA** statement to concatenate or column-laminate data sets, to optimize measured I-V, C-V, transient, or s-parameter data.

You can also use the **.DATA** statement for a first or second sweep variable, when you characterize cells, and test worst-case corners. Simulation reads data measured in a lab, such as transistor I-V data, one transistor at a time, in an outer analysis loop. Within the outer loop, the analysis reads data for each transistor (IDS curve, GDS curve, and so on), one curve at a time, in an inner analysis loop.

The **.DATA** statement specifies parameters that change values, and the sets of values to assign during each simulation. The required simulations run as an internal loop. This bypasses reading-in the netlist and setting-up the simulation, which saves computing time. In internal loop simulation, you can also plot simulation results against each other, and print them in a single output.

You can enter any number of parameters in a **.DATA** block. The **.AC**, **.DC**, and **.TRAN** statements can use external and inline data provided in **.DATA** statements. The number of data values per line does not need to correspond to the number of parameters. For example, you do not need to enter 20 values on each line in the **.DATA** block, if each simulation pass requires 20 parameters: the program reads 20 values on each pass, no matter how you format the values.

Each **.DATA** statement can contain up to 50 parameters. If you need more than 50 parameters in a single **.DATA** statement, place 50 or fewer parameters in the **.DATA** statement, and use .ALTER statements for the remaining parameters.

HSPICE refers to **.DATA** statements by their datanames, so each dataname must be unique. HSPICE support three **.DATA** statement formats:

- Inline data, which is parameter data, listed in a **.DATA** statement block. The *datanm* parameter, in a **.DC**, **.AC**, or **.TRAN** analysis statement, calls this statement. The number of parameters that HSPICE reads, determines the number of columns of data. The physical number of data numbers per line does not need to correspond to the number of parameters. For example, if the simulation needs 20 parameters, you do not need 20 numbers per line.

- Data that is concatenated from external files. Concatenated data files are files with the same number of columns, placed one after another.
- Data that is column-laminated from external files. Column lamination means that the columns of files with the same number of rows, are arranged side-by-side.

To use external files with the **.DATA** format:

- Use the MER and LAM keywords to tell HSPICE to expect external file data, rather than inline data.
- Use the FILE keyword to specify the external filename.
- You can use simple file names, such as out.dat, without the single or double quotes (‘ ’ or “ ”), but use the quotes when file names start with numbers, such as “1234.dat”.
- File names are case sensitive on Unix systems.

For data-driven analysis, specify the start time (time 0) in the analysis statement, so analysis correctly calculates the stop time.

The following shows how different types of analysis use **.DATA** statements.

Operating point:

.DC DATA = dataname

DC sweep:

.DC vin 1 5 .25 SWEEP DATA = dataname

AC sweep:

.AC dec 10 100 10meg SWEEP DATA = dataname

TRAN sweep:

.TRAN 1n 10n SWEEP DATA = dataname

Command Argument	Definition
colnum	Column number in the data file, for the parameter value. The column does not need to be the same between files.
datanm	Data name, referenced in the **.TRAN**, **.DC**, or **.AC** statement.
filenamei	Data file to read. HSPICE concatenates files in the order they appear in the **.DATA** statement. You can specify up to 10 files.
fileouti	Data file name, where simulation writes concatenated data. This file contains the full syntax for an inline **.DATA** statement, and can replace the .DATA statement that created it in the netlist. You can output the file, and use it to generate one data file from many.
LAM	Column-laminated (parallel merging) data files to use.
MER	Concatenated (series merging) data files to use.
pnami	Parameter names, used for source value, element value, device size, model parameter value, and so on. You must declare these names in a **.PARAM** statement.
pvali	Parameter value.

SEE ALSO:

.ENDDATA

.DC

SYNTAX:

Sweep or Parameterized Sweep:

.DC *var1* START = *start1* STOP = *stop1* STEP = *incr1*

.DC *var1* START = <*param_expr1*>
+ STOP = <*param_expr2*> STEP = <*param_expr3*>

.DC *var1 start1 stop1 incr1*
+ <SWEEP *var2 type np start2 stop2*>

.DC *var1 start1 stop1 incr1* <*var2 start2 stop2 incr2*>

Data-Driven Sweep:

.DC *var1 type np start1 stop1* <SWEEP DATA = *datanm*>

.DC DATA = *datanm*<SWEEP *var2 start2 stop2 incr2*>

.DC DATA = *datanm*

Monte Carlo:

.DC *var1 type np start1 stop1* <SWEEP MONTE = *val*>

.DC MONTE = *val*

Optimization:

.DC DATA = *datanm* OPTIMIZE = *opt_par_fun*
+ RESULTS = *measnames* MODEL = *optmod*

.DC *var1 start1 stop1* SWEEP OPTIMIZE = OPT*xxx*
+ RESULTS = *measname* MODEL = *optmod*

EXAMPLE 1:

```
.DC VIN 0.25 5.0 0.25
```

This example sweeps the value of the VIN voltage source, from 0.25 volts to 5.0 volts, in increments of 0.25 volts.

EXAMPLE 2:

```
.DC VDS 0 10 0.5 VGS 0 5 1
```

This example sweeps the drain-to-source voltage, from 0 to 10 V, in 0.5 V increments, at VGS values of 0, 1, 2, 3, 4, and 5 V.

EXAMPLE 3:

```
.DC TEMP -55 125 10
```

This example starts a DC analysis of the circuit, from -55°C to 125°C, in 10°C increments.

EXAMPLE 4:

```
.DC TEMP POI 5 0 30 50 100 125
```

This script runs a DC analysis, at five temperatures: 0, 30, 50, 100, and 125°C.

EXAMPLE 5:

```
.DC xval 1k 10k .5k SWEEP TEMP LIN 5 25 125
```

This example runs a DC analysis on the circuit, at each temperature value. The temperatures result from a linear temperature sweep, from 25°C to 125°C (five points), which sweeps a resistor value named *xval*, from 1 k to 10 k, in 0.5 k increments.

EXAMPLE 6:

```
.DC DATA = datanm SWEEP par1 DEC 10 1k 100k
```

This example specifies a sweep of the *par1* value, from 1 k to 100 k, in increments of 10 points per decade.

EXAMPLE 7:

```
.DC par1 DEC 10 1k 100k SWEEP DATA = datanm
```

This example also requests a DC analysis, at specified parameters in the **.DATA** *datanm* statement. It also sweeps the par1 parameter, from 1k to 100k, in increments of 10 points per decade.

EXAMPLE 8:

```
.DC par1 DEC 10 1k 100k SWEEP MONTE = 30
```

This example invokes a DC sweep of the par1 parameter from 1k to 100k by 10 points per decade, using 30 randomly generated (Monte Carlo) values.

EXAMPLE 9:

```
* Schmitt Trigger Example
*file: bjtschmt.spbipolar schmitt trigger
.OPTION post = 2
vcc 6 0 dc 12
vin 1 0 dc 0 pwl(0,0 2.5u,12 5u,0)
cb1 2 4 .1pf
rc1 6 2 1k
rc2 6 5 1k
rb1 2 4 5.6k
rb2 4 0 4.7k
re 3 0 .47k
*
diode 0 1 dmod
q1 2 1 3 bmod 1 ic = 0,8
q2 5 4 3 bmod 1 ic = .5,0.2
*
.dc vin 0,12,.1
.model dmod d is = 1e-15 rs = 10
.model bmod npn is = 1e-15 bf = 80 tf = 1n
+ cjc = 2pf cje = 1pf rc = 50 rb = 100 vaf = 200
.plot v(1) v(5)
.graph dc model = schmittplot input = v(1)
+ output = v(5) 4.0 5.0
```

```
.model schmittplot plot xscal = 1 yscal = 1
+ xmin = .5u xmax = 1.2u
.end
```

DESCRIPTION:

You can use the **.DC** statement in DC analysis, to:

- Sweep any parameter value.
- Sweep any source value.
- Sweep temperature range.
- Perform a DC Monte Carlo (random sweep) analysis.
- Perform a data-driven sweep.
- Perform a DC circuit optimization, for a data-driven sweep.
- Perform a DC circuit optimization, using start and stop.
- Perform a DC model characterization.

The format for the **.DC** statement depends on the application that uses it.

Command Argument	**Definition**
DATA = *datanm*	*Datanm* is the reference name of a **.DATA** statement.
incr1 …	Voltage, current, element, or model parameters; or temperature increments.
MODEL	Specifies the optimization reference name. The **.MODEL OPT** statement uses this name in an optimization analysis

Command Argument	Definition
MONTE = *val*	*val* is the number of randomly-generated values, which you can use to select parameters from a distribution. The distribution can be *Gaussian*, *Uniform*, or *Random Limit.*
np	Number of points per decade or per octave, or just number of points, based on which keyword precedes it.
OPTIMIZE	Specifies the parameter reference name, used for optimization in the **.PARAM** statement
RESULTS	Measure name used for optimization in the **.MEASURE** statement
start1 …	Starting voltage, current, element, or model parameters; or temperature values. If you use the POI (list of points) variation type, specify a list of parameter values, instead of *start stop*.
stop1 …	Final voltage, current, any element, model parameter, or temperature values.
SWEEP	Keyword, to indicate that a second sweep has a different type of variation (DEC, OCT, LIN, POI, or DATA statement; or MONTE = *val*)
TEMP	Keyword, to indicate a temperature sweep.
type	Can be any of the following keywords: • DEC — decade variation • OCT — octave variation • LIN — linear variation • POI — list of points

Command Argument	Definition
var1 …	• Name of an independent voltage or current source, or • Name of any element or model parameter, or • TEMP keyword (indicating a temperature sweep). HSPICE supports a source value sweep, which refers to the source name (SPICE style). However, if you select a parameter sweep, a **.DATA** statement, and a temperature sweep, then you must select a parameter name for the source value. A later .DC statement must refer to this name. The parameter name must not start with V, I, or TEMP.

.DCVOLT

SYNTAX:

.DCVOLT V(*node1*) = *val1* V(*node2*) = *val2* ...

.DCVOLT V *node1 val1 <node2 val2 ...>*

EXAMPLE:

```
.DCVOLT 11 5 4 -5 2 2.2
```

DESCRIPTION:

Use the **.IC** statement, or the **.DCVOLT** statement, to set transient initial conditions in HSPICE How it initializes depends on whether the **.TRAN** analysis statement includes the UIC parameter.

If you specify the UIC parameter in the **.TRAN** statement, HSPICE does not calculate the initial DC operating point, but directly enters transient analysis. Transient analysis uses the **.IC** initialization values as part of the solution, for timepoint zero (calculating the zero timepoint applies a fixed equivalent voltage source). The **.IC** statement is equivalent to specifying the IC parameter on each element statement, but is more convenient. You can still specify the IC parameter, but it does not have precedence over values set in the **.IC** statement.

If you do *not* specify the UIC parameter in the **.TRAN** statement, HSPICE computes the DC operating point solution, before the transient analysis. The node voltages that you specify in the **.IC** statement are fixed, to determine the DC operating point. Transient analysis releases the initialized nodes, to calculate the second and later time points.

Command Argument	Definition
val1 ...	Specifies voltages. The significance of these voltages depends on whether you specify the UIC parameter in the **.TRAN** statement.
node1 ...	Node numbers or names can include full paths, or circuit numbers.

SEE ALSO:

.IC

.DEL LIB

SYNTAX:

.DEL LIB *'<filepath>filename' entryname*

.DEL LIB *libnumber entryname*

EXAMPLE 1:

This example uses an **.ALTER** block.

```
FILE1: ALTER1 TEST CMOS INVERTER
   .OPTION ACCT LIST
   .TEMP 125
   .PARAM WVAL = 15U VDD = 5
   *
   .OP
   .DC VIN 0 5 0.1
   .PLOT DC V(3) V(2)
   *
   VDD 1 0 VDD
   VIN 2 0
   *
   M1 3 2 1 1 P 6U 15U
   M2 3 2 0 0 N 6U W = WVAL
   *

.LIB 'MOS.LIB' NORMAL
.ALTER
   .DEL LIB 'MOS.LIB' NORMAL       $removes LIB from memory

$PROTECTION
.PROT                              $protect statements
                                   $below .PROT
   .LIB 'MOS.LIB' FAST             $get fast model library

.UNPROT
   .ALTER
   .OPTION NOMOD OPTS              $suppress printing model
                                   $parameters and print the
                                   $option summary
   .TEMP -50 0 50                  $run with different
                                   $temperatures
   .PARAM WVAL = 100U VDD = 5.5     $change the parameters
   VDD 1 0 5.5                     $using VDD 1 0 5.5 to
                                   $change the power supply
                                   $VDD value doesn't
                                   $work
```

```
VIN 2 0 PWL 0NS 0 2NS 5 4NS 0 5NS 5
                            $change the input
                            $source
.OP VOL                     $node voltage table of
                            $operating points
.TRAN 1NS 5NS               $run with transient
                            $also

M2 3 2 0 0 N 6U WVAL        $change channel width
.MEAS SW2 TRIG V(3) VAL = 2.5 RISE = 1 TARG V(3)
+ VAL = VDD CROSS = 2       $measure output
*
.END
```

Example 1 calculates a DC transfer function for a CMOS inverter.

1. First, HSPICE simulates the device, using the NORMAL inverter model from the MOS.LIB library.
2. Using the **.ALTER** block and the **.LIB** command, HSPICE substitutes a faster CMOS inverter, FAST, for NORMAL.
3. HSPICE then resimulates the circuit.
4. Using the second **.ALTER** block, HSPICE executes DC transfer analysis simulations at three different temperatures, and with an n-channel width of 100 mm, instead of 15 mm.
5. HSPICE also runs a transient analysis, in the second **.ALTER** block. Use the **.MEASURE** statement to measure the rise time of the inverter.

EXAMPLE 2:

This example uses an **.ALTER** block.

```
FILE2: ALTER2.SP CMOS INVERTER USING SUBCIRCUIT
.OPTION LIST ACCT

.MACRO INV 1 2 3
M1 3 2 1 1 P 6U 15U
M2 3 2 0 0 N 6U 8U
.LIB 'MOS.LIB' NORMAL
.EOM INV

XINV 1 2 3 INV
VDD 1 0 5
VIN 2 0
.DC VIN 0 5 0. 1
.PLOT V(3) V(2)

.ALTER
.DEL LIB 'MOS.LIB' NORMAL
.TF V(3) VIN            $DC small-signal transfer
                        $function

*
.MACRO INV 1 2 3        $change data within
                        $subcircuit def
M1 4 2 1 1 P 100U 100U  $change channel
                        $length,width,also
                        $topology
M2 4 2 0 0 N 6U   8U    $change topology
R4 4 3 100              $add the new element
C3 3 0 10P              $add the new element
.LIB 'MOS.LIB' SLOW     $set slow model library

$.INC 'MOS2.DAT'        $not allowed to be used
                        $inside subcircuit, allowed
                        $outside subcircuit
.EOM INV

.END
```

In this example, the **.ALTER** block adds a resistor and capacitor network to the circuit. The network connects to the output of the inverter, and HSPICE simulates a DC small-signal transfer function.

DESCRIPTION:

Use the **.DEL LIB** statement to remove library data from memory. The next time you run a simulation, the **.DEL LIB** statement removes the **.LIB** call statement, with the same library number and entry name, from memory. You can then use a **.LIB** statement to replace the deleted library.

You can use the **.DEL LIB** statement with the **.ALTER** statement.

Command Argument	Definition
entryname	Entry name, used in the library call statement to delete.
filename	Name of a file to delete from the data file. The file path, plus the file name, can be up to 256 characters long. You can use any file name that is valid for the operating system that you use. Enclose the file path and file name in single or double quote marks.
filepath	Path name of a file, if the operating system supports tree-structured directories.
libnumber	Library number, used in the library call statement to delete.

.DISTO

SYNTAX:

.DISTO *Rload <inter <skw2 <refpwr <spwf>>>>*

EXAMPLE:

```
.DISTO RL 2 0.95 1.0E-3 0.75
```

DESCRIPTION:

The **.DISTO** statement computes the distortion characteristics of the circuit in an AC small-signal, sinusoidal, steady-state analysis.

The program computes and reports five distortion measures at the specified load resistor. The analysis assumes that the input uses one or two signal frequencies.

- HSPICE uses the first frequency (F1, the nominal analysis frequency) to calculate harmonic distortion. The .AC statement frequency-sweep sets it.
- HSPICE uses the optional second input frequency (F2) to calculate intermodulation distortion. To set it implicitly, specify the skw2 parameter, which is the F2/F1 ratio

HSPICE performs only one distortion analysis per simulation. If your design contains more than one **.DISTO** statement, HSPICE runs only the last statement.
The **.DISTO** statement calculates distortions for diodes, BJTs (levels 1, 2, 3, and 4), and MOSFETs (Level49 and Level53, Version 3.22).

Command Argument	Definition
Rload	The resistor element name of the output load resistor, into which the output power feeds.
refpwr	Reference power level, used to compute the distortion products. If you omit *refpwr*, the default value is 1mW, measured in decibels magnitude (dbM). The value must be ≥ *1e-10*.
skw2	Ratio of the second frequency (F2) to the nominal analysis frequency (F1), in the range *1e-3* < *skw2* < *0.999*. If you omit *skw2*, the default value is 0.9.
spwf	Amplitude of the second frequency (F2). The value must be ≥ *1e-3*. Default = 1.0.
inter	Interval at which HSPICE prints a distortion-measure summary. Specifies a number of frequency points in the AC sweep (see the *np* parameter in the .AC command). • If you omit *inter*, or set it to zero, HSPICE does not print a summary. To print or plot the distortion measures, use the .PRINT or .PLOT statement. • If you set inter to 1 or higher, HSPICE prints a summary of the first frequency, and of each subsequent inter-frequency increment. To obtain a summary printout for only the first and last frequencies, set inter equal to the total number of increments needed, to reach fstop in the .AC statement. For a summary printout of only the first frequency, set inter to greater than the total number of increments required, to reach *fstop*. HSPICE prints an extensive summary from the distortion analysis, for each frequency listed. Use the *inter* parameter in the **.DISTO** statement to limit the amount of output generated.

.DISTO Value	Description
DIM2	Intermodulation distortion, first difference. Relative magnitude and phase of the frequency component (F1 - F2).
DIM3	Intermodulation distortion, second difference. The relative magnitude and phase of the frequency component (2 · F1 - F2).
HD2	Second-order harmonic distortion. Relative magnitude and phase of the frequency component 2 · F1 (ignores F2).
HD3	Third-order harmonic distortion. Relative magnitude and phase of the frequency component 3 · F1 (ignores F2).
SIM2	Intermodulation distortion, sum. Relative magnitude and phase of the frequency component (F1 + F2).

SEE ALSO:

.AC

.DOUT

SYNTAX:

.DOUT *nd* VTH (*time state* < *time state* >)

.DOUT *nd* VLO VHI (*time state* < *time state* >)

EXAMPLE:

```
.PARAM VTH = 3.0
.DOUT node1 VTH(0.0n 0 1.0n 1
+ 2.0n X 3.0n U 4.0n Z 5.0n 0)
```

The **.PARAM** statement in this example sets the VTH variable value to 3. The **.DOUT** statement, operating on the node1 node, uses VTH as its threshold voltage.

When *node1* is above 3V, it is a logic 1; otherwise, it is a logic 0.

- At 0ns, the expected state of *node1* is logic-low.
- At 1ns, the expected state is logic-high.
- At 2ns, 3ns, and 4ns, the expected state is “do not care”.
- At 5ns, the expected state is again logic low.

DESCRIPTION:

The digital output (**.DOUT**) statement specifies the expected final state of an output signal, in HSPICE.

During simulation, HSPICE compares simulation results with the expected output. If the states are different, HSPICE reports an error.

Command Argument	Definition
nd	Node name.
time	Absolute timepoint.
state	Expected condition of the *nd* node at the specified *time:* • 0 expect ZERO,LOW. • 1 expect ONE,HIGH. • else Don't care.
VTH	Single voltage threshold.
VLO	Voltage of the logic-low state.
VHI	Voltage of the logic-high state.

:

.DOUT State Value	Description
0	expect ZERO
1	expect ONE
X, x	do not care
U, u	do not care
Z, z	expect HIGH IMPEDANCE (don't care)

SEE ALSO:

.GRAPH
.MEASURE
.PLOT
.PRINT
.PROBE
.STIM

.EBD

SYNTAX:

.EBD *ebdname*
+ file = ’*filename*’
+ model = ’*modelname*’
+ component = ’*compname*:*reference_designator*’
+ {component = ’*compname*:*reference_designator*’...}

EXAMPLE:

```
.ebd ebd
  + file = ’test.ebd’
  + model = ’16Meg X 8 SIMM Module’
  + component = ’cmpnt:u21’
.ibis cmpnt
  + file = ’ebd.ibs’
  + component = ’SIMM’
  + hsp_ver = 2003.09 nowarn
```

This example corresponds to the following *.ebd* file:

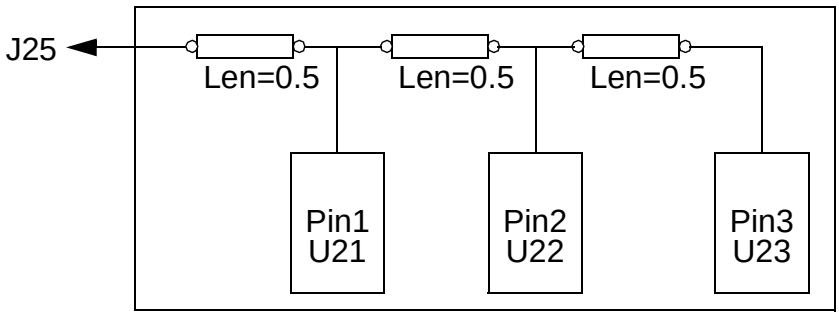
```
  .....................
  [Begin Board Description] 16Meg X 8 SIMM Module
  ...................
  [Pin List] signal_name
  J25          POWER5
  [Path Description] CAS_2
  Pin J25
  Len = 0.5 L=8.35n C=3.34p R=0.01 /
  Node u21.1
  Len = 0.5 L=8.35n C=3.34p R=0.01 /
  Node u22.2
  Len = 0.5 L=8.35n C=3.34p R=0.01 /
  Node u23.3
```

DESCRIPTION:

The **.EBD** command provides the IBIS(V 3.2) EBD feature. HSPICE and Star-Sim simulators use the *.ebd* file when simulating the line connected with the u21 reference_designator. The format of node names is *ebdname_SignalName*. For example, the format of a node name called J25 is `ebd_POWER5` (see Figure 2-1).

Command Argument	**Definition**
compname	Name of a *.ibs* file that describes a component.
reference_designator	Reference designator that maps the component.

Figure 2-1 Circuit Connection for EBD Example



.ELSE

SYNTAX:

.IF (*condition1***)**
...
<.ELSEIF (*condition2***) >**
...
<.ELSE>
...
.ENDIF

EXAMPLE:

```
.IF a=b
.INCLUDE /myhome/subcircuits/diode_circuit1
...
.ELSEIF a=c
.INCLUDE /myhome/subcircuits/diode_circuit2
...
.ELSE
.INCLUDE /myhome/subcircuits/diode_circuit3
...
.ENDIF
```

DESCRIPTION:

.ELSE precedes one or more commands in a conditional block. HSPICE executes these commands by default, if the conditions in the preceding **.IF** statement, and in all of the preceding **.ELSEIF** statements in the same conditional block, are all false.

SEE ALSO:

.ELSEIF
.ENDIF
.IF

.ELSEIF

SYNTAX:

.IF (*condition1***)**
...
<.ELSEIF (*condition2***) >**
...
<.ELSE>
...
.ENDIF
.ELSE

EXAMPLE:

```
.IF a=b
.INCLUDE /myhome/subcircuits/diode_circuit1
...
.ELSEIF a=c
.INCLUDE /myhome/subcircuits/diode_circuit2
...
.ELSE
.INCLUDE /myhome/subcircuits/diode_circuit3
...
.ENDIF
```

DESCRIPTION:

HSPICE executes the commands that follow the first **.ELSEIF** statement, only if *condition1* in the preceding **.IF** statement is false, and condition2 in the first **.ELSEIF** statement is true.

If condition1 in the **.IF** statement and *condition2* in the first **.ELSEIF** statement are both false, then HSPICE moves on to the next **.ELSEIF** statement. If this second **.ELSEIF** condition is true, HSPICE executes the commands that follow the second **.ELSEIF** statement, instead of the commands after the first **.ELSEIF** statement.

Command Argument	Definition
condcition2	Condition that must be true, before HSPICE executes the commands that follow the **.ELSEIF** statement.

SEE ALSO:

.ELSE
.ENDIF
.IF

.END

SYNTAX:

.END <*comment*>

EXAMPLE:

```
MOS OUTPUT
   .OPTION NODE NOPAGE
   VDS 3 0
   VGS 2 0
   M1 1 2 0 0 MOD1 L = 4U W = 6U AD = 10P AS = 10P
   .MODEL MOD1 NMOS VTO = -2 NSUB = 1.0E15
TOX = 1000
   + UO = 550
   VIDS 3 1
   .DC   VDS 0 10 0.5    VGS 0 5 1
   .PRINT DC I(M1) V(2)
.END MOS OUTPUT

MOS CAPS
   .OPTION SCALE = 1U SCALM = 1U WL ACCT
   .OP
   .TRAN .1 6
   V1 1 0 PWL 0 -1.5V 6 4.5V
   V2 2 0 1.5VOLTS
   MODN1 2 1 0 0 M 10 3

   .MODEL M NMOS VTO = 1 NSUB = 1E15 TOX = 1000
   + UO = 800 LEVEL = 1 CAPOP = 2

   .PLOT TRAN V(1) (0,5) LX18(M1) LX19(M1) LX20(M1)
   + (0,6E-13)
.END MOS CAPS
```

DESCRIPTION:

An **.END** statement must be the last statement in the input netlist file. The period preceding **END** is a required part of the statement.

Any text that follows the **.END** statement is a comment, and has no effect on that simulation.

An input file that contains more than one simulation run must include an **.END** statement for each simulation run. You can concatenate several simulations into a single file.

Command Argument	**Definition**
<*comment*>	Can be any comment. Typically, the comment is the name of the netlist file, or of the simulation run, that this command terminates.

.ENDDATA

SYNTAX:

.ENDDATA

DESCRIPTION:

Use the **.ENDDATA** statement to end a .DATA block in an HSPICE input netlist.

SEE ALSO:

.DATA

.ENDIF

SYNTAX:

.IF (*condition1***)**
...
<.ELSEIF (*condition2***) >**
...
<.ELSE>
...
.ENDIF
.ELSE

EXAMPLE:

```
.IF a=b
.INCLUDE /myhome/subcircuits/diode_circuit1
...
.ELSEIF a=c
.INCLUDE /myhome/subcircuits/diode_circuit2
...
.ELSE
.INCLUDE /myhome/subcircuits/diode_circuit3
...
.ENDIF
```

DESCRIPTION:

This command ends a conditional block of commands that begins with an **.IF** statement.

SEE ALSO:

.ELSE
.ELSEIF
.IF

.ENDL

SYNTAX:

.ENDL

DESCRIPTION:

Use the **.ENDL** statement to end a .LIB statement in an HSPICE input netlist.

SEE ALSO:

.LIB

.ENDS

SYNTAX:

.ENDS <*SUBNAME*>

EXAMPLE 1:

```
.ENDS mos_circuit
```

This example terminates a subcircuit named *mos_circuit*.

EXAMPLE 2:

```
.ENDS
```

If you omit the subcircuit name, as in this second example, this statement terminates all subcircuit definitions that begin with a **.SUBCKT** statement.

DESCRIPTION:

Use the .ENDS statement to terminate a **.SUBCKT** statement.

This statement must be the last for any subcircuit definition that starts with a **.SUBCKT** command.

You can nest subcircuit references (calls) within subcircuits, in HSPICE.

Command Argument	Definition
<*SUBNAME*>	Name of the subcircuit description to terminate, that begins with a **.SUBCKT** command.

SEE ALSO:

.SUBCKT

.EOM

SYNTAX:

.EOM <*SUBNAME*>

EXAMPLE 1:

```
.EOM diode_circuit
```

This example terminates a subcircuit named `diode_circuit`.

EXAMPLE 2:

```
.EOM
```

If you omit the subcircuit name, as in this second example, this statement terminates all subcircuit definitions that begin with a **.MACRO** statement.

DESCRIPTION:

Use the **.EOM** statement to terminate a **.MACRO** statement.

This statement must be the last for any subcircuit definition that starts with a **.MACRO** command.

You can nest subcircuit references (calls) within subcircuits, in HSPICE.

Command Argument	Definition
<*SUBNAME*>	Name of the subcircuit description to terminate, that begins with a **.SUBCKT** command.

SEE ALSO:

.MACRO

.FFT

SYNTAX:

.FFT <*output_var*> <START=*value*> <STOP=*value*>
+ <NP=*value*> <FORMAT=*keyword*>
+ <WINDOW=*keyword*> <ALFA=*value*>
+ <FREQ=*value*> <FMIN=*value*> <FMAX=*value*>

EXAMPLE 1:

```
.FFT v(1)

.FFT v(1,2) np=1024 start=0.3m stop=0.5m freq=5.0k
+ window=kaiser alfa=2.5

.FFT I(rload) start=0m to=2.0m fmin=100k fmax=120k
+ format=unorm

.FFT par('v(1) + v(2)') from=0.2u stop=1.2u
+ window=harris
```

EXAMPLE 2:

```
.FFT v(1) np=1024

.FFT v(2) np=1024
```

This example generates an .ft0 file for the FFT of v(1), and an .ft1 file for the FFT of v(2).

DESCRIPTION:

The **.FFT** statement uses internal time point values to calculate the Discrete Fourier Transform (DFT) value, which HSPICE uses for spectrum analysis. A DFT uses sequences of time values to determine the frequency content of analog signals, in circuit simulation.

You can specify only one output variable in an **.FFT** command. The following is an incorrect use of the command, because it contains two variables in one **.FFT** command:

```
.FFT v(1) v(2) np=1024
```

Command Argument	Definition
output_var	Can be any valid output variable, such as voltage, current, or power.
START	Start of the output variable waveform to analyze. Defaults to the START value in the .TRAN statement, which defaults to 0.
FROM	An alias for START in .FFT statements.
STOP	End of the output variable waveform to analyze. Defaults to the TSTOP value in the .TRAN statement.
TO	An alias for STOP, in .FFT statements
NP	Number of points to use in the FFT analysis. NP must be a power of 2. If NP is not a power of 2, HSPICE automatically adjusts it to the closest higher number that is a power of 2. Default=1024.
FORMAT	Specifies the output format: • NORM= normalized magnitude (default) • UNORM=unnormalized magnitude

Command Argument	Definition
WINDOW	Specifies the window type to use: • RECT=simple rectangular truncation window (default). • BART=Bartlett (triangular) window. • HANN=Hanning window. • HAMM=Hamming window. • BLACK=Blackman window. • HARRIS=Blackman-Harris window. • GAUSS=Gaussian window. • KAISER=Kaiser-Bessel window.
ALFA	Parameter to use in GAUSS and KAISER windows, to control the highest side-lobe level, bandwidth, and so on. 1.0 <= ALFA <= 20.0 Default=3.0
FREQ	Frequency to analyze. If FREQ is non-zero, the output lists only the harmonics of this frequency, based on FMIN and FMAX. HSPICE also prints the THD for these harmonics. Default=0.0 (Hz).
FMIN	Minimum frequency for which HSPICE prints FFT output into the listing file. THD calculations also use this frequency. T = (STOP-START) Default=1.0/T (Hz).
FMAX	Maximum frequency for which HSPICE prints FFT output into the listing file. THD calculations also use this frequency. Default=0.5*NP*FM IN (Hz).

SEE ALSO:

.TRAN

.FOUR

SYNTAX:

.FOUR *freq ov1 <ov2 ov3 ...>*

EXAMPLE:

```
.FOUR 100K V(5)
```

DESCRIPTION:

This statement performs a Fourier analysis, as part of the transient analysis. You can use the **.FOUR** statement in HSPICE perform the Fourier analysis over the interval (tstop-fperiod, tstop), where:

- tstop is the final time, specified for the transient analysis.
- fperiod is a fundamental frequency period (*freq* parameter).

HSPICE performs Fourier analysis on 501 points of transient analysis data on the last 1/f time period, where f is the fundamental Fourier frequency. HSPICE interpolates transient data, to fit on 501 points, running from (tstop-1/f) to tstop.

To calculate the phase, the normalized component, and the Fourier component, HSPICE uses 10 frequency bins. The Fourier analysis determines the DC component, and the first nine AC components. For improved accuracy, the **.FOUR** statement can use non-linear, instead of linear, interpolation.

Command Argument	**Definition**
freq	Fundamental frequency
ov1 …	Output variables to analyze.

SEE ALSO:
.TRAN

.FSOPTIONS

SYNTAX:

.FSOPTIONS *name* <ACCURACY=LOW|MEDIUM|HIGH>
+ <GRIDFACTOR=*val*> <PRINTDATA=YES|NO>
+ <COMPUTEG0=YES|NO> <COMPUTEGD=YES|NO>
+ <COMPUTERO=YES|NO> <COMPUTERS=YES|NO>

DESCRIPTION:

Use the **.FSOPTIONS** statement to set various options for the field solver. The following rules apply to the Field Solver when specifying options with the **.FSOPTIONS** statement:

- The field solver always computes the L and C matrices.
- If COMPUTERS=YES, then the field solver starts, and calculates Lo, Ro, and Rs.
- For each accuracy mode, the field solver uses either the pre-defined number of segments, or the number of segments that you specified. It then multiplies this number times the GRIDFACTOR, to obtain the final number of segments.

Because a wide range of applications are available, the pre-defined accuracy level might not be accurate enough for some applications. If you need a higher accuracy than the value that the HIGH option sets, then increase either the GRIDFACTOR value, or the N/NH/NW values, to increase the mesh density.

Command Argument	Definition
name	Option name.
ACCURACY	Sets the solver accuracy to one of the following: • LOW • MEDIUM • HIGH
GRIDFACTOR	Multiplication factor (integer) to determine the final number of segments used to define the shape. If you set COMPUTERS=yes, the field solver does not use this parameter to compute Ro and Rs values.
PRINTDATA	The solver prints output matrices.
COMPUTEGO	The solver computes the static conductance matrix.
COMPUTEGD	The solver computes the dielectric loss matrix.
COMPUTERO	The solver computes the DC resistance matrix.
COMPUTERS	The solver computes the skin-effect resistance matrix. This parameter uses the filament method solver to compute R_0 and R_S.

SEE ALSO:

.LAYERSTACK
.MATERIAL
.SHAPE

.GLOBAL

SYNTAX:

.GLOBAL *node1 node2 node3* ...

EXAMPLE:

This example shows global definitions for VDD and `input_sig` nodes.

```
.GLOBAL VDD input_sig
```

DESCRIPTION:

The **.GLOBAL** statement globally assigns a node name, in HSPICE. This means that all references to a global node name, used at any level of the hierarchy in the circuit, connect to the same node.

The most common use of a **.GLOBAL** statement is if your netlist file includes subcircuits. This statement assigns a common node name to subcircuit nodes. Another common use of **.GLOBAL** statements is to assign power supply connections of all subcircuits. For example, **.GLOBAL** VCC connects all subcircuits with the internal node name VCC.

Ordinarily, in a subcircuit, the node name consists of the circuit number, concatenated to the node name. When you use a **.GLOBAL** statement, HSPICE does not concatenate the node name with the circuit number, and assigns only the global name. You can then exclude the power node name in the subcircuit or macro call.

Command Argument	Definition
node1 *node2*	Name of a global nodes, such as supply and clock names; overrides local subcircuit definitions.

.GRAPH

SYNTAX:

.GRAPH *antype* <MODEL = *mname*> <*unam1* = > *ov1*,
+ <*unam2* = >*ov2* ... <*unamn* = >*ovn* (*plo*,*phi*)

EXAMPLE:

```
.GRAPH DC cgb = lx18(m1) cgd = lx19(m1)
+ cgs = lx20(m1)
.GRAPH DC MODEL = plotbjt
+ model_ib = i2(q1)    meas_ib = par(ib)
+ model_ic = i1(q1)    meas_ic = par(ic)
+ model_beta = par('i1(q1)/i2(q1)')
+ meas_beta = par('par(ic)/par(ib)')(1e-10,1e-1)
.MODEL plotbjt PLOT MONO = 1 YSCAL = 2 XSCAL = 2
+ XMIN = 1e-8 XMAX = 1e-1
```

DESCRIPTION:

Use the **.GRAPH** statement when you need high-resolution plots of HSPICE simulation results. You cannot use **.GRAPH** statements in the PC version of HSPICE.

Each **.GRAPH** statement creates a new *.gr#* file, where # ranges first from 0 to 9, and then from a to z. You can create up to 10000 graph files.

You can include wildcards in **.GRAPH** statements.

Command Argument	Definition
antype	Type of analysis for the specified plots (outputs). Analysis types are: DC, AC, TRAN, NOISE, or DISTO.
mname	Plot model name, referenced in the **.GRAPH** statement. Use **.GRAPH** and its plot name to create high-resolution plots directly from HSPICE.
unam1…	You can define output names, which correspond to the ov1 ov2 … output variables (*unam1 unam2 ...*), and use them as labels, instead of output variables, for a high resolution graphic output.
ov1 …	Output variables to print. Can be voltage, current, or element template variables, from a different type of analysis. You can also use algebraic expressions as output variables, but you must define them inside the PAR( ) statement.
plo, phi	Lower and upper plot limits. Set the plot limits only at the end of the **.GRAPH** statement.

SEE ALSO:

.DOUT
.MEASURE
.PLOT
.PRINT
.PROBE
.STIM

.IBIS

SYNTAX:

.IBIS cname *keyword_1 = value_1* ...
+ [*keyword_M= value_M*]

DESCRIPTION:

This is the general syntax for the **.IBIS** command when used with a component. The optional keywords are in square brackets.

Command Argument	Definition
cname	Instance name of this ibis command
keyword_i= value_i	Assigns the *value_i* value, to the *keyword_i* keyword. Optional keywords are in the square brackets.

SEE ALSO:

.EBD
.PKG

.IC

SYNTAX:

.IC V(*node1*) = *val1* V(*node2*) = *val2* ...

EXAMPLE:

```
.IC V(11) = 5 V(4) = -5 V(2) = 2.2
```

DESCRIPTION:

Use the **.IC** statement, or the **.DCVOLT** statement, to set transient initial conditions in HSPICE How it initializes depends on whether the **.TRAN** analysis statement includes the UIC parameter.

If you specify the UIC parameter in the **.TRAN** statement, HSPICE does not calculate the initial DC operating point, but directly enters transient analysis. Transient analysis uses the **.IC** initialization values as part of the solution, for timepoint zero (calculating the zero timepoint applies a fixed equivalent voltage source). The **.IC** statement is equivalent to specifying the IC parameter on each element statement, but is more convenient. You can still specify the IC parameter, but it does not have precedence over values set in the **.IC** statement.

If you do *not* specify the UIC parameter in the **.TRAN** statement, HSPICE computes the DC operating point solution, before the transient analysis. The node voltages that you specify in the **.IC** statement are fixed, to determine the DC operating point. Transient analysis releases the initialized nodes, to calculate the second and later time points.

Command Argument	Definition
val1 ...	Specifies voltages. The significance of these voltages depends on whether you specify the UIC parameter in the **.TRAN** statement.
node1 ...	Node numbers or names can include full paths, or circuit numbers.

SEE ALSO:

.DCVOLT
.TRAN

.IF

SYNTAX:

.IF (*condition1***)**
...
<.ELSEIF (*condition2***) >**
...
<.ELSE>
...
.ENDIF
.ELSE

EXAMPLE:

```
.IF a=b
.INCLUDE /myhome/subcircuits/diode_circuit1
...
.ELSEIF a=c
.INCLUDE /myhome/subcircuits/diode_circuit2
...
.ELSE
.INCLUDE /myhome/subcircuits/diode_circuit3
...
.ENDIF
```

DESCRIPTION:

.IF marks the start of a conditional block in a netlist. HSPICE executes the commands that follow an **.IF** statement, only if *condition1* is true. If *condition1* is false, HSPICE moves on to the first **.ELSEIF** statement. If this **.ELSEIF** condition is true, HSPICE executes the commands that follow the **.ELSEIF** statement, instead of the commands after the **.IF** statement.

Command Argument	Definition
condcition1	Condition that must be true, before HSPICE executes the commands that follow the **.IF** statement.

SEE ALSO:

.ELSE
.ELSEIF
.ENDIF

.INCLUDE

SYNTAX:

.INCLUDE '*<filepath> filename*'

EXAMPLE:

```
.INCLUDE /myhome/subcircuits/diode_circuit
```

DESCRIPTION:

You can include a netlist as a subcircuit in one or more other netlists. To include another netlist in the current netlist, use the **.INCLUDE** statement.

Command Argument	Definition
filepath	Path name of a file, for computer operating systems that support tree-structured directories. A .INC file can contain nested .INC calls to itself, or to another .INC file. If you use a relative path in a nested .INC call, the path starts from the directory of the parent .INC file, not from the work directory. If the path starts from the work directory, HSPICE can also find the .INC file, but prints a warning.
filename	Name of a file to include in the data file. The file path, plus the file name, can be up to 1024 characters long. You can use any valid file name for the computer's operating system. You *must* enclose the file path and name in single or double quotation marks.

.LAYERSTACK

SYNTAX:

.LAYERSTACK *sname* <BACKGROUND=*mname*>
+ <LAYER=(*mname*,*thickness*) ...>

DESCRIPTION:

A layer stack defines a stack of dielectric or metal layers.

You must associate each transmission line system with *one*, and *only* one, layer stack. However, you can associate a single-layer stack with *many* transmission line systems.

In the layer stack:

- Layers are listed from bottom to top.
- Metal layers (ground planes) are located only at the bottom, only at the top, or both at the top and bottom.
- Layers are stacked in the y-direction, and the bottom of a layer stack is at y=0.
- All conductors must be located above y=0.
- Background material must be dielectric.

The following limiting cases apply to the **.LAYERSTACK** command:

- Free space without ground:
  ```
  .LAYERSTACK mystack
  ```
- Free space with a (bottom) ground plane:
  ```
  .LAYERSTACK halfSpace PEC 0.1mm
  ```

Command Argument	Definition
sname	Layer stack name.
mname	Material name.
BACKGROUND	Background dielectric material name. By default, the Field Solver assumes AIR for the background.
thickness	Layer thickness.

SEE ALSO:

.FSOPTIONS
.MATERIAL
.SHAPE

.LIB

SYNTAX:

Use the following syntax for library calls:

.LIB '*<filepath> filename*' *entryname*

Use the following syntax to define library files:

.LIB *entryname1*
. $ ANY VALID SET OF HSPICE STATEMENTS
.ENDL *entryname1*
.LIB *entryname2*
.
. $ ANY VALID SET OF HSPICE STATEMENTS
.ENDL *entryname2*
.LIB *entryname3*
.
. $ ANY VALID ET OF HSPICE STATEMENTS
.ENDL *entryname3*

EXAMPLE 1:

```
* Library call
.LIB 'MODELS' cmos1
```

EXAMPLE 2:

```
.LIB MOS7
$ Any valid set of HSPICE commands
.
.
.
.ENDL MOS7
```

EXAMPLE 3:

The following are an illegal example and a legal example of nested .LIB statements for the *file3* library.

Illegal:

```
.LIB MOS7
...
.LIB 'file3' MOS7 $ This call is illegal in MOS7
library
...
...
.ENDL
```

Legal:

```
.LIB MOS7
...
.LIB 'file1' MOS8
.LIB 'file2' MOS9
.LIB CTT $ file2 is already open for the CTT
  $ entry point
.ENDL
```

EXAMPLE 4:

```
.LIB TT
$TYPICAL P-CHANNEL AND N-CHANNEL CMOS LIBRARY
$ PROCESS: 1.0U CMOS, FAB7
$ following distributions are 3 sigma ABSOLUTE
GAUSSIAN
.PARAM TOX = AGAUSS(200,20,3)$ 200 angstrom +/- 20a
+ XL = AGAUSS(0.1u,0.13u,3)$ polysilicon CD
+ DELVTON = AGAUSS(0.0,.2V,3)$ n-ch threshold
change
+ DELVTOP = AGAUSS(0.0,.15V,3)
  $ p-ch threshold change
.INC '/usr/meta/lib/cmos1_mod.dat'
  $ model include file
.ENDL TT
```

```
.LIB FF
$HIGH GAIN P-CH AND N-CH CMOS LIBRARY 3SIGMA VALUES
.PARAM TOX = 220 XL = -0.03 DELVTON = -.2V
+ DELVTOP = -0.15V
.INC '/usr/meta/lib/cmos1_mod.dat'
   $ model include file
.ENDL FF
```

This example is a **.LIB** of model skew parameters, and features both worst-case and statistical distribution data. The statistical distribution median value is the default, for all non-Monte Carlo analysis. The model is in the /usr/meta/lib/cmos1_mod.dat include file.

```
.MODEL NCH NMOS LEVEL = 2 XL = XL TOX = TOX
+ DELVTO = DELVTON .....

.MODEL PCH PMOS LEVEL = 2 XL = XL TOX = TOX
+ DELVTO = DELVTOP .....
```

The model keyword (left side) equates to the skew parameter (right side). A model keyword can be the same as a skew parameter.

DESCRIPTION:

To create and read from libraries of commonly-used commands, device models, subcircuit analysis, and statements (library calls) in library files, use the **.LIB** call statement. As HSPICE encounters each **.LIB** call name in the main data file, it reads the corresponding entry from the designated library file, until it finds an **.ENDL** statement.

You can also place a **.LIB** call statement in an **.ALTER** block.

To build libraries (library file definition), use the **.LIB** statement in a library file. For each macro in a library, use a library definition statement (.LIB entryname) and an **.ENDL** statement.

The **.LIB** statement begins the library macro, and the **.ENDL** statement ends the library macro. The text after a library file entry name must consist of HSPICE statements.

Library calls can call other libraries (nested library calls), if they are different files. You can nest library calls to any depth. Use nesting with the .ALTER statement, to create a sequence of model runs. Each run can consist of similar components, using different model parameters, without duplicating the entire input file.

The simulator uses the .LIB statement and the .INCLUDE statement, to access the models and skew parameters. The library contains parameters that modify **.MODEL** statements.

Command Argument	Definition
filepath	Path to a file. Used where a computer supports tree-structured directories. When the LIB file (or alias) is in the same directory where you run HSPICE, you do not need to specify a directory path; the netlist runs on any machine. Use the "../" syntax in the filepath, to designate the parent directory of the current directory.
entryname	Entry name, for the section of the library file to include. The first character of an entryname cannot be an integer.
filename	Name of a file to include in the data file. The combination of filepath plus filename can be up to 256 characters long, structured as any filename that is valid for the computer's operating system. Enclose the file path and file name in single or double quotation marks. Use the "../" syntax in the filename, to designate the parent directory of the current directory.

SEE ALSO:

.ENDL

.LIN

SYNTAX:

.LIN <sparcalc = [1|0] <modelname = ...>>
+ <filename = ...> <format=[selem|citi|touchstone]>
+ <noisecalc = [1|0] <gdcalc = [1|0]>

EXAMPLE:

```
.LIN sparcalc=1 modelname=my_custom_model
filename=mydesign format=touchstone noisecalc=1
gdcalc=1
```

This example extracts noise and linear transfer parameters for a general multi-port network, performs 2-port noise analysis, and performs group delay analysis, for a model named *my_custom_model*. The output is in TOUCHSTONE format, in the *mydesign* output file.

DESCRIPTION:

The **.LIN** command extracts noise and linear transfer parameters for a general multi-port network.

When used with the **.AC** command, **.LIN** makes available a broad set of linear port-wise measurements:

- Multi-port scattering [S] parameters.
- Noise parameters.
- Stability factors.
- Gain factors.
- Matching coefficients.

The **.LIN** command computes the S (scattering), Y (admittance), and Z (impedance) parameters directly, based on the location of the port (P) elements in your circuit, and the specified values for their reference impedances. For example, the following port element specifications identify 50-ohm reference impedances, at the input and output.

Command Argument	Definition
sparcalc	If 1, extract S parameters (default).
modelname	Model name listed in the **.MODEL** statement, in the .sc# model output file.
filename	Output file name (default=netlist name).
format	Output file format: • selem is for S element .sc# format, which you can include in the netlist. • citi is CITI file format. • touchstone is TOUCHSTONE file format.
noisecalc	If 1, extract noise parameters (perform 2-port noise analysis). Default=0.
gdcalc	If 1, extract group delay (perform group delay analysis). Default=0.

.LOAD

SYNTAX:

.LOAD <FILE = *load_file*> <RUN = PREVIOUS | CURRENT>

EXAMPLE 1:

```
.TITLE
.SAVE FILE=design.ic
.LOAD FILE=design.ic0
                $load--design.ic0 save--design.ic0
.alter
...             $load--none   save--design.ic1
.alter
...              $load--none   save--design.ic2
.end
```

This example loads a file name *design.ic0*, which you previously saved using a **.SAVE** command.

EXAMPLE 2:

```
.TITLE
.SAVE FILE=design.ic
.LOAD FILE=design.ic RUN=PREVIOUS
               $load--none   save--design.ic0
.alter
...           $load--design.ic0 save--design.ic1
.alter
...           $load--design.ic1 save--design.ic2
.end
```

EXAMPLE 3:

```
.TITLE
.SAVE FILE=design.ic

.LOAD FILE=design.ic RUN=CURRENT
             $load--design.ic0 save--design.ic0
.alter
...          $load--design.ic1 save--design.ic1
.alter
...          $load--design.ic2 save--design.ic2
.end
```

DESCRIPTION:

Use the **.LOAD** statement to input the contents of a file, that you stored using the **.SAVE** statement in HSPICE.

Files stored with the **.SAVE** statement contain operating point information, for the point in the analysis at which you executed **.SAVE**.

Do not use the **.LOAD** command for concatenated netlist files.

Command Argument	**Definition**
load_file	Name of the file, in which **.SAVE** saved an operating point, for the circuit under simulation.The format of the file name is <*design*>.ic#. Default is <*design*>.ic0, where *design* is the root name of the design.
RUN=	Used only outside of **.ALTER** statements, in a netlist that contains **.ALTER** statements. The format of file name is <*design*>.ic.
PREVIOUS	Each **.ALTER** run uses the saved operating point from the previous **.ALTER** run in the same simulation.
CURRENT	Each **.ALTER** run uses the saved operating point from the current **.ALTER** run in the last simulation.

SEE ALSO:
.SAVE

.MACRO

SYNTAX:

.MACRO *subnam n1 <n2 n3 …> <parnam = val>*
.EOM

EXAMPLE 1:

```
*FILE SUB2.SP TEST OF SUBCIRCUITS
.OPTION LIST ACCT
  V1 1 0 1
.PARAM P5 = 5 P2 = 10
.SUBCKT SUB1 1 2 P4 = 4
  R1 1 0 P4
  R2 2 0 P5
  X1 1 2 SUB2 P6 = 7
  X2 1 2 SUB2
.ENDS
*
.MACRO SUB2 1 2 P6 = 11
  R1 1 2 P6
  R2 2 0 P2
.EOM
  X1 1 2 SUB1 P4 = 6
  X2 3 4 SUB1 P6 = 15
  X3 3 4 SUB2
*
.MODEL DA D CJA = CAJA CJP = CAJP VRB = -20
IS = 7.62E-18
+   PHI = .5 EXA = .5 EXP = .33
.PARAM CAJA = 2.535E-16 CAJP = 2.53E-16
.END
```

The preceding example defines two subcircuits: SUB1 and SUB2. These are resistor divider networks, whose resistance values are parameters (variables). The X1, X2, and X3 statements call these subcircuits. Because the resistor values are different in each call, these three calls produce different subcircuits.

EXAMPLE 2:

```
.SUBCKT Inv a y Strength = 3
   Mp1 <MosPinList> pMosMod L = 1.2u W = 'Strength * 2u'
   Mn1 <MosPinList> nMosMod L = 1.2u W = 'Strength * 1u'
.ENDS
...
xInv0 a y0 Inv$ Default devices: p device = 6u,
   $ n device = 3u
xInv1 a y1 Inv Strength = 5$ p device = 10u, n
device = 5u
xInv2 a y2 Inv Strength = 1$ p device =  2u, n
device = 1u
...
```

This example implements an inverter that uses a *Strength* parameter. By default, the inverter can drive three devices. Enter a new value for the *Strength* parameter in the element line, to select larger or smaller inverters for the application.

DESCRIPTION:

You can create a subcircuit description for a commonly-used circuit, and include one or more references to the subcircuit in your netlist.

To define a subcircuit in your netlist, use the **.MACRO** statement. Use the **.EOM** statement to terminate a **.MACRO** statement.

Command Argument	**Definition**
subnam	Specifies a reference name for the subcircuit model call.
n1 …	Node numbers for external reference; cannot be the ground node (zero). Any element nodes that are in the subcircuit, but are not in this list, are strictly local, with three exceptions: • Ground node (zero). • Nodes assigned using BULK = node in MOSFET or BJT models. • Nodes assigned using the **.GLOBAL** statement.

Command Argument	Definition
parnam	A parameter name set to a value. Use only in the subcircuit. To override this value, assign it in the subcircuit call, or set a value in a **.PARAM** statement.
SubDefaultsList	<*SubParam1*>=<*Expression*> [<*SubParam2*>=<*Expression*>...]

SEE ALSO:

.ENDS
.EOM

.MALIAS

SYNTAX:

.MALIAS *model_name=alias_name1 <alias_name2 ...>*

- *model_name* is the model name defined in the .model card.
- *alias_name1*... is the alias that an instance (element) of the model references.

EXAMPLE:

```
*file: test malias statement
.OPTION acct tnom=50 list gmin=1e-14 post
.temp 0.0 25
.tran .1 2
vdd 2 0 pwl 0 -1 1 1
d1 2 1 zend dtemp=25
d2 1 0 zen dtemp=25
* malias statements
.malias zendef = zen zend
* model definition
.model zendef d (vj=.8 is=1e-16 ibv=1e-9 bv=6.0
rs=10
+ tt=0.11n n=1.0 eg=1.11 m=.5 cjo=1pf tref=50)
.end
```

- *zendef* is a diode model
- *zen* and *zend* are its aliases.
- The *zendef* model points to both the *zen* and *zend* aliases.

DESCRIPTION:

You can use the **.MALIAS** statement to assign an alias (another name) to a diode, BJT, JFET, or MOSFET model that you defined in a **.MODEL** statement.

.MALIAS differs from **.ALIAS** in two ways:

- The alias in an **.ALIAS** statement is defined in a **.MODEL** card, but the model card does not define the alias in a **.MALIAS** statement.
- The **.ALIAS** command works only if you include **.ALTER** in the netlist. You can use **.MALIAS** without **.ALTER**.

You can us **.MALIAS** to alias to a model name that you defined in a **.MODEL** statement, or to alias to a subcircuit name that you defined in a **.SUBCKT** statement. The syntax for **.MALIAS** is the same in either usage.

SEE ALSO:

.ALIAS

.MATERIAL

SYNTAX:

.MATERIAL *mname* METAL|DIELECTRIC <ER=*val*>
+ <UR=*val*> <CONDUCTIVITY=*val*>
+ <LOSSTANGENT=*val*>

DESCRIPTION:

The field solver assigns the following default values for metal:

- CONDUCTIVITY = -1 (perfect conductor)
- ER = 1
- UR = 1

PEC is a pre-defined metal name. You cannot redefine its default values.

The field solver assigns the following default values for dielectrics:

- CONDUCTIVITY = 0 (lossless dielectric)
- LOSSTANGENT = 0 (lossless dielectric)
- ER = 1
- UR = 1

AIR is a pre-defined dielectric name. You cannot redefine its default values.

Because the field solver does not currently support magnetic materials, it ignores UR values.

Command Argument	Definition
mname	Material name.
METAL|DIELECTRIC	Material type: METAL or DIELECTRIC.
ER	Dielectric constant (relative permittivity).
UR	Relative permeability.
CONDUCTIVITY	Static field conductivity of conductor or lossy dielectric (S/m).
LOSSTANGENT	Alternating field loss tangent of dielectric (tan δ).

SEE ALSO:

.LAYERSTACK

.MEASURE

DESCRIPTION:

Use the **.MEASURE** statement to modify information, and to define the results of successive HSPICE simulations. The .MEASURE statement prints user-defined electrical specifications of a circuit. Optimization uses **.MEASURE** statements extensively. The specifications include:

- propagation
- delay
- rise time
- fall time
- peak-to-peak voltage
- minimum and maximum voltage over a specified period
- other user-defined variables

You can also use **.MEASURE** with either the error function or GOAL parameter, to optimize circuit component values, and to curve-fit measured data to model parameters.

The **.MEASURE** statement can use several different formats, depending on the application. You can use it for either DC sweep, AC, or transient analysis.

SEE ALSO:

.DOUT
.GRAPH
.PLOT
.PRINT
.PROBE
.STIM

.MEASURE (Rise, Fall, and Delay Measurements)

SYNTAX:

.MEASURE <DC|AC|TRAN> *result* TRIG … TARG …
+ <GOAL = *val*> <MINVAL = *val*> <WEIGHT = *val*>

EXAMPLE 1:

```
* Example of rise/fall/delay measurement
.MEASURE TRAN tdlay TRIG V(1) VAL = 2.5 TD = 10n
+ RISE = 2 TARG V(2) VAL = 2.5 FALL = 2
```

This example measures the propagation delay between nodes 1 and 2, for a transient analysis. HSPICE measures the delay from the second rising edge of the voltage at node 1, to the second falling edge of node 2. The measurement begins when the second rising voltage at node 1 is 2.5 V, and ends when the second falling voltage at node 2 is 2.5 V. The TD = 10n parameter counts the crossings, after 10 ns has elapsed. HSPICE prints results as tdlay = <*value*>.

EXAMPLE 2:

```
.MEASURE TRAN riset TRIG I(Q1) VAL = 0.5m RISE = 3
+ TARG I(Q1) VAL = 4.5m RISE = 3

* Rise/fall/delay measure with TRIG and TARG specs
.MEASURE pwidth TRIG AT = 10n TARG V(IN) VAL = 2.5
+ CROSS = 3
```

In the last example, TRIG. AT = 10n starts measuring time at t = 10 ns, in the transient analysis. The TARG parameters end time measurement, when V(IN) = 2.5 V, on the third crossing. pwidth is the printed output variable.

If you use the **.TRAN** statement with a **.MEASURE** statement, do not use a non-zero START time in **.TRAN** statement, or the **.MEASURE** results might be incorrect.

DESCRIPTION:

Use the Rise, Fall, and Delay form of the **.MEASURE** statement to measure independent-variable (time, frequency, or any parameter or temperature) differential measurements such as rise time, fall time, slew rate, or any measurement that requires determining independent variable values. This format specifies TRIG and TARG substatements. These two statements specify the beginning and end of a voltage or current amplitude measurement.

Command Argument	**Definition**
MEASURE	Specifies measurements. You can abbreviate to MEAS.
result	Name associated with the measured value, in the HSPICE output. This example measures the independent variable, beginning at the trigger, and ending at the target: • Transient analysis measures time. • AC analysis measures frequency. • DC analysis measures the DC sweep variable. If simulation reaches the target before the trigger activates, the resulting value is negative. Do not use DC, TRAN, or AC as the *result* name.
TRIG…	Identifies the beginning of trigger specifications. TRIG *trig_var* VAL = *trig_val* <TD = *time_delay*> + <CROSS = *c*> <RISE = *r*> <FALL = *f*> TRIG_SPEC and TARG_SPEC can also use the syntax: TRIG AT = time

Command Argument	Definition
TARG …	Identifies the beginning of target specifications.You can use the LAST keyword in TARG_SPEC to indicate the last event. TARG *targ_var* VAL = *targ_val* <TD = *time_delay*> + <CROSS = *c* | LAST> <RISE = *r* | LAST> + <FALL = *f* | LAST>
<DC|AC|TRAN>	Specifies the analysis type of the measurement. If you omit this parameter, HSPICE uses the last analysis mode that you requested.
GOAL	Specifies the desired measure value in ERR calculation for optimization. To calculate the error, the simulation uses the equation: $ERRfun = (GOAL - result)/GOAL$.
MINVAL	If the absolute value of GOAL is less than MINVAL, the MINVAL replaces the GOAL value, in the denominator of the ERRfun expression. Used only in ERR calculation for optimization. Default = 1.0e-12.
WEIGHT	Multiplies the calculated error by the weight value. Used only in ERR calculation for optimization. Default = 1.0.

TRIG/TARG Parameter	Definition
TRIG	Indicates the beginning of the trigger specification.
trig_val	Value of *trig_var*, which increments the counter for crossings, rises, or falls, by one.
trig_var	Specifies the name of the output variable, that determines the logical beginning of a measurement. If HSPICE reaches the target before the trigger activates, .MEASURE reports a negative value.
TARG	Indicates the beginning of the target signal specification.

TRIG/TARG Parameter	Definition
targ_val	Specifies the value of the *targ_var*, which increments the counter for crossings, rises, or falls, by one.
targ_var	Name of the output variable, at which HSPICE determines the propagation delay with respect to the *trig_var*.
time_delay	Amount of simulation time that must elapse, before HSPICE enables the measurement. Simulation counts the number of crossings, rises, or falls, only after the *time_delay* value. Default trigger delay is zero.

.MEASURE (Average, RMS, and Peak Measurements)

SYNTAX:

.MEASURE < TRAN > *out_var func var*
+ FROM = *start* TO = *end*

EXAMPLE 1:

```
.MEAS TRAN RMSVAL RMS V(OUT) FROM = 0NS TO = 10NS
```

In this example, the **.MEASURE** statement calculates the RMS voltage of the OUT node, from 0ns to 10ns. It then labels the result RMSVAL.

EXAMPLE 2:

```
.MEAS MAXCUR MAX I(VDD) FROM = 10NS TO = 200NS
```

In this example, the **.MEASURE** statement finds the maximum current of the VDD voltage supply, between 10ns and 200ns in the simulation. The result is called MAXCUR.

EXAMPLE 3:

```
.MEAS P2P PP PAR('V(OUT)/V(IN)')
+ FROM = 0NS TO = 200NS
```

In this example, the **.MEASURE** statement uses the ratio of V(OUT) and V(IN) to find the peak-to-peak value, in the interval of 0ns to 200ns.

DESCRIPTION:

This **.MEASURE** statement reports the average, RMS, or peak value of the specified output variable.

Command Argument	Definition
varname	User-defined variable name for the measurement.
func	One of the following keywords: • AVG: Average area under *var*, divided by the period of interest. • MAX: Maximum value of *var* over the specified interval. • MIN: Minimum value of *var* over the specified interval. • PP: Peak-to-peak: reports the maximum value, minus the minimum of *var* over the specified interval. • RMS: Root mean squared: calculates the square root of the area under the var^2 curve, divided by the period of interest. • INTEG: Integral of *var* over the specified period.
out_var var	Name of the output variable, which can be either the node voltage or the branch current of the circuit. You can also use an expression, consisting of the node voltages or the branch current.
start	Starting time of the measurement period.
end	Ending time of the measurement period.

.MEASURE (FIND and WHEN)

SYNTAX:

.MEASURE <DC|TRAN| AC> *result*
+ WHEN *out_var* = *val* <TD = *val*>
+ < RISE = *r* | LAST > < FALL = *f* | LAST >
+ < CROSS = *c* | LAST >
+ <GOAL = *val*> <MINVAL = *val*> <WEIGHT = *val*>

.MEASURE <DC|TRAN|AC> *result*
+ WHEN *out_var1* = *out_var2*
+ < TD = *val* > < RISE = *r* | LAST >
+ < FALL = *f* | LAST >
+ < CROSS = *c*| LAST > <GOAL = *val*>
+ <MINVAL = *val*> <WEIGHT = *val*>

.MEASURE <DC|TRAN|AC> *result* FIND *out_var1*
+ WHEN *out_var2* = *val* < TD = *val* >
+ < RISE = *r* | LAST >
+ < FALL = *f* | LAST > < CROSS = *c*| LAST >
+ <GOAL = *val*> <MINVAL = *val*> <WEIGHT = *val*>

.MEASURE <DC|TRAN|AC> *result* FIND *out_var1*
+ WHEN *out_var2* = *out_var3* <TD = *val* >
+ < RISE = *r* | LAST > < FALL = *f* | LAST >
+ <CROSS = *c* | LAST> <GOAL = *val*>
+ <MINVAL = *val*> <WEIGHT = *val*>

.MEASURE <DC|TRAN|AC> *result* FIND *out_var1*
+ AT = *val* <GOAL = *val*> <MINVAL = *val*>
+ <WEIGHT = *val*>

EXAMPLE:

```
* MEASURE statement using FIND/WHEN
.MEAS TRAN TRT FIND PAR('V(3)-V(4)')
+ WHEN V(1)=PAR('V(2)/2') RISE = LAST
.MEAS STIME WHEN V(4) = 2.5 CROSS = 3
```

In this example, the first measurement, TRT, calculates the difference between V(3) and V(4), when V(1) is half the voltage of V(2) at the last rise event.

The second measurement, STIME, finds the time when V(4) is 2.5V at the third rise-fall event. A CROSS event is a rising or falling edge.

DESCRIPTION:

The FIND and WHEN functions of the **.MEASURE** statement specify to measure:

- Any independent variables (time, frequency, parameter).
- Any dependent variables (voltage or current, for example).
- Derivative of a dependent variable, if a specific event occurs.

Command Argument	Definition
CROSS = c RISE = r FALL = f	Numbers indicate which CROSS, FALL, or RISE event to measure. For example: .meas tran tdlay trig v(1) val=1.5 td=10n + rise=2 targ v(2) val=1.5 fall=2 In the above example, rise=2 specifies to measure the v(1) voltage, only on the first two rising edges of the waveform. The value of these first two rising edges is 1. However, trig v(1) val=1.5 indicates to trigger when the voltage on the rising edge voltage is 1.5, which never occurs on these first two rising edges. So the v(1) voltage measurement never finds a trigger. • RISE = r, the WHEN condition is met, and measurement occurs after the designated signal has risen r rise times. • FALL = f, measurement occurs when the designated signal has fallen f fall times. A crossing is either a rise or a fall, so for CROSS = c, measurement occurs when the designated signal has achieved a total of c crossing times, as a result of either rising or falling. For TARG, the LAST keyword specifies the last event.
LAST	HSPICE measures when the last CROSS, FALL, or RISE event occurs. • CROSS = LAST, measurement occurs the last time the WHEN condition is true, for a rising or falling signal. • FALL = LAST, measurement occurs the last time the WHEN condition is true, for a falling signal. • RISE = LAST, measurement occurs the last time the WHEN condition is true, for a rising signal. LAST is a reserved word; you cannot use it as a parameter name in the above .MEASURE statements.

Command Argument	Definition
AT = val	Special case for trigger specification. *val* is: • Time for TRAN analysis. • Frequency for AC analysis. • Parameter for DC analysis. The trigger determines where measurement starts.
<DC|AC|TRAN>	Analysis type for the measurement. If you omit this parameter, HSPICE assumes the last analysis type that you requested.
FIND	Selects the FIND function.
GOAL	Desired .MEASURE value. Optimization uses this value in ERR calculation. The following equation calculates the error: $ERRfun = (GOAL - result)/GOAL$.
LAST	Starts measurement at the last CROSS, FALL, or RISE event. • For CROSS = LAST, measurement starts the last time the WHEN condition is true, for either a rising or falling signal. • For FALL = LAST, measurement starts the last time the WHEN condition is true, for a falling signal. • For RISE = LAST, measurement starts the last time the WHEN condition is true for a rising signal. LAST is a reserved word. Do not use it as a parameter name in these .MEASURE statements.
MINVAL	If the absolute value of GOAL is less than MINVAL, then MINVAL replaces the GOAL value in the denominator of the ERRfun expression. Used only in ERR calculation for optimization. Default = 1.0e-12.
out_var(1,2,3)	These variables establish conditions that start a measurement.
result	Name of a measured value, in the HSPICE output.
TD	Time at which measurement starts.

Command Argument	Definition
WEIGHT	Multiplies the calculated error by the weight value. Used only in ERR calculation for optimization. Default = 1.0.
WHEN	Selects the WHEN function.

.MEASURE (Equation Evaluation/ Arithmetic Expression)

SYNTAX:

.MEASURE <DC|TRAN|AC> *result* PARAM = ’*equation*’
+ <GOAL = *val*> <MINVAL = *val*>

.MEASURE < TRAN > *varname* PARAM = “*expression*”

EXAMPLE:

```
.MEAS TRAN V3MAX MAX V(3) FROM 0NS TO 100NS
.MEAS TRAN V2MIN MIN V(2) FROM 0NS TO 100NS
.MEAS VARG PARAM = ‘(V2MIN + V3MAX)/2’
```

The first two measurements, V3MAX and V2MIN, set up the variables for the third measurement statement.

- V3MAX is the maximum voltage of V(3) between 0ns and 100ns of the simulation.
- V2MIN is the minimum voltage of V(2) during that same interval.
- VARG is the mathematical average of the V3MAX and V2MIN measurements.

DESCRIPTION:

Use the Equation Evaluation form of the **.MEASURE** statement to evaluate an equation, that is a function of the results of previous **.MEASURE** statements. The equation must not be a function of node voltages or branch currents.

The *expression* option is an arithmetic expression, that uses results from other prior **.MEASURE** statements.

Expressions used in arithmetic expression must not be a function of node voltages or branch currents. Expressions used in all other **.MEASURE** statements can contain either

node voltages or branch currents, but must not use results from other **.MEASURE** statements.

.MEASURE (Average, RMS, MIN, MAX, INTEG, and PP)

SYNTAX:

.MEASURE <DC|AC|TRAN> *result func out_var*
+ <FROM = *val*> <TO = *val*> <GOAL = *val*>
+ <MINVAL = *val*> <WEIGHT = *val*>

EXAMPLE 1:

```
.MEAS TRAN avgval AVG V(10) FROM = 10ns TO = 55ns
```

This example calculates the average nodal voltage value for node 10, during the transient sweep, from the time 10 ns to 55 ns. It prints out the result as *avgval*.

EXAMPLE 2:

```
.MEAS TRAN MAXVAL MAX V(1,2) FROM = 15ns  TO = 100ns
```

This example finds the maximum voltage difference between nodes 1 and 2, for the time period from 15 ns to 100 ns.

EXAMPLE 3:

```
.MEAS TRAN MINVAL MIN V(1,2) FROM = 15ns TO = 100ns
.MEAS TRAN P2PVAL PP I(M1)  FROM = 10ns TO = 100ns
```

DESCRIPTION:

Average (AVG), RMS, MIN, MAX, and peak-to-peak (PP) measurement modes report statistical functions of the output variable, rather than analysis values.

- AVG calculates the area under an output variable, divided by the periods of interest.
- RMS divides the square root of the area under the output variable square, by the period of interest.

- MIN reports the minimum value of the output function, over the specified interval.
- MAX reports the maximum value of the output function, over the specified interval.
- PP (peak-to-peak) reports the maximum value, minus the minimum value, over the specified interval.

AVG, RMS, and INTEG have no meaning in a DC data sweep, so if you use them, HSPICE issues a warning message.

Command Argument	**Definition**
<DC|AC|TRAN>	Specifies the analysis type for the measurement. If you omit this parameter, HSPICE assumes the last analysis mode that you requested.
FROM	Specifies the initial value for the *func* calculation. For transient analysis, this value is in units of time.
TO	Specifies the end of the *func* calculation.
GOAL	Specifies the .MEASURE value. Optimization uses this value for ERR calculation. This equation calculates the error: $ERRfun = (GOAL - result)/GOAL$

Command Argument	Definition
func	Indicates one of the measure statement types: • AVG (average): Calculates the area under the out_var, divided by the periods of interest. • MAX (maximum): Reports the maximum value of the out_var, over the specified interval. • MIN (minimum): Reports the minimum value of the out_var, over the specified interval. • PP (peak-to-peak): Reports the maximum value, minus the minimum value, of the out_var, over the specified interval. • RMS (root mean squared): Calculates the square root of the area under the out_var2 curve, divided by the period of interest.
result	Name of the measured value, in the output. The value is a function of the variable (*out_var*) and *func*.
out_var	Name of any output variable whose function (*func*) the simulation measures.
WEIGHT	Multiplies the calculated error, by the weight value. Used only in ERR calculation for optimization. Default = 1.0.

.MEASURE (Integral Function)

SYNTAX:

.MEASURE <DC|AC|TRAN> *result* INTEGRAL *out_var*
+ <FROM = *val*> <TO = *val*> <GOAL = *val*>
+ <MINVAL = *val*> <WEIGHT = *val*>

EXAMPLE:

```
.MEAS TRAN charge INTEG I(cload) FROM = 10ns
+ TO = 100ns
```

This example calculates the integral of *I(cload)*, from 10 ns to 100 ns.

DESCRIPTION:

The INTEGRAL function reports the integral of an output variable, over a specified period.

The INTEGRAL function (with func), uses the same syntax as the average (AVG), RMS, MIN, MAX, and peak-to-peak (PP) measurement mode, to defined the INTEGRAL (INTEG).

.MEASURE (Derivative Function)

SYNTAX:

.MEASURE <DC|AC|TRAN> *result* DERIVATIVE *out_var*
+ AT = *val* <GOAL = *val*> <MINVAL = *val*>
+ <WEIGHT = *val*>

.MEASURE <DC|AC|TRAN> *result* DERIVATIVE *out_var*
+ WHEN *var2* = *val* <RISE = *r* | LAST>
+ <FALL = *f* | LAST> <CROSS = *c* | LAST> <TD = *tdval*>
+ <GOAL = *goalval*> <MINVAL = *minval*>
+ <WEIGHT = *weightval*>

.MEASURE <DC|AC|TRAN> *result* DERIVATIVE *out_var*
+ WHEN *var2* = *var3* <RISE = *r* | LAST>
+ <FALL = *f* | LAST> <CROSS = *c* | LAST> <TD = *tdval*>
+ <GOAL = *goalval*> <MINVAL = *minval*>
+ <WEIGHT = *weightval*>

EXAMPLE 1:

```
.MEAS TRAN slew rate DERIV V(out) AT = 25ns
```

This example calculates the derivative of V(out), at 25 ns.

EXAMPLE 2:

```
.MEAS TRAN slew DERIV v(1) WHEN v(1) = ’0.90*vdd’
```

This example calculates the derivative of v(1), when v(1) is equal to 0.9*vdd.

EXAMPLE 3:

```
.MEAS AC delay DERIV ’VP(output)/360.0’ AT = 10khz
```

This example calculates the derivative of VP(output)/360.0, when the frequency is 10 kHz.

DESCRIPTION:

The DERIVATIVE function provides the derivative of:

- An output variable, at a specified time or frequency.
- Any sweep variable, depending on the type of analysis.
- A specified output variable, when some specific event occurs.

Command Argument	**Definition**
AT = val	Value of *out_var*, at which the derivative is found.
CROSS = c RISE = r FALL = f	The numbers indicate which occurrence of a CROSS, FALL, or RISE event starts a measurement. For RISE = r, when the designated signal has risen r rise times, the WHEN condition is met, and measurement starts. For FALL = f, measurement starts when the designated signal has fallen f fall times. A crossing is either a rise or a fall, so for CROSS = c, measurement starts when the designated signal has achieved a total of c crossing times, as a result of either rising or falling.
<DC|AC|TRAN>	Specifies the analysis type to measure. If you omit this parameter, HSPICE or HSPICE XT/RF assumes the last analysis mode that you requested.
DERIVATIVE	Selects the derivative function. You can abbreviate to DERIV.
GOAL	Specifies the desired .MEASURE value. Optimization uses this value for ERR calculation. This equation calculates the error: $ERRfun = (GOAL - result)/GOAL$

Command Argument	Definition
LAST	Measures when the last CROSS, FALL, or RISE event occurs. CROSS = LAST, measures the last time the WHEN condition is true, for a rising or falling signal. FALL = LAST, measures the last time WHEN is true, for a falling signal. RISE = LAST, measures the last time WHEN is true, for a rising signal. LAST is a reserved word; do not use it as a parameter name in the above .MEASURE statements.
MINVAL	If the absolute value of GOAL is less than MINVAL, MINVAL replaces the GOAL value in the denominator of the ERRfun expression. Used only in ERR calculation for optimization. Default = 1.0e-12.
out_var	Variable for which HSPICE finds the derivative.
result	Name of the measured value, in the output.
TD	Identifies the time when measurement starts.
var(2,3)	These variables establish conditions that start a measurement.
WEIGHT	Multiplies the calculated error, between result and GOAL, by the weight value. Used only in ERR calculation for optimization. Default = 1.0.
WHEN	Selects the WHEN function.

.MEASURE (Error Function)

SYNTAX:

.MEASURE <DC|AC|TRAN> *result*
+ ERRfun *meas_var calc_var*
+ <MINVAL = *val*> < IGNORE | YMIN = *val*>
+ <YMAX = *val*> <WEIGHT = *val*> <FROM = *val*>
+ <TO = *val*>

DESCRIPTION:

The relative error function reports the relative difference between two output variables. You can use this format in optimization and curve-fitting of measured data. The relative error format specifies the variable to measure and calculate, from the **.PARAM** variable. To calculate the relative error between the two, HSPICE uses the ERR, ERR1, ERR2, or ERR3 function. With this format, you can specify a group of parameters to vary, to match the calculated value and the measured data.

Command Argument	Definition
<DC|AC|TRAN>	Specifies the analysis type, for the measurement. If you omit this parameter, HSPICE assumes the last analysis mode that you requested.
result	Name of the measured result, in the output.
ERRfun	ERRfun indicates which error function to use: ERR, ERR1, ERR2, or ERR3.
meas_var	Name of any output variable or parameter, in the data statement. *M* denotes the *meas_var*, in the error equation.

Command Argument	Definition
calc_var	Name of the simulated output variable or parameter, in the .MEASURE statement, to compare with *meas_var*. *C* is the *calc_var* in the error equation.
IGNOR|YMIN	If the absolute value of meas_var is less than the IGNOR value, then the ERRfun calculation does not consider this point. Default = 1.0e-15.
FROM	Specifies the beginning of the ERRfun calculation. For transient analysis, the *from* value is in units of time. Defaults to the first value of the sweep variable.
WEIGHT	Multiplies the calculated error, by the weight value. Used only in ERR calculation for optimization. Default = 1.0.
YMAX	If the absolute value of meas_var is greater than the YMAX value, then the ERRfun calculation does not consider this point. Default = 1.0e+15.
TO	End of the ERRfun calculation. Default is last value of the sweep variable.
MINVAL	If the absolute value of meas_var is less than MINVAL, MINVAL replaces the meas_var value in the denominator of the ERRfun expression. Used only in ERR calculation for optimization. Default = 1.0e-12.

.MODEL

SYNTAX:

.MODEL *mname type* <VERSION = *version_number*>
+ <*pname1 = val1 pname2 = val2* ...>

.MODEL *mname* OPT <*parameter=val* ...>

The following is the **.MODEL** syntax for use with **.GRAPH**:

.MODEL *mname* PLOT (*pnam1 = val1 pnam2 = val2*….)

EXAMPLE:

```
.MODEL MOD1 NPN BF=50 IS=1E-13 VBF=50 AREA=2 PJ=3,
+ N=1.05
```

DESCRIPTION:

Use the **.MODEL** command to include an instance (element) of a pre-defined HSPICE model in your input netlist.

For each optimization within a data file, specify a **.MODEL** statement. HSPICE can then execute more than one optimization per simulation run. The *.MODEL* optimization statement defines:

- Convergence criteria.
- Number of iterations.
- Derivative methods.

Command Argument	Definition
mname	Model name reference. Elements must use this name to refer to the model. If model names contain periods (.), the automatic model selector might fail. When used with **.GRAPH**, this is the plot model name, referenced in **.GRAPH** statements.
type	Selects a model type. Must be one of the following. AMP operational amplifier model C capacitor model CORE magnetic core model D diode model L inductor model or magnetic core mutual inductor model NJF n-channel JFET model NMOS n-channel MOSFET model NPN npn BJT model OPT optimization model PJF p-channel JFET model PLOT plot model for the .GRAPH statement PMOS p-channel MOSFET model PNP pnp BJT model R resistor model U lossy transmission line model (lumped) W lossy transmission line model SP S parameter

Command Argument	Definition
pname1 ...	Parameter name. Assign a model parameter name (*pname1*) from the parameter names for the appropriate model type. Each model section provides default values. For legibility, enclose the parameter assignment list in parentheses, and use either blanks or commas to separate each assignment. Use a plus sign (+) to start a continuation line. When used with **.GRAPH**, each **.GRAPH** statement model includes several model parameters. If you do not specify model parameters, HSPICE uses the default values of the model parameters, described in the following table. *Pnamn* is one of the model parameters of a **.GRAPH** statement, and *valn* is the value of pnamn. Valn can be more than one parameter.
VERSION	HSPICE version number. Allows portability of the BSIM (LEVEL=13) and BSIM2 (LEVEL = 39) models, between HSPICE releases. HSPICE release numbers, and the corresponding version numbers, are: HSPICE *release* — *Version number* 9007B — 9007.02 9007D — 9007.04 92A — 92.01 92B — 92.02 93A — 93.01 93A.02 — 93.02 95.3 — 95.3 96.1 — 96.1 The VERSION parameter is valid only for LEVEL 13 and LEVEL 39 models. Use it with HSPICE Release H93A.02 and higher. If you use the parameter with any other model, or with a release before H93A.02, HSPICE issues a warning, but the simulation continues.You can also use VERSION to denote the BSIM3v3 version number only, in model LEVELs 49 and 53. For LEVELs 49 and 53, the HSPVER parameter denotes the HSPICE release number.
PLOT	Keyword for a **.GRAPH** statement model.

You can specify the following OPT parameters in the **.MODEL** statement:

Parameter	Description
mname	Model name. Elements use this name to refer to the model.
CENDIF	Selects different derivative methods. Default=1.0e-9. The following calculates the gradient of the RESULTS functions: ||Transpose(Jacobi(F(X))) * F(X)||, where F(X) is the RESULT function If the resulting gradient is less than CENDIF, HSPICE uses more accurate but more time-consuming derivative methods. By default, HSPICE uses faster but less-accurate derivative methods. To use the more-accurate methods, set CENDIF to a larger value than GRAD. If the gradient of the RESULTS function is less than GRAD, optimization finishes before CENDIF takes effect. • If the value is too large, the optimizer requires more CPU time. • If the value is too small, the optimizer might not find as accurate an answer.
CLOSE	Initial estimate of how close parameter initial value estimates are, to the solution. CLOSE multiplies changes in new parameter estimates. If you use a large CLOSE value, the optimizer takes large steps toward the solution. For a small value, the optimizer takes smaller steps toward the solution. You can use a smaller value for close parameter estimates, and a larger value for rough initial guesses. Default=1.0. • If CLOSE is greater than 100, the steepest descent in the Levenburg-Marquardt algorithm dominates. • If CLOSE is less than 1, the Gauss-Newton method dominates. For more details, see L. Spruiell, “Optimization Error Surfaces,” *Meta-Software Journal,* Volume 1, Number 4, December 1994.

Parameter	Description
CUT	Modifies CLOSE, depending on how successful iterations are, toward the solution. If the last iteration succeeds, descent toward the CLOSE solution decreases by the CUT value. That is, CLOSE = CLOSE / CUT If the last iteration was not a successful descent to the solution, CLOSE increases by CUT squared. That is, CLOSE = CLOSE * CUT * CUT CUT drives CLOSE up or down, depending on the relative success in finding the solution. The CUT value must be > 1. Default = 2.0.
DIFSIZ	Increment change in a parameter value, for gradient calculations ($\Delta x = DIFSIZ \cdot max(x, 0.1))$. If you specify delta in a .PARAM statement, then Δx = delta. Default = 1e-3.
GRAD	Represents possible convergence, if the gradient of the RESULTS function is less than GRAD. Most applications use values of 1e-6 to 1e-5. Too large a value can stop the optimizer before finding the best solution. Too small a value requires more iterations. Default=1.0e-6.
ITROPT	Maximum number of iterations. Typically, you need no more than 20-40 iterations, to find a solution. Too many iterations can imply that the RELIN, GRAD, or RELOUT values are too small. Default=20.
LEVEL	Selects an optimizing algorithm. Currently, the only option is LEVEL=1, a modified Levenburg-Marquardt algorithm.
MAX	Sets the upper limit on CLOSE. Use values > 100. Default=6.0e+5.
PARMIN	Allows better control of incremental parameter changes, during error calculations. Default=0.1. This produces more control over the trade-off between simulation time and optimization result accuracy. To calculate parameter increments, HSPICE uses the relationship: Dpar_val = DIFSIZ · MAX(par_val, PARMIN)

Parameter	Description
RELIN	Sets the relative input parameter (delta_par_val / MAX(par_val,1e-6)), for convergence. If all optimizing input parameters vary by no more than RELIN between iterations, the solution converges. RELIN is a relative variance test, so a value of 0.001 implies that optimizing parameters vary by less than 0.1%, from one iteration to the next. Default=0.001.
RELOUT	Sets the relative tolerance to finish optimization. For RELOUT=0.001, if the relative difference in the RESULTS functions, from one iteration to the next, is less than 0.001, then optimization is finished. Default=0.001.

.NET

SYNTAX:

One-Port Network

.NET *input* <RIN = *val*>

.NET *input* <*val*>

Two-Port Network

.NET *output input* <ROUT = *val*> <RIN = *val*>

EXAMPLE:

One-Port Network

```
.NET              VINAC        RIN = 50
.NET              IIN          RIN = 50
```

Two-Port Network

```
.NET V(10,30)     VINAC        ROUT = 75RIN = 50
.NET I(RX)        VINAC        ROUT = 75RIN = 50
```

DESCRIPTION:

You can use the **.NET** statement to compute parameters for:

- Z impedance matrix.
- Y admittance matrix.
- H hybrid matrix
- S scattering matrix.

HSPICE also computes:

- Input impedance.
- Output impedance.
- Admittance.

This analysis is part of AC small-signal analysis. To run network analysis, specify the frequency sweep for the AC statement.

Command Argument	Definition
input	Name of the voltage or current source for AC input.
output	Output port. It can be: • An output voltage, V(n1,n2). • An output current, I(source), or I(element).
RIN	Keyword, for input or source resistance. RIN calculates output impedance, output admittance, and scattering parameters. The default RIN value is 1 ohm.
ROUT	Keyword, for output or load resistance. ROUT calculates input impedance, admittance, and scattering parameters. Default=1 ohm.

SEE ALSO:
.AC

.NODESET

SYNTAX:

.NODESET V(*node1*) = *val1* <V(*node2*) = *val2* ...>

or

.NODESET *node1 val1 <node2 val2>*

EXAMPLE:

```
.NODESET V(5:SETX) = 3.5V V(X1.X2.VINT) = 1V
.NODESET V(12) = 4.5 V(4) = 2.23
.NODESET 12 4.5 4 2.23 1 1
```

DESCRIPTION:

.NODESET initializes all specified nodal voltages, for DC operating point analysis. Use the **.NODESET** statement, to correct convergence problems in DC analysis. If you set the node values in the circuit, close to the actual DC operating point solution, you enhance convergence of the simulation. The HSPICE simulator uses the NODESET voltages, only in the first iteration.

Command Argument	Definition
node1 ...	Node numbers or names can include full paths or circuit numbers.
val1	Specifies voltages.

.NOISE

SYNTAX:

.NOISE *ovv srcnam inter*

EXAMPLE:

```
.NOISE V(5) VIN 10
```

DESCRIPTION:

Use the **.NOISE** and **.AC** statements to control the noise analysis of the circuit.

Command Argument	Definition
ovv	Nodal voltage output variable. Defines the node at which HSPICE sums the noise.
srcnam	Name of the independent voltage or current source, to use as the noise input reference
inter	Interval at which HSPICE prints a noise analysis summary. *inter* specifies how many frequency points to summarize in the AC sweep. If you omit *inter*, or set it to zero, HSPICE does not print a summary. If inter is equal to or greater than one, HSPICE prints summary for the first frequency, and once for each subsequent increment of the inter frequency. The noise report is sorted according to the contribution of each node to the overall noise level.

SEE ALSO:

.AC

.OP

SYNTAX:

.OP <*format*> <*time*> <*format*> <*time*>... <*interpolation*>

EXAMPLE 1:

```
.OP .5NS CUR 10NS VOL 17.5NS 20NS 25NS
```

This example calculates:

- Operating point voltages and currents, for the DC solution.
- Currents at 10 ns, for the transient analysis.
- Voltages at 17.5 ns, 20 ns and 25 ns, for the transient analysis.

EXAMPLE 2:

```
.OP
```

This example calculates the complete DC operating point solution. The next section shows a printout of the solution.

DESCRIPTION:

When you include an **.OP** statement in an input file, HSPICE calculates the DC operating point of the circuit. You can also use the **.OP** statement to produce an operating point, during a transient analysis. You can include only one .OP statement in a simulation.

If an analysis requires calculating an operating point, you do not need to specify the **.OP** statement; HSPICE calculates an operating point. If you use a **.OP** statement, and if you include the UIC keyword in a **.TRAN** analysis statement, then simulation omits the time = 0 operating point analysis, and issues a warning in the output listing.

Command Argument	Definition
format	Any of the following keywords. Only the first letter is required. Default = ALL • ALL: Full operating point, including voltage, currents, conductances, and capacitances. This parameter outputs voltage/current for the specified time. • BRIEF: Produces a one-line summary of each element's voltage, current, and power. Current is stated in milliamperes, and power is in milliwatts. • CURRENT: Voltage table, with a brief summary of element currents and power. • DEBUG: Usually invoked only if a simulation does not converge. Debug prints back the non-convergent nodes, with the new voltage, old voltage, and the tolerance (degree of non-convergence). It also prints back the non-convergent elements, with their tolerance values. • NONE: Inhibits node and element print-outs, but performs additional analysis that you specify. • VOLTAGE: Voltage table only. The preceding keywords are mutually- exclusive; use only one at a time.
time	Place this parameter directly after ALL, VOLTAGE, CURRENT, or DEBUG. It specifies the time at which HSPICE prints the report.
interpolation	Selects an interpolation method for **.OP** time points during transient analysis, or no interpolation. Only the first character is required; that is, typing *i* has the same effect as typing *interpolation*. Default is not active. If you specify *interpolation*, all of the time points in the **.OP** statement (except time=0) use the interpolation method to calculate the OP value during the transient analysis. If you use this keyword, it must be at the end of the **.OP** statement. HSPICE ignores any word after this keyword.

.OPTION

For detailed descriptions of all simulation options that you can set using **.OPTION** commands, see Chapter 3, “Options in HSPICE Netlists”

.PARAM

SYNTAX:

Simple parameter assignment:

.PARAM <*ParamName*>=<*RealNumber*>

Algebraic parameter assignments:

.PARAM <*ParamName*>='<*AlgebraicExpression*>'

.PARAM <*ParamName1*>=<*ParamName2*>

User-defined functions:

.PARAM <*ParamName*>(<*pv1*>[<*pv2*>])='<*Expression*>'

Pre-defined analysis functions:

.PARAM <*FunctionName*> = <*Value*>

Optimized parameter assignment:

.PARAM *parameter*=OPT*xxx* (*initial_guess*, *low_limit*, + *upper_limit*)

.PARAM *parameter*=OPT*xxx* (*initial_guess*, *low_limit*, + *upper_limit*, *delta*)

EXAMPLE 1:

```
* Simple parameter assignment
.PARAM power_cylces=256
```

EXAMPLE 2:

```
* Numerical parameter assignment
.PARAM TermValue = 1g
   rTerm Bit0 0 TermValue
   rTerm Bit1 0 TermValue
...
```

EXAMPLE 3:

```
* Parameter assignment using expressions
.PARAM Pi         = ’355/113’
.PARAM Pi2        = ’2*Pi’
.PARAM npRatio    = 2.1
.PARAM nWidth     = 3u
.PARAM pWidth     = ’nWidth * npRatio’
Mp1  ... <pModelName> W = pWidth
Mn1  ... <nModelName> W = nWidth
...
```

EXAMPLE 4:

```
* Algebraic parameter
.param x=cos(2)+sin(2)
```

EXAMPLE 5:

```
* Algebraic expression as an output variable
.PRINT DC v(3) gain=PAR(‘v(3)/v(2)’)
+ PAR(‘V(4)/V(2)’)
```

EXAMPLE 6:

```
* My own user-defined functions
.PARAM <MyFunc( x, y )> = ‘Sqrt((x*x)+(y*y))’

.PARAM CentToFar (c)  = ’(((c*9)/5)+32)’
.PARAM F(p1,p2) = ’Log(Cos(p1)*Sin(p2))’
.PARAM SqrdProd (a,b) = ’(a*a)*(b*b)’
```

EXAMPLE 7:

```
* Pre-defined analysis function
.PARAM mcVar = Agauss(1.0,0.1)
```

EXAMPLE 8:

```
.PARAM vtx=OPT1(.7,.3,1.0) uox=OPT1(650,400,900)
```

In this example, *uox* and *vtx* are the variable model parameters, which optimize a model for a selected set of electrical specifications.

The estimated initial value for the *vtx* parameter is 0.7 volts. You can vary this value within the limits of 0.3 and 1.0 volts, for the optimization procedure. The optimization parameter reference name (OPT1) references the associated optimization analysis statement (not shown).

DESCRIPTION:

The **.PARAM** statement defines parameters. Parameters in HSPICE are names that have associated numeric values.

A parameter definition in HSPICE always uses the last value found in the input netlist (subject to local versus global parameter rules).

Use any of the following methods to define parameters:

- A simple parameter assignment is a constant real number. The parameter keeps this value, unless a later definition changes its value, or an algebraic expression assigns a new value during simulation. HSPICE does not warn you if it reassigns a parameter.

- An algebraic parameter (equation) is an algebraic expression of real values, a predefined or user-defined function, or circuit or model values. Enclose a complex expression in single quotes to invoke the algebraic processor, *unless* the expression begins with an alphabetic character and contains no spaces. A simple expression consists of a single parameter name. To use an algebraic expression as an output variable in a **.PRINT**, **.PLOT**, or **.PROBE** statement, use the PAR keyword.
- A user-defined function assignment is similar to an algebraic parameter. HSPICE extends the algebraic parameter definition to include function parameters, used in the algebraic that defines the function. You can nest user-defined functions up to three deep.
- A pre-defined analysis function. HSPICE provides several specialized analysis types, which require a way to control the analysis:
  - Temperature functions (*fn*)
  - Optimization guess/range
  - frequency
  - time
  - Monte Carlo functions

Command Argument	Definition
OPTxxx	Optimization parameter reference name. The associated optimization analysis references this name. Must agree with the OPTxxx name in the analysis command associated with an OPTIMIZE keyname.
parameter	Parameter to vary. • Initial value estimate • Lower limit. • Upper limit. If the optimizer does not find the best solution within these constraints, it attempts to find the best solution without constraints.
delta	The final parameter value is the initial guess ± (*n*·delta). If you do not specify *delta*, the final parameter value is between *low_limit* and *upper_limit*. For example, you can use this parameter to optimize transistor drawn widths and lengths, which must be quantized.

.PKG

SYNTAX:

.PKG *pkgname*
+file= ’*pkgfilename*’
+model= ’*pkgmodelname*’

EXAMPLE 1:

```
.pkg p_test
+ file=’processor_clk_ff.ibs’
+ model=’FCPGA_FF_PKG’
```

EXAMPLE 2:

The following example shows how pin1 is referenced:

```
p_test_pin1_dia and p_test_pin1
```

The element name becomes:

```
w_p_test_pin1_?? or r_p_test_pin1_?? ...
```

DESCRIPTION:

The **.PKG** command provides the IBIS(V 3.2) Package Model feature. It supports both sections and matrixes.

Command Argument	Definition
pkgname	package card name
pkgfilename	name of a .pkg or .ibs file that contains package models.
pkgmodelname	working model in the .pkg file

The **.PKG** command automatically creates a series of elements (W or rlc). The following nodes are referenced in the netlist:

- Nodes on the die side:

  ’*pkgname*’_’*pinname*’_dia

- Nodes on the pin side:

  ’*pkgname*’_’*pinname*’

See Example 2 for how pin1 is referenced.

SEE ALSO:
.EBD
.IBIS

.PLOT

SYNTAX:

.PLOT *antype ov1 <(plo1,phi1)> <ov2> <(plo2,phi2)> ...>*

EXAMPLE 1:

```
.PLOT DC V(4) V(5) V(1) PAR(`I1(Q1)/I2(Q1)')
.PLOT TRAN V(17,5) (2,5) I(VIN) V(17) (1,9)
.PLOT AC VM(5) VM(31,24) VDB(5) VP(5) INOISE
```

- In the first line, PAR plots the ratio of the collector current and the base current, for the Q1 transistor.
- In the second line, the VDB output variable plots the AC analysis results (in decibels), for node 5.
- In the third line, the AC plot can include NOISE results and other variables that you specify.

EXAMPLE 2:

```
.PLOT AC ZIN YOUT(P) S11(DB) S12(M) Z11(R)
.PLOT DISTO HD2 HD3(R) SIM2
.PLOT TRAN V(5,3) V(4) (0,5) V(7) (0,10)
.PLOT DC V(1) V(2) (0,0) V(3) V(4) (0,5)
```

In the last line above, HSPICE sets the plot limits for V(1) and V(2), but you specify 0 and 5 volts as the plot limits for V(3) and V(4).

DESCRIPTION:

The **.PLOT** statement plots the output values of one or more variables, in a selected HSPICE analysis. Each **.PLOT** statement defines the contents of one plot, which can contain more than one output variable.

If more than one output variable appears on the same plot, HSPICE prints *and* plots the first variable specified. To print out more than one variable, include another **.PLOT** statement.

You can include wildcards in **.PLOT** statements.

Command Argument	Definition
antype	Type of analysis for the specified plots. Analysis types are: DC, AC, TRAN, NOISE, or DISTO.
ov1 …	Output variables to plot: voltage, current, or element template, from a DC, AC, TRAN, NOISE, or DISTO analysis. See the next sections for syntax.
plo1, phi1 …	Lower and upper plot limits. The plot for each output variable uses the first set of plot limits, after the output variable name. Set a new plot limit for each output variable, after the first plot limit. For example, to plot all output variables that use the same scale, specify one set of plot limits at the end of the **.PLOT** statement. If you set the plot limits to (0,0) HSPICE automatically sets the plot limits.

SEE ALSO:

.DOUT
.GRAPH
.MEASURE
.PRINT
.PROBE
.STIM

.PRINT

SYNTAX:

.PRINT *antype ov1 <ov2 … >*

EXAMPLE 1:

```
* CASE 1
.print v(din) i(mxn18)
.dc vdin 0 5.0 0.05
.tran 1ns 60ns

* CASE 2
.dc vdin 0 5.0 0.05
.tran 1ns 60ns
.print v(din) i(mxn18)

* CASE 3
.dc vdin 0 5.0 0.05
.print v(din) i(mxn18)
.tran 1ns 60ns
```

- If you replace the **.PRINT** statement with:

  ```
  .print TRAN v(din) i(mnx)
  ```

  then all three cases have identical .sw0 and .tr0 files.

- If you replace the .print statement with:

  ```
  .print DC v(din) i(mnx)
  ```

  then the .sw0 and .tr0 files are different.

EXAMPLE 2:

```
.PRINT TRAN V (4) I(VIN) PAR(`V(OUT)/V(IN)')
```

This example prints the results of a transient analysis, for the nodal voltage named 4. It also prints the current, through the voltage source named VIN. It also prints the ratio of the nodal voltage at the OUT and IN nodes.

EXAMPLE 3:

```
.PRINT AC VM(4,2) VR(7) VP(8,3) II(R1)
```

- Depending on the value of the ACOUT option, VM(4,2) prints the AC magnitude of the voltage difference, or the difference of the voltage magnitudes, between nodes 4 and 2.
- VR(7) prints the real part of the AC voltage, between node 7 and ground.
- Depending on the ACOUT value, VP(8,3) prints the phase of the voltage difference between nodes 8 and 3, or the difference of the phase of voltage at node 8 and voltage at node 3.
- II(R1) prints the imaginary part of the current, through R1.

EXAMPLE 4:

```
.PRINT AC ZIN YOUT(P) S11(DB) S12(M) Z11(R)
```

This example prints:

- The magnitude of the input impedance.
- The phase of the output admittance.
- Several S and Z parameters.

This statement accompanies a network analysis, using the **.AC** and **.NET** analysis statements.

EXAMPLE 5:

```
.PRINT DC V(2) I(VSRC) V(23,17) I1(R1) I1(M1)
```

This example prints the DC analysis results for several different nodal voltages and currents, through:

- The resistor named R1.
- The voltage source named VSRC.
- The drain-to-source current of the MOSFET named M1.

EXAMPLE 6:

```
.PRINT NOISE INOISE
```

This example prints the equivalent input noise.

EXAMPLE 7:

```
.PRINT DISTO HD3 SIM2(DB)
```

This example prints the magnitude of third-order harmonic distortion, and the decibel value of the intermodulation distortion sum, through the load resistor that you specify in the **.DISTO** statement.

EXAMPLE 8:

```
.PRINT AC INOISE ONOISE VM(OUT) HD3
```

This statement includes NOISE, DISTO, and AC output variables in the same **.PRINT** statement in HSPICE.

EXAMPLE 9:

```
.PRINT pj1 = par(‘p(rd) +p(rs)‘)
```

This statement prints the value of pj1, with the specified function.

HSPICE ignores **.PRINT** statement references to nonexistent netlist part names, and prints those names in a warning.

EXAMPLE 10:

Derivative function:

```
.PRINT der=deriv('v(NodeX)')
```

Integrate function:

```
.PRINT int = integ('v(NodeX)')
```

The parameter can be a node voltage, or a reasonable expression.

EXAMPLE 11:

```
.print p1 = 3
.print p2 = par("p1*5")
```

You can use p1 and p2 as parameters in netlist. The p1 value is 3; the p2 value is 15.

DESCRIPTION:

The **.PRINT** statement specifies output variables, for which HSPICE prints values. You can include wildcards in **.PRINT** statements.

You can also use the *iall* keyword in a **.PRINT** statement, to print all branch currents of all diode, BJT, JFET, or MOSFET elements in your circuit design.

Command Argument	Definition
antype	Type of analysis for outputs. Antype is one of the following types: DC, AC, TRAN, NOISE, or DISTO.
ov1 …	Output variables to print. These are voltage, current, or element template variables, from a DC, AC, TRAN, NOISE, or DISTO analysis.

SEE ALSO:

.DOUT
.GRAPH
.MEASURE
.PLOT
.PROBE
.STIM

.PROBE

SYNTAX:

.PROBE *antype ov1 <ov2 ...>*

EXAMPLE 1:

```
.PROBE DC V(4) V(5) V(1) beta = PAR(`I1(Q1)/I2(Q1)')
```

EXAMPLE 2:

```
* Derivative function
.PROBE der=deriv('v(NodeX)')

* Integrate function
.PROBE int = integ('v(NodeX)')
```

DESCRIPTION:

The **.PROBE** statement saves output variables into interface and graph data files.The parameter can be a node voltage, or a reasonable expression. You can include wildcards in .PROBE statements.

Command Argument	Definition
antype	Type of analysis for the specified plots. Analysis types are: DC, AC, TRAN, NOISE, or DISTO.
ov1 …	Output variables to plot: voltage, current, or element template variables from a DC, AC, TRAN, NOISE, or DISTO analysis. **.PROBE** can include more than one output variable.

SEE ALSO:

.DOUT
.GRAPH
.MEASURE
.PLOT
.PRINT
.STIM

.PROTECT

SYNTAX:

.PROTECT

DESCRIPTION:

The **.PROTECT** statement keeps models and cell libraries private.

- The **.PROTECT** statement suppresses printing text from the list file, such as when you use the BRIEF option.
- The **.UNPROTECT** command restores normal output functions.
- Any elements and models located between a **.PROTECT** and an **.UNPROTECT** statement, inhibit the element and model listing from the LIST option.
- The **.OPTION NODE** nodal cross reference, and the .OP operating point printout, do not list any nodes that are contained within the **.PROTECT** and **.UNPROTECT** statements.

SEE ALSO:

.UNPROTECT

.PZ

SYNTAX:

.PZ *output input*

.PZ *ov srcname*

EXAMPLE:

```
.PZ   V(10)   VIN
.PZ   I(RL)   ISORC
```

- In the first pole/zero analysis, the output is the voltage for node 10, and the input is the VIN independent voltage source.
- In the second pole/zero analysis, the output is the branch current for the RL branch, and the input is the ISORC independent current source.

DESCRIPTION:

The **.PZ** command performs pole/zero analysis (you do not need to specify **.OP**). See “Pole/Zero Analysis” in the *HSPICE Applications Manual*, for complete information about pole/zero analysis.

Command Argument	Definition
input	Input source. Can be the name of any independent voltage or current source.
output	Output variables, which can be: • Any node voltage, V(*n*). • Any branch current, I(*branch_name*).
ov	Output variable: a node voltage V(n), or branch current I(element)
srcnam	Input source: an independent voltage or current source name

SEE ALSO:

.DC

.SAMPLE

SYNTAX:

.SAMPLE FS = *freq* <TOL = *val*> <NUMF = *val*>
+ <MAXFLD = *val*> <BETA = *val*>

DESCRIPTION:

To acquire data from analog signals, use the **.SAMPLE** statement, with the **.NOISE** and **.AC** statements, to analyze data sampling noise in HSPICE. The SAMPLE analysis performs a noise-folding analysis, at the output node.

Command Argument	Definition
FS = freq	Sample frequency, in Hertz.
TOL	Sampling-error tolerance: the ratio of the noise power (in the highest folding interval) to the noise power (in baseband). Default = `1.0e-3`.
NUMF	Maximum number of frequencies that you can specify. The algorithm requires about ten times this number of internally-generated frequencies, so keep this value small. Default = `100`.
MAXFLD	Maximum number of folding intervals (default = `10.0`). The highest frequency (in Hertz) that you can specify is: FMAX = MAXFLD · FS
BETA	Optional noise integrator (duty cycle), at the sampling node: BETA = 0 no integrator BETA = 1 simple integrator (default) If you clock the integrator (integrates during a fraction of the 1/FS sampling interval), then set BETA to the duty cycle of the integrator.

.SAVE

SYNTAX:

.SAVE <TYPE = *type_keyword*> <FILE = *save_file*>
+ <LEVEL = *level_keyword*> <TIME = *save_time*>

EXAMPLE:

```
.TEMP -25 0 25
.SAVE TYPE=NODESET FILE=my_design.ic0 LEVEL=ALL
+ TIME=0
```

This example saves the operating point corresponding to **.TEMP** -25, to a file named *my_design.ic0*.

DESCRIPTION:

The **.SAVE** statement in HSPICE stores the operating point of a circuit, in a file that you specify. For quick DC convergence in subsequent simulations, use the **.LOAD** statement to input the contents of this file. HSPICE saves the operating point by default, even if the HSPICE input file does not contain a **.SAVE** statement. To not save the operating point, specify **.SAVE** LEVEL = NONE.

You can save the operating point data as either an **.IC** or a **.NODESET** statement.

A parameter or temperature sweep saves only the first operating point.

Command Argument	Definition
type_keyword	Storage method, for saving the operating point. The type can be one of the following. Default is NODESET. • **.NODESET**: Stores the operating point as a .NODESET statement. Later simulations initialize all node voltages to these values, if you use the .LOAD statement. If circuit conditions change incrementally, DC converges within a few iterations. • **.IC**: Stores the operating point as a .IC statement. Later simulations initialize node voltages to these values if the netlist includes the **.LOAD** statements.
save_file	Name of the file that stores DC operating point data. The file name format is <*design*>.ic#. Default is <*design*>.ic0.
level_keyword	Circuit level, at which you save the operating point. The level can be one of the following. • ALL (default): Saves all nodes, from the top to the lowest circuit level. This option offers the greatest improvement in simulation time. • TOP: Saves only nodes in the top-level design. Does not save subcircuit nodes. • NONE: Does not save the operating point.
save_time	Time during transient analysis, when HSPICE saves the operating point. HSPICE requires a valid transient analysis statement, to save a DC operating point. Default = 0.

SEE ALSO:

.LOAD

.SENS

SYNTAX:

.SENS *ov1 <ov2 ...>*

EXAMPLE:

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

DESCRIPTION:

The **.SENS** command obtains DC small-signal sensitivities of output variables, for circuit parameters (you do not need to specify **.OP**).

If the input file includes a **.SENS** statement, HSPICE determines DC small-signal sensitivities for each specified output variable, relative to every circuit parameter. The sensitivity measurement is the partial derivative of each output variable, for a specified circuit element, measured at the operating point, and normalized to the total change in output magnitude. Therefore, the sum of the sensitivities of all elements is 100%. HSPICE calculates sensitivities for:

- resistors
- voltage sources
- current sources
- diodes
- BJTs (including Level 4, the VBIC95 model)
- MOSFETs (Level49 and Level53, Version=3.22).

You can perform only one **.SENS** analysis per simulation. If you specify more than one **.SENS** statement, HSPICE runs only the last .SENS statement.

The **.SENS** statement can generate very large amounts of output for large circuits.

.

Command Argument	Definition
ov1 ov2 ...	Branch currents, or nodal voltage, for DC component-sensitivity analysis

SEE ALSO:

.DC

.SHAPE

SYNTAX:

.SHAPE *sname Shape_Descriptor*

DESCRIPTION:

Use the **.SHAPE** statement to define a shape. The Field Solver uses the shape to describe a cross-section of the conductor.

Command Argument	Definition
sname	Shape name.
Shape_Descriptor	One of the following: • Rectangle • Circle • Strip • Polygon

SEE ALSO:

.FSOPTIONS
.MATERIAL
.SHAPE

.SHAPE (Defining Rectangles)

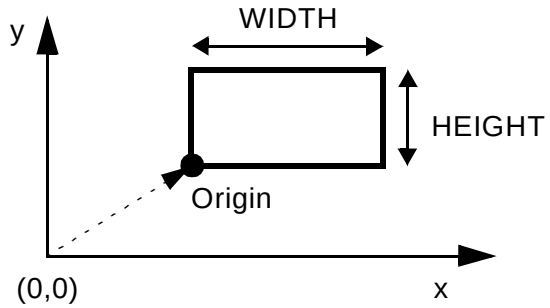
SYNTAX:

.SHAPE RECTANGLE WIDTH=*val* HEIGHT=*val* <NW=*val*> + <NH=*val*>

DESCRIPTION:

Use the RECTANGLE option to define a rectangle. Normally, you do not need to specify the NW and NH values because the field solver automatically sets these values, depending on the *accuracy* mode. You can specify both values, or specify only one of these values and let the solver determine the other.

Figure 2-2 Coordinates of a Rectangle



Command Argument	Definition
WIDTH	Width of the rectangle (size in the *x*-direction).
HEIGHT	Height of the rectangle (size in the *y*-direction).
NW	Number of horizontal (x) segments in a rectangle, with a specified width.
NH	Number of vertical (y) segments in a rectangle, with a specified height.

.SHAPE (Defining Circles)

SYNTAX:

.SHAPE CIRCLE RADIUS=*val* <N=*val*>

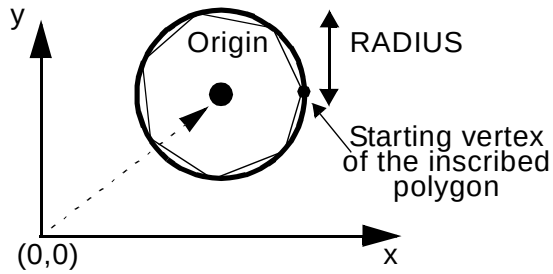
DESCRIPTION:

The CIRCLE option to defines a circle in the Field Solver. The Field Solver approximates a circle as an inscribed regular polygon, with *N* edges. The more edges, the more accurate the circle approximation is.

Do not use the CIRCLE descriptor to model actual polygons; instead use the POLYGON descriptor.

Normally, you do not need to specify the N value, because the field solver automatically sets this value, depending on the *accuracy* mode. But you can specify this value if you need to

Figure 2-3 Coordinates of a Circle



Command Argument	Definition
RADIUS	Radius of the circle.
N	Number of segments to approximate a circle with a specified radius.

.SHAPE (Defining Polygons)

SYNTAX:

.SHAPE POLYGON VERTEX=(*x1 y1 x2 y2* ...)
+ <N=(*n1*,*n2*,...)>

EXAMPLE 1:

The following rectangular polygon uses the default number of segments:

```
POLYGON VERTEX=(1 10 1 11 5 11 5 10)
```

EXAMPLE 2:

The following rectangular polygon uses five segments for each edge:

```
POLYGON VERTEX=(1 10 1 11 5 11 5 10) N=5
```

EXAMPLE 3:

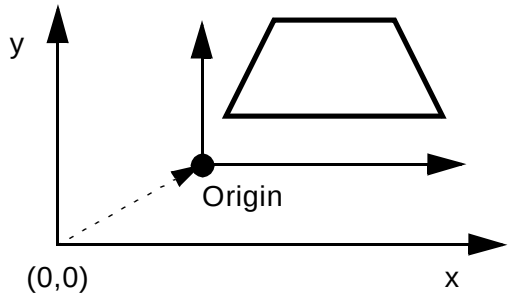
Rectangular polygon, using the different number of segments for each edge:

```
POLYGON VERTEX=(1 10 1 11 5 11 5 10) N=(5 3
     + 5 3)
```

DESCRIPTION:

The POLYGON command option defines a polygon in a Field Solver. The specified coordinates are within the local coordinate, with respect to the origin of a conductor.

Figure 2-4 Coordinates of a Polygon



Command Argument	Definition
VERTEX	(*x*, *y*) coordinates of vertices. Listed either in clockwise or counter-clockwise direction.
N	Number of segments that define the polygon, with the specified *X* and *Y* coordinates. You can specify a different *N* value for each edge. If you specify only one *N* value, then the Field Solver uses this value for *all* edges. For example, the first value of *N, n1*, corresponds to the number of segments for the edge from (*x1 y1*) to (*x2 y2*).

.SHAPE (Defining Strip Polygons)

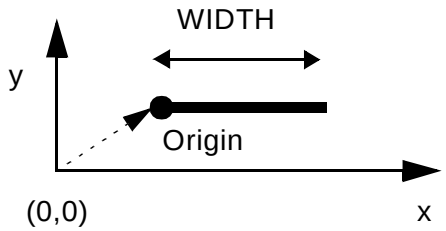
SYNTAX:

.SHAPE STRIP WIDTH=*val* <N=*val*>

DESCRIPTION:

Normally, you do not need to specify the N value, because the field solver automatically sets this value, depending on the *accuracy* mode. But you can specify this value if you need to. The field solver (filament method) does not support this shape.

Figure 2-5 Coordinates of a Strip Polygon



Command Argument	Definition
WIDTH	Width of the strip (size in the *x*-direction).
N	Number of segments that define the strip shape, with the specified width.

.STIM

SYNTAX:

General

.STIM <tran|ac|dc> PWL|DATA|VEC
+ <filename=*output_filename*> ...

PWL Source (Transient Analysis Only)

.STIM [tran] PWL [filename=*output_filename*]
+ [*name1*=] *ovar1* [*node1*=n+] [*node2*=n-]
+ [[*name2*=]*ovar2* [*node1*=n+] [*node2*=n-] ...]
+ [from=*val*] [to=*val*] [npoints=*val*]

.STIM [tran] PWL [filename=*output_filename*]
+ [*name1*=] *ovar1* [*node1*=n+] [*node2*=n-]
+ [[*name2*=]*ovar2* [*node1*=n+] [*node2*=n-] ...]
+ indepvar=[(]*t1* [*t2* ...])]

Data Card

.STIM [tran |ac |dc] DATA [filename=*output_filename*]
+ *dataname* [*name1*=] *ovar1*
+ [[*name2*=]*ovar2* ...] [from= *val*] [to=*val*]
+ [npoints=*val*] [indepout=*val*]

.STIM [tran |ac |dc] DATA [filename=*output_filename*]
+ *dataname* [*name1*=] *ovar1*
+ [[*name2*=]*ovar2* ...] indepvar=[(]*t1* [*t2* ...])]
+ [indepout=*val*]

Digital Vector File (Transient Analysis Only)

.**STIM** [tran] VEC [filename=*output_filename*]
+ vth=*val* vtl=*val* [voh=*val*] [vol=*val*]
+ [name1=] *ovar1* [[name2=] *ovar2* ...]
+ [from=*val*] [to=*val*] [npoints=*val*]

.STIM [tran] VEC [filename=*output_filename*]
+ vth=*val* vtl=*val* [voh=*val*] [vol=*val*]
+ [name1=] *ovar1* [[name2=] *ovar2* ...]
+ indepvar=[(]*t1* [*t2* ...[)]]

Brackets [ ] in the syntax enclose comments, which are optional.

DESCRIPTION:

You can use the **.STIM** statement to reuse the results (output) of one simulation, as input stimuli in a new simulation.

The **.STIM** statement specifies:

- Expected stimulus (PWL Source, DATA CARD, or VEC FILE).
- Signals to transform.
- Independent variables.

One **.STIM** statement produces one corresponding output file.

Table 2-1 .STIM PWL Source Arguments

Command Argument	Definition
tran	Transient simulation.
filename	Output file name. If you do not specify a file, HSPICE uses the input filename.
namei	PWL Source Name that you specify. The name must start with V (for a voltage source) or I (for a current source).
ovari	Output variable that you specify. *ovar* can be: • Node voltage. • Element current. • Parameter string. If you use a parameter string, you must specify *name1*. For example: v(1), i(r1), v(2,1), par('v(1)+v(2)')
node1	Positive terminal node name.
node2	Negative terminal node name.
from	Keyword; specifies the time to start output of simulation results. For transient analysis, uses the time units that you specified.
npoints	Number of output time points.
to	Keyword; specifies the time to end output of simulation results. For transient analysis, uses the time units that you specified. The *from* value can be greater than the *to* value.
indepvar	Keyword; specifies dispersed (independent-variable) time points. You must specify dispersed time points in *increasing* order.

Table 2-2 .STIM Data Card Arguments

Command Argument	Definition
tran | ac | dc	Selects the simulation type: transient, AC, or DC.
filename	Output file name. If you do not specify a file, HSPICE uses the input filename.
dataname	Name of the data card to generate.
from	Keyword; specifies the time to start output of simulation results. For transient analysis, uses the time units that you specified.
to	Keyword; specifies the time to end output of simulation results. For transient analysis, uses the time units that you specified.
namei	Name of a parameter of the data card to generate.
npoints	Number of output independent-variable points.
indepvar	Keyword; specifies dispersed independent-variable points.
indepout	Indicates whether to generate the independent variable column. • indepout, indepout = 1, or on, produces the independent variable column. You can specify the independent-variables in any order. • indepout= 0 or off (default) does not create an independent variable column. You can place the indepout field anywhere, after the ovari field.

Table 2-3 .STIM Vector File Arguments

Command Argument	Definition
namei	Signal name that you specify.
filename	Output file name. If you do not specify a file, HSPICE uses the input filename.
ovari	Output variable that you specify. *ovar* can only be a node voltage.
from	Keyword; specifies the time to start output of simulation results. For transient analysis, uses the time units that you specified.
to	Keyword; time to the end output of simulation results. For transient analysis, uses the specified time units.The *from* value can be greater than the *to* value.
npoints	Number of output time points.
indepvar	Keyword; specifies dispersed independent-variable points. You must specify dispersed time points in *increasing* order.
vth	High voltage threshold.
vtl	Low voltage threshold.
voh	Logic-high voltage for each output signal.
vol	Logic-low voltage for each output signal.
ovari	Output variable that you specify. *ovar* can be: • Node voltage. • Element current. • Element templates. • Parameter string. For example: v(1), i(r1), v(2,1), par('v(1)+v(2)'), LX1(m1), LX2(m1)

SEE ALSO:

.DOUT
.GRAPH
.MEASURE
.PLOT
.PRINT
.PROBE

.SUBCKT

SYNTAX:

.SUBCKT *subnam n1 <n2 n3 …> <parnam = val>*
.ENDS

.SUBCKT <*SubName*><*PinList*>[<*SubDefaultsList*>]
.ENDS

EXAMPLE 1:

```
*FILE SUB2.SP TEST OF SUBCIRCUITS
.OPTION LIST ACCT
  V1 1 0 1
.PARAM P5 = 5 P2 = 10
.SUBCKT SUB1 1 2 P4 = 4
  R1 1 0 P4
  R2 2 0 P5
  X1 1 2 SUB2 P6 = 7
  X2 1 2 SUB2
.ENDS
*
.MACRO SUB2 1 2 P6 = 11
  R1 1 2 P6
  R2 2 0 P2
.EOM
  X1 1 2 SUB1 P4 = 6
  X2 3 4 SUB1 P6 = 15
  X3 3 4 SUB2
*
.MODEL DA D CJA = CAJA CJP = CAJP VRB = -20
IS = 7.62E-18
+   PHI = .5 EXA = .5 EXP = .33
.PARAM CAJA = 2.535E-16 CAJP = 2.53E-16
.END
```

The preceding example defines two subcircuits: SUB1 and SUB2. These are resistor divider networks, whose resistance values are parameters (variables). The X1, X2,

and X3 statements call these subcircuits. Because the resistor values are different in each call, these three calls produce different subcircuits.

EXAMPLE 2:

```
.SUBCKT Inv a y Strength = 3
   Mp1 <MosPinList> pMosMod L = 1.2u W = 'Strength * 2u'
   Mn1 <MosPinList> nMosMod L = 1.2u W = 'Strength * 1u'
.ENDS
...
xInv0 a y0 Inv$ Default devices: p device = 6u,
   $ n device = 3u
xInv1 a y1 Inv Strength = 5$ p device = 10u, n
device = 5u
xInv2 a y2 Inv Strength = 1$ p device =  2u, n
device = 1u
...
```

This example implements an inverter that uses a *Strength* parameter. By default, the inverter can drive three devices. Enter a new value for the *Strength* parameter in the element line, to select larger or smaller inverters for the application.

DESCRIPTION:

You can create a subcircuit description for a commonly-used circuit, and include one or more references to the subcircuit in your netlist.

To define a subcircuit in your netlist, use the **.SUBCKT** statement.

When you use hierarchical subcircuits, you can pick default values for circuit elements in a **.SUBCKT** command. You can use this feature in cell definitions, to simulate the circuit

with typical values. Use the **.ENDS** statement to terminate a **.SUBCKT** statement.

Command Argument	Definition
subnam	Specifies a reference name for the subcircuit model call.
n1 …	Node numbers for external reference; cannot be the ground node (zero). Any element nodes that are in the subcircuit, but are not in this list, are strictly local, with three exceptions: • Ground node (zero). • Nodes assigned using BULK = node in MOSFET or BJT models. • Nodes assigned using the **.GLOBAL** statement.
parnam	A parameter name set to a value. Use only in the subcircuit. To override this value, assign it in the subcircuit call, or set a value in a **.PARAM** statement.
SubDefaultsList	<*SubParam1*>=<*Expression*> [<*SubParam2*>=<*Expression*>...]

SEE ALSO:

.ENDS
.EOM

.TEMP

SYNTAX:

.TEMP *t1 <t2 <t3 ...>>*

EXAMPLE 1:

```
.TEMP -55.0 25.0 125.0
```

The **.TEMP** statement sets the circuit temperatures for the entire circuit simulation. To simulate the circuit, using individual elements or model temperatures, HSPICE uses:

- Temperature, as set in the **.TEMP** statement.
- TNOM option setting (or the TREF model parameter).
- DTEMP element temperature.

EXAMPLE 2:

```
.TEMP 100
D1 N1 N2 DMOD DTEMP=30
D2 NA NC DMOD
R1 NP NN 100 TC1=1 DTEMP=-30

.MODEL DMOD D IS=1E-15 VJ=0.6 CJA=1.2E-13
+ CJP=1.3E-14 TREF=60.0
```

In this example:

- The **.TEMP** statement sets the circuit simulation temperature to 100°C.
- You do not specify TNOM, so it defaults to 25°C.
- The temperature of the diode is 30°C above the circuit temperature, as set in the DTEMP parameter.

That is:

- D1temp = 100°C + 30°C = 130°C.
- HSPICE simulates the D2 diode at 100°C.
- R1 simulates at 70°C.

Because the diode model statement specifies TREF at 60°C, HSPICE derates the specified model parameters by:

- 70°C (130°C - 60°C) for the D1 diode.
- 40°C (100°C - 60°C) for the D2 diode.
- 45°C (70°C - TNOM) for the R1 resistor.

DESCRIPTION:

To specify the circuit temperature for an HSPICE simulation, use the **.TEMP** statement, or the TEMP parameter in the **.DC**, **.AC**, and **.TRAN** statements. HSPICE compares the circuit simulation temperature against the reference temperature in the TNOM control option. HSPICE uses the difference between the circuit simulation temperature and the TNOM reference temperature to define derating factors for component values.

HSPICE permits only one temperature for the entire circuit. Multiple definitions of the **.TEMP** statements in a circuit behave as a sweep function.

Note: HSPICE allows multiple **.TEMP** statements in the netlist, and performs multiple DC, AC or TRAN analyses for each temperature. Do not set the temperature to the same value multiple times.

Command Argument	**Definition**
t1 t2	Temperatures, in °C, at which HSPICE simulates the circuit.

.TF

SYNTAX:

.TF *ov srcnam*

EXAMPLE:

```
.TF V(5,3) VIN
.TF I(VLOAD) VIN
```

For the first example, HSPICE computes the ratio of V(5,3) to VIN. This is the ratio of small-signal input resistance at VIN, to the small-signal output resistance (measured across nodes 5 and 3). If you specify more than one **.TF** statement in a single simulation, HSPICE runs only the last **.TF** statement.

DESCRIPTION:

The **.TF** command calculates DC small-signal values for transfer functions (ratio of output variable, to input source). You do not need to specify **.OP**.

The transfer function statement (**.TF**) defines small-signal output and input, for DC small-signal analysis. When you use the **.TF** statement, HSPICE computes:

- DC small-signal value of the transfer function (output/ input),.
- Input resistance.
- Output resistance.

Command Argument	Definition
ov	Small-signal output variable.
srcnam	Small-signal input source.

SEE ALSO:

.DC

.TITLE

SYNTAX:

.TITLE <*string_of_up_to_72_characters*>

or

<*string_of_up_to_72_characters*>

EXAMPLE:

```
.TITLE my-design_netlist
```

DESCRIPTION:

You set the simulation title in the first line of the input file. HSPICE always reads this line, and uses it as the title of the simulation, regardless of the line's contents. The simulation prints the title verbatim, in each section heading of the output listing file.

To set the title, you can place a **.TITLE** statement on the first line of the netlist. However, HSPICE does not *require* the **.TITLE** syntax.

In the second form of the syntax, the string is the first line of the input file. The first line of the input file is always the implicit title. If any statement appears as the first line in a file, simulation interprets it as a title, and does not execute it.

An **.ALTER** statement does not support using the **.TITLE** statement. To change a title for a **.ALTER** statement, place the title content in the **.ALTER** statement itself.

Command Argument	Definition
string	Any character string up to 72 characters long.

.TRAN

SYNTAX:

Single-Point Analysis

.TRAN *tincr1 tstop1 <tincr2 tstop2 ...tincrN tstopN>*
+ <START = *val*> <UIC>

Double-Point Analysis

.TRAN *tincr1 tstop1 <tincr2 tstop2 ...tincrN tstopN>*
+ <START = *val*> <UIC>
+ <SWEEP *var type np pstart pstop*>

.TRAN *tincr1 tstop1 <tincr2 tstop2 ...tincrN tstopN>*
+ <START = *val*> <UIC>
+ <SWEEP *var* START="*param_expr1*"
+ STOP="*param_expr2*"
+ STEP="*param_expr3*">

.TRAN *tincr1 tstop1 <tincr2 tstop2 ... tincrN tstop*N>
+ <START=*val*> <UIC>
+ <SWEEP *var start_expr stop_expr step_expr*>

Data-Driven Sweep

.TRAN DATA = *datanm*

.TRAN *tincr1 tstop1 <tincr2 tstop2 ...tincrN tstopN>*
+ <START = *val*> <UIC> <SWEEP DATA = *datanm*>

.TRAN DATA = *datanm*<SWEEP *var type np pstart pstop*>

.TRAN DATA=*datanm* <SWEEP *var* START="*param_expr1*"
+STOP="*param_expr2*" STEP="*param_expr3*">

.TRAN DATA=*datanm*
+ <SWEEP *var start_expr stop_expr step_expr*>

Monte Carlo Analysis

.TRAN *tincr1 tstop1 <tincr2 tstop2 ...tincrN tstopN>*
+ <START = *val*> <UIC><SWEEP MONTE = *val*>

Optimization

.TRAN DATA = *datanm* OPTIMIZE = *opt_par_fun*
+ RESULTS = *measnames* MODEL = *optmod*

.TRAN <DATA=*filename*> SWEEP OPTIMIZE=OPT*xxx*
+ RESULTS=*ierr1* ... *ierrn* MODEL=*optmod*

EXAMPLE 1:

```
.TRAN 1NS 100NS
```

This example performs and prints the transient analysis, every 1 ns, for 100 ns.

EXAMPLE 2:

```
.TRAN .1NS 25NS 1NS 40NS START = 10NS
```

This example performs the calculation every 0.1 ns, for the first 25 ns; and then every 1 ns, until 40 ns. Printing and plotting begin at 10 ns.

EXAMPLE 3:

```
.TRAN 10NS 1US UIC
```

This example performs the calculation every 10 ns, for 1 μs. This example bypasses the initial DC operating point calculation. It uses the nodal voltages, specified in the .IC statement (or by IC parameters in element statements), to calculate the initial conditions.

EXAMPLE 4:

```
.TRAN 10NS 1US UIC SWEEP TEMP -55 75 10
```

This example increases the temperature by 10 °C, through the range -55 °C to 75 °C. It also performs transient analysis, for each temperature.

EXAMPLE 5:

```
.TRAN 10NS 1US SWEEP load POI 3 1pf 5pf 10pf
```

This example analyzes each load parameter value, at 1 pF, 5 pF, and 10 pF.

EXAMPLE 6:

.TRAN data = *dataname*

This example is a data-driven time sweep. It uses a data file as the sweep input. If the parameters in the data statement are controlling sources, then a piecewise linear specification must reference them.

DESCRIPTION:

.TRAN starts a transient analysis, which simulates a circuit at a specific time.

Command Argument	**Definition**
DATA = datanm	Data name, referenced in the **.TRAN** statement.
MONTE = val	Produces a specified number (*val*) of randomly-generated values. HSPICE uses them to select parameters from a *Gaussian*, *Uniform*, or *Random Limit.*
np	Number of points, or number of points per decade or octave, depending on what keyword precedes it.
param_expr...	Expressions you specify: *param_expr1...param_exprN*.

Command Argument	Definition
pincr	Voltage, current, element, or model parameter; or any temperature increment value. If you set the *type* variation, use *np* (number of points), not *pincr*.
pstart	Starting voltage, current, or temperature; or any element or model parameter value. If you set the *type* variation to POI (list of points), use a list of parameter values, instead of *pstart pstop*.
pstop	Final voltage, current, or temperature; or element or model parameter value.
START	Time when printing or plotting begins. The START keyword is optional: you can specify a start time without the keyword. If you use **.TRAN** with **.MEASURE**, a non-zero START time can cause incorrect **.MEASURE** results. Do not use non-zero START times in **.TRAN** statements, when you also use **.MEASURE**.
SWEEP	Keyword. Indicates that **.TRAN** specifies a second sweep.
tincr1…	Specifies the printing or plotting increment for printer output, and the suggested computing increment for post-processing.
tstop1…	Time when a transient analysis stops incrementing by the first specified time increment (*tincr1*). If another tincr-tstop pair follows, analysis continues with a new increment.
UIC	Uses the nodal voltages specified in the .IC statement (or in the *IC* = parameters of the various element statements) to calculate initial transient conditions, rather than solving for the quiescent operating point.
type	Specifies any of the following keywords: • DEC – decade variation. • OCT – octave variation (the value of the designated variable is eight times its previous value). • LIN – linear variation. • POI – list of points.

Command Argument	Definition
var	Name of an independent voltage or current source, any element or model parameter, or the TEMP keyword (indicating a temperature sweep). You can use a source value sweep, referring to the source name (SPICE style). However, if you specify a parameter sweep, a .DATA statement, and a temperature sweep, you must choose a parameter name for the source value, and subsequently refer to it in the **.TRAN** statement. The parameter name must not start with V or I.

.UNPROTECT

SYNTAX:

.UNPROTECT

DESCRIPTION:

In HSPICE, the **.UNPROTECT** statement restores normal output functions that a **.PROTECT** statement restricted.

- Any elements and models located between **.PROTECT** and **.UNPROTECT** statements, inhibit the element and model listing from the LIST option.
- Neither the **.OPTION NODE** cross reference, nor the **.OP** operating point printout, list any nodes within the **.PROTECT** and **.UNPROTECT** statements.

SEE ALSO:

.PROTECT

.VEC

SYNTAX:

.VEC ‘*digital_vector_file*’

DESCRIPTION:

You can call a digital vector file from an HSPICE netlist. A digital vector file consists of three parts:

- *Vector Pattern Definition* section
- *Waveform Characteristics* section
- *Tabular Data* section.

The **.VEC** file must be a text file. If you transfer it between Unix and Windows, use *text* mode.

.WIDTH

SYNTAX:

.WIDTH OUT = {80 |132}

EXAMPLE:

```
.WIDTH OUT = 132 $ SPICE compatible style
.OPTION CO = 132  $ preferred style
```

DESCRIPTION:

Use the **.WIDTH** statement to define the print-out width in HSPICE.

Permissible values for OUT are 80 and 132. You can also use **.OPTION** CO to set the OUT value.

Command Argument	Definition
OUT	Output print width.

3

Options in HSPICE Netlists

The **.OPTION** command sets a wide variety of HSPICE simulation options. This chapter describes all of the simulation options that you can set, using the various forms of the **.OPTION** command.

All of the options described in this chapter start with **.OPTION**. For example, ACCT refers to the **.OPTION** ACCT command. **CO**=*x* refers to the **.OPTION** CO=*x* command, and so forth.

.OPTION

SYNTAX:

.OPTION *opt1 <opt2 opt3 ...>*

EXAMPLE:

```
.OPTION BRIEF $ Sets BRIEF to 1 (turns it on)
* Netlist, models,
...
.OPTION BRIEF = 0 $ Turns BRIEF off
```

This example sets the **BRIEF** option to 1, to suppress a printout. It then resets **BRIEF** to 0 later in the input file, to resume the printout.

DESCRIPTION:

You can modify various aspects of a Synopsys HSPICE simulation, including:

- output types
- accuracy
- speed
- convergence

To set control options, use **.OPTION** statements. You can set any number of options in one **.OPTION** statement, and you can include any number of **.OPTION** statements in an input netlist file.

The following pages list all **.OPTION** command options. The remainder of this chapter describes these options. For instructions on how to use options that are relevant to a specific simulation type, see the appropriate DC, transient, and AC analysis chapters in the *HSPICE Simulation and Analysis User Guide*.

Most options default to 0 (OFF) when you do not assign a value, using either **.OPTION** <*opt*> = <*val*> or the option with no assignment: **.OPTION** <*opt*>. Each option description in this section also shows the default value.

To reset options, set them to zero (**.OPTION** <*opt*> = 0). To redefine an option, enter a new **.OPTION** statement; HSPICE uses the last definition.

Command Argument	**Definition**
opt1 ...	Specifies any input control options. Many options are in the form <*opt*> = *x*, where <*opt*> is the option name and *x* is the value assigned to that option. This section describes all options.

The following **.OPTION** command options are available:

- General Control Options
- CPU Options
- Interface Options
- Analysis Options
- Error Options
- Version Option
- Model Analysis Options
- DC Opertaing Point, DC Sweep, and Pole/Zero Options
- Transient and AC Small Signal Analysis Options
- Transient Control Options

- Input/Output Options
- AC Control Options

General Control Options

.OPTION ACCT
.OPTION ACOUT
.OPTION ALT999 or ALT9999
.OPTION ALTCHK
.OPTION BEEP
.OPTION BINPRINT
.OPTION BRIEF
.OPTION CO
.OPTION INGOLD
.OPTION LENNAM
.OPTION LIST
.OPTION MEASDGT
.OPTION NODE
.OPTION NOELCK
.OPTION NOMOD
.OPTION NOPAGE
.OPTION NOTOP
.OPTION NUMDGT
.OPTION NXX
.OPTION OPTLST
.OPTION OPTS
.OPTION PATHNUM
.OPTION PLIM
.OPTION POST_VERSION
.OPTION SEARCH
.OPTION STATFL
.OPTION VERIFY

CPU Options

.OPTION CPTIME
.OPTION EPSMIN
.OPTION EXPMAX
.OPTION LIMTIM

Interface Options

.OPTION ARTIST
.OPTION CDS
.OPTION CSDF
.OPTION DLENCSDF
.OPTION MEASOUT
.OPTION MENTOR
.OPTION MONTECON
.OPTION POST
.OPTION PROBE
.OPTION PSF
.OPTION SDA
.OPTION ZUKEN

Analysis Options

.OPTION ASPEC
.OPTION FFTOUT
.OPTION LIMPTS
.OPTION PARHIER
.OPTION SEED
.OPTION SPICE

Error Options

.OPTION BADCHR
.OPTION DIAGNOSTIC
.OPTION NOWARN
.OPTION WARNLIMIT

Version Option

.OPTION H9007

Model Analysis Options

General Model Analysis Options

.OPTION DCAP
.OPTION MODSRH
.OPTION SCALE
.OPTION HIER_SCALE
.OPTION MODMONTE
.OPTION TNOM
.OPTION XDTEMP

MOSFET Model Analysis Options

.OPTION CVTOL
.OPTION DEFAD
.OPTION DEFAS
.OPTION DEFL
.OPTION DEFNRD
.OPTION DEFNRS
.OPTION DEFPD
.OPTION DEFPS
.OPTION DEFW
.OPTION SCALM
.OPTION WL

Inductor Model Analysis Options

.OPTION GENK
.OPTION KLIM

BJT and Diode Model Analysis Options

.OPTION EXPLI

DC Opertaing Point, DC Sweep, and Pole/Zero Options

DC Accuracy Options

.OPTION ABSH
.OPTION ABSI
.OPTION ABSMOS
.OPTION ABSTOL
.OPTION ABSVDC
.OPTION DI
.OPTION KCLTEST
.OPTION MAXAMP
.OPTION RELH
.OPTION RELI
.OPTION RELMOS
.OPTION RELV
.OPTION RELVDC

DC Matrix Options

.OPTION ITL1
.OPTION ITL2
.OPTION NOPIV
.OPTION PIVOT
.OPTION PIVREF
.OPTION PIVREL
.OPTION PIVTOL
.OPTION SPARSE

DC Pole/Zero I/O Options

.OPTION CAPTAB
.OPTION DCCAP
.OPTION VFLOOR

DC Convergence Options

.OPTION CONVERGE
.OPTION CSHDC
.OPTION DCFOR
.OPTION DCHOLD
.OPTION DCSTEP
.OPTION DCON
.OPTION DCTRAN
.OPTION DV
.OPTION GMAX
.OPTION GMINDC
.OPTION GRAMP
.OPTION GSHUNT
.OPTION ICSWEEP
.OPTION ITLPTRAN
.OPTION NEWTOL
.OPTION OFF
.OPTION RESMIN

DC Initialization Control Options

.OPTION ABSTOL
.OPTION CAPTAB
.OPTION CSHDC
.OPTION DCCAP
.OPTION DCFOR
.OPTION DCHOLD
.OPTION DCSTEP
.OPTION DV
.OPTION GRAMP
.OPTION GSHUNT
.OPTION ICSWEEP
.OPTION ITLPTRAN
.OPTION ITL1
.OPTION ITL2
.OPTION KCLTEST
.OPTION MAXAMP
.OPTION NEWTOL
.OPTION NOPIV
.OPTION OFF
.OPTION PIVOT
.OPTION PIVREF
.OPTION PIVTOL
.OPTION RESMIN
.OPTION SPARSE

Transient and AC Small Signal Analysis Options

Transient/AC Accuracy Options

.OPTION ABSH
.OPTION ABSV
.OPTION ACCURATE
.OPTION ACOUT
.OPTION CHGTOL
.OPTION CSHUNT
.OPTION DI
.OPTION GMIN
.OPTION GSHUNT
.OPTION MAXAMP
.OPTION RELH
.OPTION RELI
.OPTION RELQ
.OPTION RELTOL
.OPTION RELV
.OPTION RISETIME
.OPTION TRTOL
.OPTION VNTOL

Transient/AC Speed Options

.OPTION AUTOSTOP
.OPTION BKPSIZ
.OPTION BYPASS
.OPTION BYTOL
.OPTION FAST
.OPTION ITLPZ
.OPTION MBYPASS

Transient/AC Timestep Options

.OPTION ABSVAR
.OPTION DELMAX
.OPTION DVDT
.OPTION FS
.OPTION FT
.OPTION IMAX
.OPTION IMIN
.OPTION ITL3
.OPTION ITL4
.OPTION ITL5
.OPTION TIMERES

Transient/AC Algorithm Options

.OPTION DVTR
.OPTION IMAX
.OPTION IMIN
.OPTION ITL3
.OPTION ITL4
.OPTION ITL5
.OPTION LVLTIM
.OPTION MAXORD
.OPTION METHOD
.OPTION MU
.OPTION PURETP
.OPTION TRCON

.BIASCHK Options

.OPTION BIASFILE
.OPTION BIAWARN

Transient Control Options

Transient Control Method Options

.OPTION BYPASS
.OPTION CSHUNT
.OPTION DVDT
.OPTION GSHUNT
.OPTION INTERP
.OPTION ITRPRT
.OPTION MAXORD
.OPTION METHOD
.OPTION TRCON

Transient Control Tolerance Options

.OPTION ABSH
.OPTION ABSV
.OPTION ABSVAR
.OPTION ACCURATE
.OPTION BYTOL
.OPTION CHGTOL
.OPTION DI
.OPTION FAST
.OPTION MAXAMP
.OPTION MBYPASS
.OPTION MU
.OPTION RELH
.OPTION RELI
.OPTION RELQ
.OPTION RELTOL
.OPTION RELV
.OPTION RELVAR
.OPTION SLOPETOL
.OPTION TIMERES
.OPTION TRTOL
.OPTION VNTOL

Transient Control Limit Options

.OPTION AUTOSTOP
.OPTION BKPSIZ
.OPTION DELMAX
.OPTION DVTR
.OPTION FS
.OPTION FT
.OPTION GMIN
.OPTION IMAX
.OPTION IMIN
.OPTION ITL3
.OPTION ITL4
.OPTION ITL5
.OPTION RMAX
.OPTION RMIN
.OPTION VFLOOR

Transient Control Matrix Options

.OPTION GMIN
.OPTION PIVOT

Iteration Count Dynamic Timestep Options

.OPTION IMAX
.OPTION IMIN

Input/Output Options

.OPTION INTERP
.OPTION ITRPRT
.OPTION MEASFAIL
.OPTION MEASSORT
.OPTION PUTMEAS
.OPTION UNWRAP

AC Control Options

.OPTION ABSH
.OPTION ACOUT
.OPTION DI
.OPTION MAXAMP
.OPTION RELH
.OPTION UNWRAP

.OPTION ABSH

SYNTAX:

.OPTION ABSH=x

DESCRIPTION:

Sets the absolute current change, through voltage-defined branches (voltage sources and inductors). Use ABSH with DI and RELH, to check for current convergence. The default is 0.0.

.OPTION ABSI

SYNTAX:

.OPTION ABSI=x

DESCRIPTION:

Sets the absolute error tolerance for branch currents, in diodes, BJTs, and JFETs, during DC and transient analysis. Decrease ABSI, if accuracy is more important than convergence time.

To analyze currents less than 1 nanoamp, change ABSI to a value at least two orders of magnitude smaller than the minimum expected current.

The default is 1e-9 for KCLTEST = 0, or 1e-6 for KCLTEST = 1.

.OPTION ABSMOS

SYNTAX:

.OPTION ABSMOS=x

DESCRIPTION:

Current error tolerance (for MOSFET devices), in DC or transient analysis. The ABSMOS setting determines whether the drain-to-source current solution has converged. The drain-to-source current converged if:

- The difference between the drain-to-source current in the last iteration, versus the present iteration, is less than ABSMOS, or
- This difference is greater than ABSMOS, but the percent change is less than RELMOS.

If other accuracy tolerances also indicate convergence, HSPICE solves the circuit at that timepoint, and calculates the next timepoint solution. For low-power circuits, optimization, and single transistor simulations, set ABSMOS = 1e-12. Default is 1e-6 (amperes).

.OPTION ABSTOL

SYNTAX:

.OPTION ABSTOL=x

DESCRIPTION:

Sets the absolute error tolerance for branch currents, for DC and transient analysis. Decrease ABSTOL, if accuracy is more important than convergence time. ABSTOL is the same as ABSI.

SEE ALSO:

.OPTION ABSI

.OPTION ABSV

SYNTAX:

.OPTION ABSV=x

DESCRIPTION:

Sets absolute minimum voltage for DC and transient analysis. ABSV is the same as VNTOL. If accuracy is more critical than convergence, decrease ABSV. If you need voltages less than 50 microvolts, reduce ABSV to two orders of magnitude less than the smallest desired voltage. This ensures at least two significant digits. Typically, you do not need to change ABSV, except to simulate a high-voltage circuit. A reasonable value for 1000-volt circuits is 5 to 50 millivolts. The default is 50 (microvolts).

.OPTION ABSVAR

SYNTAX:

.OPTION ABSVAR=x

DESCRIPTION:

Sets the absolute limit for the maximum voltage change, from one time point to the next. Use this option with the DVDT algorithm. If the simulator produces a convergent solution that is greater than ABSVAR, then HSPICE discards the solution, sets the timestep to a smaller value, and recalculates the solution. This is called a timestep reversal. The default=0.5 (volts).

.OPTION ABSVDC

SYNTAX:

.OPTION ABSVDC=x

DESCRIPTION:

Sets the minimum voltage, for DC and transient analysis. If accuracy is more critical than convergence, decrease ABSVDC. If you need voltages less than 50 micro-volts, reduce ABSVDC, to two orders of magnitude less than the smallest voltage. This ensures at least two digits of significance. Typically, you do not need to change ABSVDC, unless you simulate a high-voltage circuit. For 1000-volt circuits, a reasonable value is 5 to 50 millivolts.

The default=VNTOL (VNTOL default = 50 mV).

SEE ALSO:

.OPTION VNTOL

.OPTION ACCT

SYNTAX:

.OPTION ACCT

.OPTION ACCT=[1|2]

EXAMPLE:

```
.OPTION ACCT=2
```

DESCRIPTION:

The ACCT option in HSPICE generates a detailed accounting report.

Value	Description
.OPTION ACCT	Enables reporting.
.OPTION ACCT = 1 (default)	Is the same as ACCT, without arguments.
.OPTION ACCT = 2	Enables reporting, and matrix statistic reporting.

.OPTION ACCURATE

SYNTAX:

.OPTION ACCURATE=x

DESCRIPTION:

Selects a time algorithm that uses LVLTIM = 3 and DVDT = 2, for circuits such as high-gain comparators. Use this option with circuits that combine high gain and large dynamic range, to guarantee accurate solutions in HSPICE. When set to 1, ACCURATE sets these control options:

- LVLTIM = 3
- DVDT = 2
- RELVAR = 0.2
- ABSVAR = 0.2
- FT = 0.2
- RELMOS = 0.01

The default is 0.

SEE ALSO:

.OPTION DVDT
.OPTION LVLTIM

.OPTION ACOUT

SYNTAX:

.OPTION ACOUT=x

DESCRIPTION:

AC output calculation method, for the difference in values of magnitude, phase, and decibels. Use these values for prints and plots. The default is 1.

The default (ACOUT = 1) selects the HSPICE method, which calculates the difference of the magnitudes of the values. The SPICE method, ACOUT = 0, calculates the magnitude of the differences in HSPICE.

.OPTION ALT999 or ALT9999

SYNTAX:

.OPTION ALT999

.OPTION ALT9999

DESCRIPTION:

This option was developed to allow the**.GRAPH** statement to create more output files when you ran **.ALTER** simulations.

This option is obsolete starting with version 2003.09. Without this option, HSPICE can now generate up to 10,000 unique files.

SEE ALSO:

.ALTER
.GRAPH

.OPTION ALTCHK

SYNTAX:

.OPTION ALTCHK=x

DESCRIPTION:

By default, HSPICE automatically reports topology errors in the latest elements, in your top-level netlist. It also reports errors in elements that you redefine, using the **.ALTER** statement (altered netlist).

To disable topology checking in redefined elements (that is, to check topology only in the top-level netlist, but not in the altered netlist), set:

```
.option altchk=0
```

By default, **.OPTION** ALTCHK is set to 1:

```
.option altchk=1
.option altchk
```

This enables topology checking, in elements that you redefine using the **.ALTER** statement.

.OPTION ASPEC

SYNTAX:

.OPTION ASPEC=x

DESCRIPTION:

Sets HSPICE to ASPEC-compatibility mode. When you set this option, the simulator reads ASPEC models and netlists, and the results are compatible. Default is 0 (HSPICE mode).

If you set ASPEC, the following model parameters default to ASPEC values:

- *ACM = 1:* Changes the default values for CJ, IS, NSUB, TOX, U0, and UTRA.
- *Diode Model:* TLEV = 1 affects temperature compensation for PB.
- *MOSFET Model:* TLEV = 1 affects PB, PHB, VTO, and PHI.
- *SCALM, SCALE:* Sets the model scale factor to microns, for length dimensions.
- *WL:* Reverses implicit order for stating width and length in a MOSFET statement. The default (WL = 0) assigns the length first, then the width.

SEE ALSO:

.OPTION SCALE
.OPTION SCALM
.OPTION WL

.OPTION ARTIST

SYNTAX:

.OPTION ARTIST=x

DESCRIPTION:

ARTIST = 2 enables the Cadence Analog Artist interface. This option requires a specific license. Supported on Sun Solaris 2.5/2.7/2.8, HPUX 10.20 and 11.20, IBM AIX 4.3 platforms only. This option is not available on the PC RedHat/Linux 7.2 platforms.

.OPTION AUTOSTOP

SYNTAX:

.OPTION AUTOSTOP

DESCRIPTION:

Stops a transient analysis in HSPICE, after calculating all TRIG-TARG, FIND-WHEN, and FROM-TO measure functions. This option can substantially reduce CPU time. You can use the AUTOSTOP option with any measure type. You can also use the result of the preceding measurement, as the next measured parameter.

Also terminates the simulation, after completing all **.MEASURE** statements. This is of special interest, when testing corners.

SEE ALSO:

.MEASURE

.OPTION BADCHR

SYNTAX:

.OPTION BADCHR

DESCRIPTION:

Generates a warning, if it finds a non-printable character in an input file.

.OPTION BEEP

SYNTAX:

.OPTION BEEP=x

DESCRIPTION:

BEEP=1 sounds an audible tone when simulation returns a message, such as info: hspice job completed.

BEEP=0 turns off the audible tone.

.OPTION BIASFILE

SYNTAX:

.OPTION BIASFILE=x

EXAMPLE:

```
OPTION BIASFILE='biaschk/mos.bias'
```

DESCRIPTION:

If you use this option, HSPICE outputs the results of all **.BIASCHK** commands to a file that you specify. If you do not set this option, HSPICE outputs the **.BIASCHK** results to the **.lis* file.

SEE ALSO:

.BIASCHK

.OPTION BIAWARN

SYNTAX:

.OPTION BIAWARN=x

EXAMPLE:

```
.OPTION BIAWARN=1
```

DESCRIPTION:

For the BIAWARN option:

- If you set this option to 1, HSPICE immediately outputs a warning message, when any local max bias voltage exceeds the limit during transient analysis. After this transient analysis, HSPICE outputs the results summary, as filtered by noise.
- If you set this option to 0 (the default), HSPICE does not output a warning message during transient analysis. HSPICE outputs the results, after this transient analysis.

.OPTION BINPRINT

SYNTAX:

.OPTION BINPRINT

DESCRIPTION:

Outputs the binning parameters of the CMI MOSFET model. Currently available only for Level 57.

.OPTION BKPSIZ

SYNTAX:

.OPTION BKPSIZ=x

DESCRIPTION:

Sets the size of the breakpoint table. The default is 5000. This is an old option, provided only for backward-compatibility.

.OPTION BRIEF

SYNTAX:

.OPTION BRIEF=x

DESCRIPTION:

Stops printback of the data file, until HSPICE finds an **.OPTION** BRIEF = 0, or the **.END** statement. It also resets the LIST, NODE, and OPTS options, and sets NOMOD. BRIEF = 0 enables printback. The NXX option is the same as BRIEF.

SEE ALSO:

.END

.OPTION BYPASS

SYNTAX:

.OPTION BYPASS=x

DESCRIPTION:

Bypasses model evaluations, if the terminal voltages do not change. Can be 0 (off) or 1 (on). To speed-up simulation, this option does not update the status of latent devices. To enable bypassing, set **.OPTION** BYPASS = 1, for MOSFETs, MESFETs, JFETs, BJTs, or diodes. Default = 1.

Use the BYPASS algorithm cautiously. Some circuit types might not converge, and might lose accuracy in transient analysis and operating-point calculations.

.OPTION BYTOL

SYNTAX:

.OPTION BYTOL=x

DESCRIPTION:

Specifies a voltage tolerance, at which a MOSFET, MESFET, JFET, BJT, or diode becomes latent. HSPICE does not update status of latent devices. The default = MBYPASS x VNTOL.

SEE ALSO:

.OPTION MBYPASS
.OPTION VNTOL

.OPTION CAPTAB

SYNTAX:

.OPTION CAPTAB

DESCRIPTION:

Prints table of single-plate node capacitances, for diodes, BJTs, MOSFETs, JFETs, and passive capacitors, at each operating point.

.OPTION CDS

SYNTAX:

.OPTION CDS=x

DESCRIPTION:

CDS = 2 produces a Cadence WSF (ASCII format) post-analysis file, for Opus™. This option requires a specific license. The CDS option is the same as the SDA option.

.OPTION CHGTOL

SYNTAX:

.OPTION CHGTOL=x

DESCRIPTION:

Sets a charge error tolerance, if you set LVLTIM = 2. Use CHGTOL with RELQ to set the absolute and relative charge tolerance for all HSPICE capacitances. The default=1e-15 (coulomb).

SEE ALSO:

.OPTION CHGTOL
.OPTION LVLTIM
.OPTION RELQ

.OPTION CO

SYNTAX:

.OPTION CO=<*column_width*>

EXAMPLE:

```
* Narrow print-out (default)
.OPTION CO=80

* Wide print-out
.OPTION CO=132
```

DESCRIPTION:

The number of output variables that print on a single line of output, is a function of the number of columns. Use **.OPTION** CO to set the column width for print-outs in HSPICE.

You can set up to five output variables per 80-column output, and up to eight output variables per 132-column output, with twelve characters per column. HSPICE automatically creates additional print statements and tables, for all output variables beyond the number that the CO option specifies. The default is 80.

Command Argument	Definition
column_width	The number of characters in a single line of output.

SEE ALSO:

.WIDTH

.OPTION CONVERGE

SYNTAX:

.OPTION CONVERGE=x

DESCRIPTION:

Invokes different methods to solve non-convergence problems.

- CONVERGE = -1 : Use with DCON = -1, to disable autoconvergence.
- CONVERGE = 0 : Autoconvergence (default).
- CONVERGE = 1 : Uses the Damped Pseudo Transient algorithm. If simulation does not converge within the set CPU time (in the CPTIME control option), then simulation halts.
- CONVERGE = 2 : Uses a combination of DCSTEP and GMINDC ramping. Not used in the autoconvergence flow.
- CONVERGE = 3 : Invokes the source-stepping method. Not used in the autoconvergence flow.
- CONVERGE = 4 : Uses the *gmath* ramping method.

Even you did not set it in an **.OPTION** statement, the CONVERGE option activates if a matrix floating-point overflows, or if HSPICE reports a *timestep too small* error. Default = 0.

If a matrix floating-point overflows, then CONVERGE = 1.

.OPTION CPTIME

SYNTAX:

.OPTION CPTIME=x

DESCRIPTION:

Sets the maximum CPU time, in seconds, allotted for this simulation job. When the time allowed for the job exceeds CPTIME, HSPICE prints or plots the results up to that point, and concludes the job. Use this option if you are uncertain how long the simulation will take, especially when you debug new data files. Also see LIMTIM. Default is 1e7 (400 days).

.OPTION CSDF

SYNTAX:

.OPTION CSDF=x

DESCRIPTION:

Selects Common Simulation Data Format (Viewlogic-compatible graph data file format).

.OPTION CSHDC

SYNTAX:

.OPTION CSHDC=x

DESCRIPTION:

The same option as CSHUNT; use only with the CONVERGE option.

SEE ALSO:

.OPTION CONVERGE
.OPTION CSHUNT

.OPTION CSHUNT

SYNTAX:

.OPTION CSHUNT=x

DESCRIPTION:

Capacitance added from each node to ground, in HSPICE. Add a small CSHUNT to each node, to solve internal timestep too small problems, caused by high-frequency oscillations or numerical noise. The default=0.

.OPTION CVTOL

SYNTAX:

.OPTION CVTOL=x

DESCRIPTION:

Changes the number of numerical integration steps, when calculating the gate capacitor charge for a MOSFET, using CAPOP = 3. See the discussion of CAPOP = 3 in the “Overview of MOSFETS” chapter of the *HSPICE Elements and Device Models Manual*, for explicit equations and discussion.

.OPTION D_IBIS

SYNTAX:

.OPTION D_IBIS=’*ibis_files_directory*’

EXAMPLE:

```
.OPTION d_ibis='/home/user/ibis/models'
```

DESCRIPTION:

The **.OPTION** D_IBIS option specifies the directory containing the IBIS files. If you specify several directories, then the simulation looks for IBIS files in the local directory (the directory from which you run the simulation). It then checks the directories specified through **.OPTION** D_IBIS in the order that **.OPTION** cards appear in the netlist. You can use the D_IBIS option to specify up to four directories.

.OPTION DCAP

SYNTAX:

.OPTION DCAP

DESCRIPTION:

Selects equations, which HSPICE uses to calculate depletion capacitance for Level 1 and 3 diodes, and BJTs. The *HSPICE Elements and Device Models Manual* describes these equations.

.OPTION DCCAP

SYNTAX:

.OPTION DCCAP=x

DESCRIPTION:

Generates C-V plots. Prints capacitance values of a circuit (both model and element), during a DC analysis. You can use a DC sweep of the capacitor, to generate C-V plots. Default = 0 (off).

.OPTION DCFOR

SYNTAX:

.OPTION DCFOR=x

DESCRIPTION:

Use with **.OPTION** DCHOLD and the **.NODESET** statement, to enhance DC convergence.

DCFOR sets the number of iterations to calculate, after a circuit converges in the steady state. The number of iterations after convergence is usually zero, so DCFOR adds iterations (and computation time) to the DC circuit solution. DCFOR ensures that a circuit actually, not falsely, converges. The default is 0.

SEE ALSO:

.NODESET
.OPTION DCHOLD

.OPTION DCHOLD

SYNTAX:

.OPTION DCHOLD=x

DESCRIPTION:

Use DCFOR and DCHOLD together, to initialize DC analysis. DCFOR and DCHOLD enhance the convergence properties of a DC simulation. DCFOR and DCHOLD work with the **.NODESET** statement. Default is 1.

DCHOLD specifies how many iterations to hold a node, at the **.NODESET** voltage values. The effects of DCHOLD on convergence differ, according to the DCHOLD value, and the number of iterations before DC convergence.

If a circuit converges in the steady state, in fewer than DCHOLD iterations, the DC solution includes the values set in **.NODESET**.

If a circuit requires more than DCHOLD iterations to converge, HSPICE ignores the values set in the **.NODESET** statement, and calculates the DC solution, using the **.NODESET** fixed-source voltages open circuited.

SEE ALSO:

.NODESET
.OPTION DCFOR

.OPTION DCON

SYNTAX:

.OPTION DCON=x

DESCRIPTION:

If a circuit cannot converge, HSPICE automatically sets DCON = 1, and calculates the following:

$$DV = \max\left(0.1, \frac{V_{max}}{50}\right), \text{ if } DV = 1000$$

$$GRAMP = \max\left(6, \log_{10}\left(\frac{I_{max}}{GMINDC}\right)\right) \qquad ITL1 = ITL1 + 20 \cdot GRAMP$$

V_{max} is the maximum voltage, and I_{max} is the maximum current.

- If the circuit still cannot converge, HSPICE sets DCON = 2, which sets DV = 1e6.
- If the circuit uses discontinuous models or uninitialized flip-flops, simulation might not converge. Set DCON = -1 and CONVERGE = -1, to disable autoconvergence. HSPICE lists all non-convergent nodes and devices.

SEE ALSO:

.OPTION CONVERGE

.OPTION DCSTEP

SYNTAX:

.OPTION DCSTEP=x

DESCRIPTION:

Converts DC model and element capacitors to a conductance, to enhance DC convergence properties. HSPICE divides the value of the element capacitors by DCSTEP, to model DC conductance. The default is 0 (seconds).

.OPTION DCTRAN

SYNTAX:

.OPTION DCTRAN=x

DESCRIPTION:

Invokes different methods to solve non-convergence problems. DCTRAN is an alias for CONVERGE.

SEE ALSO:

.OPTION CONVERGE

.OPTION DEFAD

SYNTAX:

.OPTION DEFAD=x

DESCRIPTION:

The default MOSFET drain diode area in HSPICE. The default=0.

.OPTION DEFAS

SYNTAX:

.OPTION DEFAS=x

DESCRIPTION:

The default MOSFET source diode area in HSPICE. The default=0.

.OPTION DEFL

SYNTAX:

.OPTION DEFL=x

DESCRIPTION:

The default MOSFET channel length in HSPICE. The default=$1e^{-4}$m.

.OPTION DEFNRD

SYNTAX:

.OPTION DEFNRD=x

DESCRIPTION:

The default number of squares for the drain resistor on a MOSFET. The default is 0.

.OPTION DEFNRS

SYNTAX:

.OPTION DEFNRS=x

DESCRIPTION:

The default number of squares for the source resistor on a MOSFET. The default is 0.

.OPTION DEFPD

SYNTAX:

.OPTION DEFPD=x

DESCRIPTION:

The default MOSFET drain diode perimeter in HSPICE. The default is 0.

.OPTION DEFPS

SYNTAX:

.OPTION DEFPS=x

DESCRIPTION:

The default MOSFET source diode perimeter in HSPICE. The default is 0.

.OPTION DEFW

SYNTAX:

.OPTION DEFW=x

DESCRIPTION:

The default MOSFET channel width in HSPICE. The default is $1e^{-4}$m.

.OPTION DELMAX

SYNTAX:

.OPTION DELMAX=x

DESCRIPTION:

Sets the maximum Delta of the internal timestep. HSPICE automatically sets the DELMAX value, based on timestep control factors. The initial DELMAX value, shown in the HSPICE output listing, is generally not the value used for simulation.

.OPTION DI

SYNTAX:

.OPTION DI=x

DESCRIPTION:

Sets the maximum iteration-to-iteration current change, through voltage-defined branches (voltage sources and inductors). Use this option only if the value of the ABSH control option is greater than 0. The default is 0.0.

SEE ALSO:

.OPTION ABSH

.OPTION DIAGNOSTIC

SYNTAX:

.OPTION DIAGNOSTIC

DESCRIPTION:

Logs the occurrence of negative model conductances.

.OPTION DLENCSDF

SYNTAX:

.OPTION DLENCSDF=x

DESCRIPTION:

If you use the Common Simulation Data Format (Viewlogic graph data file format) as the output format, this digit length option specifies how many digits to include, in scientific notation (exponents), or to the right of the decimal point. Valid values are any integer from 1 to 10, and the default is 5.

If you assign a floating decimal point, or if you specify less than 1 or more than 10 digits, HSPICE uses the default. For example, it places 5 digits to the right of a decimal point.

.OPTION DV

SYNTAX:

.OPTION DV=x

DESCRIPTION:

Maximum iteration-to-iteration voltage change, for all circuit nodes, in both DC and transient analysis. High-gain bipolar amplifiers can require values of 0.5 to 5.0, to achieve a stable DC operating point. Large CMOS digital circuits frequently require about 1 volt. The default is 1000 (or 1e6 if DCON = 2).

.OPTION DVDT

SYNTAX:

.OPTION DVDT=x

DESCRIPTION:

Adjusts the timestep, based on rates of change for node voltage. The default is 4.

- 0 - original algorithm
- 1 - fast
- 2 - accurate
- 3,4 - balance speed and accuracy

.OPTION DVTR

SYNTAX:

.OPTION DVTR=x

DESCRIPTION:

Limits voltage in transient analysis. The default is 1000.

.OPTION EPSMIN

SYNTAX:

.OPTION EPSMIN=x

DESCRIPTION:

Specifies the smallest number that a computer can add or subtract, a constant value. Default is 1e-28.

.OPTION EXPLI

SYNTAX:

.OPTION EXPLI=x

DESCRIPTION:

Current-explosion model parameter. PN junction characteristics, above the explosion current, are linear. HSPICE determines the slope at the explosion point. This improves simulation speed and convergence.

The default is 0.0 amp/AREAeff.

.OPTION EXPMAX

SYNTAX:

.OPTION EXPMAX=x

DESCRIPTION:

Specifies the largest exponent that you can use for an exponential, before overflow occurs. Typical value for an IBM platform is `350`.

.OPTION FAST

SYNTAX:

.OPTION FAST

DESCRIPTION:

Sets additional options, which increase simulation speed, with minimal loss of accuracy.

To speed-up simulation, this option does not update the status of latent devices. Use this option for MOSFETs, MESFETs, JFETs, BJTs, and diodes. Default is 0.

A device is latent, if its node voltage variation (from one iteration to the next) is less than the value of either the BYTOL control option, or the BYPASSTOL element parameter. (If FAST is on, HSPICE sets BYTOL to different values, for different types of device models.)

Besides the FAST option, you can also use the NOTOP and NOELCK options, to reduce input pre-processing time. Increasing the value of the MBYPASS or BYTOL option, also helps simulations to run faster, but can reduce accuracy.

SEE ALSO:

.OPTION BYTOL
.OPTION MBYPASS
.OPTION NOELCK
.OPTION NOTOP

.OPTION FFTOUT

SYNTAX:

.OPTION FFTOUT=x

DESCRIPTION:

Prints 30 harmonic fundamentals, sorted by size, THD, SNR, and SFDR, but only if you specify a **.OPTION** FFTOUT statement and a **.FFT** freq=xxx statement.

SEE ALSO:

.FFT

.OPTION FS

SYNTAX:

.OPTION FS=x

DESCRIPTION:

Decreases Delta (internal timestep) by the specified fraction of a timestep (TSTEP), for the first time point of a transient. Decrease the FS value to help circuits that have timestep convergence difficulties. DVDT = 3 uses FS to control the timestep.

Delta = FS¥[MIN(TSTEP, DELMAX, BKPT)]

- You specify DELMAX.
- BKPT is related to the breakpoint of the source.
- The **.TRAN** statement sets TSTEP. Default = 0.25.

SEE ALSO:

.OPTION DVDT
.TRAN

.OPTION FT

SYNTAX:

.OPTION FT=x

DESCRIPTION:

Decreases Delta (the internal timestep), by a specified fraction of a timestep (TSTEP), for an iteration set that does not converge. If DVDT = 2 or DVDT = 4, FT controls the timestep. The default = 0.25.

SEE ALSO:

.OPTION DVDT

.OPTION GENK

SYNTAX:

.OPTION GENK=x

DESCRIPTION:

Automatically computes second-order mutual inductance, for several coupled inductors. The default=1, which enables the calculation.

.OPTION GMAX

SYNTAX:

.OPTION GMAX=x

DESCRIPTION:

Conductance, in parallel with a current source, for **.IC** and **.NODESET** initialization circuitry. Some large bipolar circuits require you to set GMAX=1, for convergence. The default=100 (mho).

SEE ALSO:

.IC
.NODESET

.OPTION GMIN

SYNTAX:

.OPTION GMIN=x

DESCRIPTION:

Minimum conductance added to all PN junctions, for a time sweep in transient analysis. The default is 1e-12.

.OPTION GMINDC

SYNTAX:

.OPTION GMINDC=x

DESCRIPTION:

Conductance in parallel to all pn junctions and MOSFET nodes except *gate*, for DC analysis. GMINDC helps overcome DC convergence problems, caused by low values of off-conductance, for pn junctions and MOSFETs. You can use GRAMP to reduce GMINDC, by one order of magnitude, for each step. Set GMINDC between 1e-4 and the PIVTOL value. The default is 1e-12.

Large values of GMINDC can cause unreasonable circuit response. If your circuit requires large values to converge, suspect a bad model or circuit. If a matrix floating-point overflows, and if GMINDC is 1.0e-12 or less, HSPICE sets it to 1.0e-11. HSPICE manipulates GMINDC in auto-converge mode.

SEE ALSO:

.OPTION PIVTOL

.OPTION GRAMP

SYNTAX:

.OPTION GRAMP=x

DESCRIPTION:

HSPICE sets this value during auto-convergence (default=0). Use GRAMP, with the GMINDC option, to find the smallest GMINDC value that results in DC convergence.

GRAMP specifies a conductance range, over which DC operating point analysis sweeps GMINDC. HSPICE replaces GMINDC values over this range, simulates each value, and uses the lowest GMINDC value where the circuit converges in a steady state.

If you sweep GMINDC between 1e-12 mhos (default) and 1e-6 mhos, GRAMP is 6 (value of the exponent difference, between the default and the maximum conductance limit). In this example:

- HSPICE first sets GMINDC to 1e-6 mhos, and simulates the circuit.
- If circuit simulation converges, HSPICE sets GMINDC to 1e-7 mhos, and simulates the circuit.
- The sweep continues until HSPICE simulates all values of the GRAMP ramp.

If the combined GMINDC and GRAMP conductance is greater than 1e-3 mho, false convergence can occur.

SEE ALSO:

.OPTION GMINDC

.OPTION GSHUNT

SYNTAX:

.OPTION GSHUNT=x

DESCRIPTION:

Conductance, added from each node to ground. Default is zero. Add a small GSHUNT to each node, to help solve Timestep too small problems, caused by either high-frequency oscillations or numerical noise.

.OPTION H9007

SYNTAX:

.OPTION H9007

DESCRIPTION:

Sets default values for general-control options, to correspond to values for HSPICE H9007D. If you set this option, HSPICE does not use the EXPLI model parameter.

SEE ALSO:

.OPTION EXPLI

.OPTION HIER_SCALE

SYNTAX:

.OPTION HIER_SCALE=x

DESCRIPTION:

If you set the HIER_SCALE option, you can use the S parameter to scale sub-circuits.

- 0 interprets S as a user-defined parameter.
- 1 interprets S as a scale parameter.

.OPTION ICSWEEP

SYNTAX:

.OPTION ICSWEEP=x

DESCRIPTION:

Saves the current analysis result of a parameter or temperature sweep, as the starting point in the next analysis in the sweep.

- If ICSWEEP = 1 (default), the next analysis uses the current results.
- If ICSWEEP = 0, the next analysis does not use the results of the current analysis.

.OPTION IMAX

SYNTAX:

.OPTION IMAX=x

DESCRIPTION:

Maximum timestep, in timestep algorithms for transient analysis. IMAX sets the maximum iterations, to obtain a convergent solution at a timepoint. If the number of iterations needed is greater than IMAX, the internal timestep (Delta) decreases, by a factor equal to the FT transient control option. HSPICE uses the new timestep to calculate a new solution. IMAX also works with the IMIN transient control option. IMAX is the same as ITL4. The default is 8.0.

.OPTION IMIN

SYNTAX:

.OPTION IMIN=x

DESCRIPTION:

Minimum timestep, in timestep algorithms for transient analysis. IMIN is the minimum number of iterations required, to obtain convergence. If the number of iterations is less than IMIN, the internal timestep (Delta) doubles.

Use this option to decrease simulation times, in circuits where the nodes are stable most of the time (such as digital circuits). If the number of iterations is greater than IMIN, the timestep stays the same, unless the timestep exceeds the IMAX option. IMIN is the same as ITL3. The default is 3.0.

SEE ALSO:

.OPTION IMAX

.OPTION INGOLD

SYNTAX:

.OPTION INGOLD=[0|1|2]

EXAMPLE:

```
.OPTION INGOLD=2
```

DESCRIPTION:

By default, HSPICE prints variable values in engineering notation:

```
F = 1e-15        M = 1e-3
P = 1e-12        K = 1e3
N = 1e-9         X = 1e6
U = 1e-6         G = 1e9
```

In contrast to exponential form, engineering notation provides two to three extra significant digits, and aligns columns to facilitate comparison. To obtain output in exponential form, specify **.OPTION** INGOLD = 1 or 2.

Value	Description	Defaults
INGOLD = 0 (default)	Engineering Format	1.234K 123M
INGOLD = 1	G Format (fixed and exponential)	1.234e+03 .123
INGOLD = 2	E Format (exponential SPICE)	1.234e+03 .123e-1

SEE ALSO:

.OPTION MEASDGT

.OPTION INTERP

SYNTAX:

.OPTION INTERP=x

DESCRIPTION:

Limits output for post-analysis tools, such as Cadence or Zuken, to only the **.TRAN** timestep intervals. By default, HSPICE outputs all convergent iterations. INTERP typically produces a much smaller design *.tr#* file.

Use INTERP = 1 with caution, when the netlist includes **.MEASURE** statements. To compute measure statements, HSPICE uses the post-processing output. Reducing post-processing output can lead to interpolation errors, in measure results.

When you run data-driven transient analysis (**.TRAN** DATA) in an optimization routine, HSPICE forces INTERP to 1. All measurement results are at the time points specified in the data-driven sweep. To measure only at converged internal timesteps (for example, to calculate the AVG or RMS), set ITRPRT = 1.

SEE ALSO:

.MEASURE
.OPTION ITRPRT
.TRAN

.OPTION ITL1

SYNTAX:

.OPTION ITL1=x

DESCRIPTION:

Maximum DC iteration limit. Increasing this value rarely improves convergence in small circuits. Values as high as 400 have resulted in convergence for some large circuits with feedback (such as operational amplifiers and sense amplifiers). However, to converge, most models do not require more than 100 iterations. Set **.OPTION** ACCT to list how many iterations an operating point requires. The default is 200.

SEE ALSO:

.OPTION ACCT

.OPTION ITL2

SYNTAX:

.OPTION ITL2=x

DESCRIPTION:

Iteration limit for the DC transfer curve. Increasing this limit improves convergence, only for very large circuits. Default is 50.

.OPTION ITL3

SYNTAX:

.OPTION ITL3=x

DESCRIPTION:

Minimum timestep, in timestep algorithms for transient analysis. ITL3 is the minimum number of iterations required, to obtain convergence. If the number of iterations is less than ITL3, the internal timestep (Delta) doubles.

Use this option to decrease simulation times, in circuits where the nodes are stable most of the time (such as digital circuits). If the number of iterations is greater than IMIN, the timestep stays the same, unless the timestep exceeds the IMAX option. ITL3 is the same as IMIN. The default is 3.0.

SEE ALSO:

.OPTION IMAX

.OPTION ITL4

SYNTAX:

.OPTION ITL4=x

DESCRIPTION:

Maximum timestep, in timestep algorithms for transient analysis. ITL4 sets the maximum iterations, to obtain a convergent solution at a timepoint. If the number of iterations needed is greater than ITL4, the internal timestep (Delta) decreases, by a factor equal to the FT transient control option. HSPICE uses the new timestep to calculate a new solution. ITL4 also works with the IMIN transient control option. ITL4 is the same as IMAX. The default is 8.0.

SEE ALSO:

.OPTION IMAX

.OPTION ITL5

SYNTAX:

.OPTION ITL5=x

DESCRIPTION:

Sets an iteration limit for transient analysis. If a circuit uses more than ITL5 iterations, the program prints all results, up to that point. The default (0.0) allows an infinite number of iterations.

.OPTION ITLPTRAN

SYNTAX:

.OPTION ITLPTRAN=x

DESCRIPTION:

Controls the iteration limit used in the final try of the pseudo-transient method, in OP or DC analysis. If simulation fails in the final try of the pseudo-transient method, enlarge this option. The default is 30.

.OPTION ITLPZ

SYNTAX:

.OPTION ITLPZ=x

DESCRIPTION:

Sets the iteration limit for pole/zero analysis. The default is 100.

.OPTION ITRPRT

SYNTAX:

.OPTION ITRPRT

DESCRIPTION:

Prints output variables at their internal time points. This option might generate a long output list.

.OPTION KCLTEST

SYNTAX:

.OPTION KCLTEST=x

DESCRIPTION:

Activates KCL (Kirchhoff's Current Law) test. increases simulation time, especially for large circuits, but very accurately checks the solution. Default=0.

If you set this value to 1, HSPICE sets these options:

- Sets RELMOS and ABSMOS options to 0 (off).
- Sets ABSI to 1e-6 A.
- Sets RELI to 1e-6.

To satisfy the KCL test, each node must satisfy this condition:

$$\left|\Sigma i_b\right| < \text{RELI} \cdot \Sigma\left|i_b\right| + \text{ABSI}$$

In this equation, the ibs are the node currents.

SEE ALSO:

.OPTION ABSI
.OPTION RELI
.OPTION RELMOS

.OPTION KLIM

SYNTAX:

.OPTION KLIM=x

DESCRIPTION:

This option sets the minimum mutual inductance, below which automatic second-order mutual inductance calculation no longer proceeds. KLIM is unitless (analogous to coupling strength, specified in the K Element). Typical KLIM values are between .5 and 0.0. The default is 0.01.

.OPTION LENNAM

SYNTAX:

.OPTION LENNAM=x

DESCRIPTION:

Maximum length of names, in the printout of operating point analysis results. Default is 8, and the maximum x value=1024.

.OPTION LIMPTS

SYNTAX:

.OPTION LIMPTS=x

DESCRIPTION:

Number of points to print or plot in AC analysis. You do not need to set LIMPTS for DC or transient analysis. HSPICE spools the output file to disk. Default=2001.

.OPTION LIMTIM

SYNTAX:

.OPTION LIMTIM=x

DESCRIPTION:

Amount of CPU time reserved to generate prints and plots, if a CPU time limit (CPTIME = x) terminates simulation. Default=2 (seconds), normally sufficient for short printouts and plots.

SEE ALSO:

.OPTION CPTIME

.OPTION LIST

SYNTAX:

.OPTION LIST

DESCRIPTION:

This option produces an element summary of the input data to print, and calculates effective sizes of elements and the key values.

- BRIEF suppresses the LIST option.
- VERIFY is an alias for LIST

SEE ALSO:

.OPTION BRIEF
.OPTION UNWRAP
.OPTION VERIFY

.OPTION LVLTIM

SYNTAX:

.OPTION LVLTIM=x

DESCRIPTION:

Selects the timestep algorithm, for transient analysis.

- LVLTIM = 1 (default) uses the DVDT timestep algorithm.
- LVLTIM = 2 uses the timestep algorithm for local truncation error.
- LVLTIM = 3 uses the DVDT timestep algorithm with timestep reversal.
- To use the GEAR method of numerical integration and linearization, select LVLTIM = 2.
- To use the TRAP linearization algorithm, select LVLTIM = 1 or 3. Using LVLTIM = 1 (DVDT option) is the default, and helps avoid internal timestep too small non-convergence.

The local truncation algorithm (LVLTIM = 2) provides a higher degree of accuracy than the TRAP method. If you use this option, errors do not propagate from time point to time point, which can result in an unstable solution.

SEE ALSO:

.OPTION DVDT

.OPTION MAXAMP

SYNTAX:

.OPTION MAXAMP=x

DESCRIPTION:

Sets the maximum current, through voltage-defined branches (voltage sources and inductors). If the current exceeds the MAXAMP value, HSPICE reports an error. The default is 0.0.

.OPTION MAXORD

SYNTAX:

.OPTION MAXORD=x

DESCRIPTION:

Maximum order of integration, for the GEAR method in HSPICE. The x value can be either 1 or 2.

- MAXORD = 1 uses the backward Euler integration method.
- MAXORD = 2 (default) is more stable, accurate, and practical.

SEE ALSO:

.OPTION METHOD

.OPTION MBYPASS

SYNTAX:

.OPTION MBYPASS=x

DESCRIPTION:

Computes the default value of the BYTOL control option:

BYTOL = MBYPASSxVNTOL

Also multiplies the RELV voltage tolerance. Set MBYPASS to about 0.1, for precision analog circuits.

- Default is 1, for DVDT = 0, 1, 2, or 3.
- Default is 2, for DVDT = 4.

SEE ALSO:

.OPTION DVDT
.OPTION RELV

.OPTION MEASDGT

SYNTAX:

.OPTION MEASDGT=x

DESCRIPTION:

Formats the **.MEASURE** statement output, in both the listing file and the **.MEASURE** output files (*.ma0*, *.mt0*, *.ms0*, and so on).

The value of x is typically between 1 and 7, although you can set it as high as 10. The default is 4.0.

For example, if MEASDGT = 5, then **.MEASURE** displays numbers as:

- Five decimal digits, for numbers in scientific notation.
- Five digits to the right of the decimal, for numbers between 0.1 and 999.

In the listing (.lis), file, all **.MEASURE** output values are in scientific notation, so **.OPTION** MEASDGT = 5 results in five decimal digits.

Use MEASDGT with **.OPTION** INGOLD = x to control the output data format.

SEE ALSO:

.OPTION INGOLD
.MEASURE

.OPTION MEASFAIL

SYNTAX:

.OPTION MEASFAIL=0|1

DESCRIPTION:

You can assign the **.OPTION** MEASFAIL option the following values:

- If MEASFAIL=0, outputs 0 into the *.mt#*, *.ms#*, or *.ma#* file, and prints failed to the listing file.
- If MEASFAIL=1 (default), prints failed into the *.mt#*, *.ms#*, or *.ma#* file, and into the listing file: **.OPTION** MEASFAIL=1 | 0

.OPTION MEASSORT

SYNTAX:

.OPTION MEASSORT=x

DESCRIPTION:

In versions of HSPICE before 2003.09, to automatically sort large numbers of **.MEASURE** statements, you could use the **.OPTION** MEASSORT statement.

- **.OPTION** MEASSORT=0 (default; did not sort **.MEASURE** statements).
- **.OPTION** MEASSORT=1 (internally sorted **.MEASURE** statements).

You needed to set this option to 1 only if you used a large number of **.MEASURE** statements, where you needed to list similar variables together (to reduce simulation time). For a small number of **.MEASURE** statements, turning on internal sorting sometimes slowed-down simulation while sorting, compared to not sorting first.

Starting in version 2003.09, this option is obsolete. Now the measure performance is order independent, and HSPICE ignores this option.

SEE ALSO:

.MEASURE

.OPTION MEASOUT

SYNTAX:

.OPTION MEASOUT=x

DESCRIPTION:

This option outputs **.MEASURE** statement values and sweep parameters into an ASCII file. Post-analysis processing (AvanWaves or other analysis tools) uses this <*design*>*.mt*# file, where # increments for each **.TEMP** or **.ALTER** block.

For example, for a parameter sweep of an output load, which measures the delay, the *.mt*# file contains data for a delay-versus-fanout plot. The default is 1. You can set this option to 0 (off) in the *hspice.ini* file.

SEE ALSO:

.ALTER
.TEMP

.OPTION MENTOR

SYNTAX:

.OPTION MENTOR=x

DESCRIPTION:

MENTOR = 2 enables the Mentor MSPICE-compatible (ASCII) interface. This option requires a specific license.

.OPTION METHOD

SYNTAX:

.OPTION METHOD=GEAR|TRAP

DESCRIPTION:

Sets the numerical integration method, for a transient analysis, to either GEAR or TRAP.

- To use GEAR, set METHOD = GEAR, which sets LVLTIM = 2.
- To change LVLTIM from 2 to 1 or 3, set LVLTIM = 1 or 3, after the METHOD = GEAR option. This overrides METHOD=GEAR, which sets LVLTIM = 2.

TRAP (trapezoidal) integration usually reduces program execution time, with more accurate results. However, this method can introduce an apparent oscillation on printed or plotted nodes, which might not result from circuit behavior. To test this, run a transient analysis, using a small timestep. If oscillation disappears, the cause was the trapezoidal method.

The GEAR method is a filter, removing oscillations that occur in the trapezoidal method. Highly non-linear circuits (such as operational amplifiers) can require very long execution times, when you use the GEAR method.

Circuits that do not converge in trapezoidal integration, often converge if you use GEAR. Default is TRAP (trapezoidal).

Gear algorithm:

OPTION METHOD = GEAR

Backward-Euler:

OPTION METHOD = GEAR MU = *0*

Trapezoidal algorithm (default):

OPTION METHOD = TRAP

SEE ALSO:

.OPTION LVLTIM
.OPTION MU

.OPTION MODMONTE

SYNTAX:

.OPTION MODMONTE=x

DESCRIPTION:

If MODMONTE=1, then within a single simulation run, each device that shares the same model card and is in the same Monte Carlo index receives a different random value for parameters that have a Monte Carlo definition.

If MODMONTE=0 (default), then within a single simulation run, each device that shares the same model card and is in the same Monte Carlo index, receives the same random value for its parameters that have a Monte Carlo definition.

.OPTION MODSRH

SYNTAX:

.OPTION MODSRH=x

EXAMPLE:

```
example.sp:
* modsrh used incorrectly
.option post modsrh=1
xi1 net8 b c t6
xi0 a b net8 t6
v1 a 0 pulse 3.3 0.0 10E-6 1E-9 1E-9
+ 25E-6 50E-6
v2 b 0 2
v3 c 0 3
.model nch nmos level=49 version=3.2
.end
```

This input file automatically searches for t6.inc. If t6.inc includes the nch model, and you set MODSRH to 1, HSPICE does not load nch. Do not set MODSRH=1 in this type of file call. Use this option in front of the **.MODEL** card definition.

DESCRIPTION:

If MODSRH=1, HSPICE does not load or reference a model described in a **.MODEL** statement, if the netlist does not use that model. This option shortens simulation run time, when the netlist references many models, but no element in the netlist calls those models. The default is MODSRH=0. If MODSRH=1, then the read-in time increases slightly.

SEE ALSO:

.MODEL

.OPTION MONTECON

SYNTAX:

.OPTION MONTECON=x

DESCRIPTION:

Continues a Monte Carlo analysis in HSPICE. Retrieves the next random value, even if non-convergence occurs. A random value can be too large, or too small, to cause convergence to fail. Other types of analysis can use this Monte Carlo random value.

.OPTION MU

SYNTAX:

.OPTION MU=x

DESCRIPTION:

This option defines the coefficient for trapezoidal integration. The value range is 0.0 to 0.5, and the default is 0.5.

.OPTION NEWTOL

SYNTAX:

.OPTION NEWTOL=x

DESCRIPTION:

Calculates one or more iterations past convergence, for every calculated DC solution and timepoint circuit solution. If you do not set NEWTOL, after HSPICE determines convergence, the convergence routine ends, and the next program step begins. The default is 0.

.OPTION NODE

SYNTAX:

.OPTION NODE=x

EXAMPLE:

```
1 M1:B D2:+ Q4:B
```

This sample part of a cross reference line indicates that the bulk of M1, the anode of D2, and the base of Q4, all connect to node 1.

DESCRIPTION:

Prints a node cross reference table. The BRIEF option suppresses NODE. The table lists each node and all elements connected to it. A code indicates the terminal of each element. A colon (`:`) separates the code from the element name.

The codes are:

+	Diode anode
-	Diode cathode
B	BJT base
B	MOSFET or JFET bulk
C	BJT collector
D	MOSFET or JFET drain
E	BJT emitter
G	MOSFET or JFET gate
S	BJT substrate
S	MOSFET or JFET source

SEE ALSO:

.OPTION BRIEF

.OPTION NOELCK

SYNTAX:

.OPTION NOELCK

DESCRIPTION:

No element check; bypasses element checking, to reduce pre-processing time for very large files.

.OPTION NOMOD

SYNTAX:

.OPTION NOMOD

DESCRIPTION:

Suppresses the printout of model parameters.

.OPTION NOPAGE

SYNTAX:

.OPTION NOPAGE

DESCRIPTION:

Suppresses page ejects for title headings.

.OPTION NOPIV

SYNTAX:

.OPTION NOPIV=x

DESCRIPTION:

Prevents HSPICE from automatically switching to pivoting matrix factors, if a nodal conductance is less than PIVTOL. NOPIV inhibits pivoting.

SEE ALSO:

.OPTION PIVTOL

.OPTION NOTOP

SYNTAX:

.OPTION NOTOP

DESCRIPTION:

Suppresses topology checks to increase the speed for pre-processing very large files.

.OPTION NOWARN

SYNTAX:

.OPTION NOWARN

DESCRIPTION:

Suppresses all warning messages, except those generated from statements in **.ALTER** blocks.

SEE ALSO:

.ALTER

.OPTION NUMDGT

SYNTAX:

.OPTION NUMDGT=x

DESCRIPTION:

Number of significant digits to print, for output variable values. The value of *x* is typically between 1 and 7, although you can set it as high as 10. Default is `4.0`. This option does not affect the accuracy of the simulation.

.OPTION NXX

SYNTAX:

.OPTION NXX

DESCRIPTION:

Stops printback of the data file, until HSPICE finds an **.OPTION** BRIEF = 0, or the .END statement. It also resets the LIST, NODE, and OPTS options, and sets NOMOD. BRIEF = 0 enables printback. NXX is the same as BRIEF.

SEE ALSO:

.OPTION BRIEF

.OPTION OFF

SYNTAX:

.OPTION CSDF=x

DESCRIPTION:

For all active devices, initializes terminal voltages to zero, if you did not initialize them to other values. For example, if you did not initialize both drain and source nodes of a transistor (using **.NODESET** or **.IC** statements, or connecting them to sources), then OFF initializes all nodes of the transistor to zero.

HSPICE checks the OFF option, before element IC parameters. If you assigned an element IC parameter to a node, simulation initializes the node to the element IC parameter value, even if the OFF option previously set it o zero.

You can use the OFF element parameter to initialize terminal voltages to zero, for specific active devices. Use the OFF option to help find exact DC operating-point solutions, for large circuits.

SEE ALSO:

.IC
.NODESET

.OPTION OPTLST

SYNTAX:

.OPTION OPTLIST

DESCRIPTION:

Outputs additional optimization information:

- No information (default).
- Prints parameter, Broyden update, and bisection results information.
- Prints gradient, error, Hessian, and iteration information.

Prints all of the above, and Jacobian.

.OPTION OPTS

SYNTAX:

.OPTION OPTS

DESCRIPTION:

Prints the current settings, for all control options. If you change any of the default values of the options, the OPTS option prints the values that the simulation actually uses. The BRIEF option suppresses OPTS.

SEE ALSO:

.OPTION BRIEF

.OPTION PARHIER

SYNTAX:

.OPTION PARHIER = < GLOBAL | LOCAL >

EXAMPLE:

```
.OPTION parhier=<global | local>
.PARAM DefPwid = 1u
.SUBCKT Inv a y DefPwid = 2u DefNwid = 1u
  Mp1 <MosPinList> pMosMod L = 1.2u W = DefPwid
  Mn1 <MosPinList> nMosMod L = 1.2u W = DefNwid
.ENDS
```

This example explicitly shows the difference between local and global scoping, for using parameters in sub-circuits.

DESCRIPTION:

Use the **.OPTION** PARHIER parameter to specify scoping rules.

The default setting is GLOBAL.

.OPTION PATHNUM

SYNTAX:

.OPTION PATHNUM

DESCRIPTION:

Prints subcircuit path numbers, instead of path names.

.OPTION PIVOT

SYNTAX:

.OPTION PIVOT=x

DESCRIPTION:

Selects a pivot algorithms. Use these algorithms to reduce simulation time, and to achieve convergence in circuits that produce hard-to-solve matrix equations. To select the pivot algorithm, set PIVOT to one of these values:

- 0: Original non-pivoting algorithm.
- 1: Original pivoting algorithm.
- 2: Picks the largest pivot in the row.
- 3: Picks the best pivot in a row.
- 10 (default): Fast, non-pivoting algorithm; requires more memory.
- 11: Fast, pivoting algorithm; requires more memory than PIVOT values less than 11.
- 12: Picks the largest pivot in the row; requires more memory than PIVOT values less than 12.
- 13: Fast, best pivot: faster; requires more memory than PIVOT values less than 13.

The fastest algorithm is PIVOT = 13, which can improve simulation time up to ten times, on very large circuits. However, PIVOT = 13 requires substantially more memory for simulation.

Some circuits with large conductance ratios, such as switching regulator circuits, might require pivoting.

If PIVTOL = 0, HSPICE automatically changes from non-pivoting, to a row-pivot strategy, if it detects any diagonal-matrix entry less than PIVTOL. This strategy provides the time and memory advantages of non-pivoting inversion, and avoids unstable simulations and incorrect results.
Use .OPTION NOPIV, to prevent HSPICE from pivoting. For very large circuits, PIVOT = 10, 11, 12, or 13, can require excessive memory.

If HSPICE switches to pivoting during a simulation, it displays the message followed by the node numbers that cause the problem:

```
pivot change on the fly
```

Use **.OPTION** NODE to cross-reference a node to an element. The SPARSE option is the same as PIVOT.

.OPTION PIVREF

SYNTAX:

.OPTION PIVREF=x

DESCRIPTION:

Pivot reference. Use PIVREF in PIVOT = 11, 12, or 13, to limit the size of the matrix. Default is 1e+8.

SEE ALSO:

.OPTION PIVOT

.OPTION PIVREL

SYNTAX:

.OPTION PIVREL=x

DESCRIPTION:

Sets the maximum and minimum ratio of a row or matrix. Use only if PIVOT = 1. Large values for PIVREL can result in very long matrix pivot times. If the value is too small, however, no pivoting occurs. Start with small values of PIVREL, using an adequate (but not excessive) value, for convergence and accuracy. The default is 1E-20 (max = 1e-20, min = 1).

SEE ALSO:

.OPTION PIVOT

.OPTION PIVTOL

SYNTAX:

.OPTION PIVTOL=x

DESCRIPTION:

Absolute minimum value for which HSPICE accepts a matrix entry as a pivot. If PIVOT=0, PIVTOL is the minimum conductance in the matrix. Default=1.0e-15.

PIVTOL must be less than GMIN or GMINDC. Values that approach 1 increase the pivot.

SEE ALSO:

.OPTION GMIN
.OPTION GMINDC
.OPTION PIVOT

.OPTION PLIM

SYNTAX:

.OPTION PLIM

DESCRIPTION:

Specifies plot size limits, for current and voltage plots:

- Finds a common plot limit, and plots all variables on one graph, at the same scale
- Enables SPICE-type plots in HSPICE, which create a separate scale and axis for each plot variable.

This option does not affect post- processing of graph data.

.OPTION POST

SYNTAX:

.OPTION POST=[0|1|2|3|ASCII|BINARY]

EXAMPLE:

```
.OPTION POST=2
```

DESCRIPTION:

Use an **.OPTION** POST statement to display high-resolution AvanWaves plots of simulation results, on either a graphics terminal or a high-resolution laser printer. Use **.OPTION** POST to provide output, without specifying other parameters. POST has defaults, which supply usable data to most parameters.

Value	Description
POST = 0	Does not output simulation results.
POST = 1, BINARY	Output format is binary.
POST = 2, ASCII	Output format is ASCII.
POST = 3	Output format is New Wave binary.

.OPTION POST_VERSION

SYNTAX:

.OPTION POST_VERSION=x

DESCRIPTION:

Sets the post-processing output version:

- x = 9007 truncates the node name in the post-processor output file, to a maximum of 16 characters.
- x = 9601 sets the node name length for the output file, consistent with the input restrictions (1024 characters).

If you set **.OPTION** POST_VERSION=2001, then the post-processing output version is switched 2001. This option shows you the new output file header, which includes the right number of output variables, rather than **** when the number exceeds 9999. If you set **.OPTION** POST_VERSION=2001 post=2 in the netlist, then HSPICE returns more-accurate ASCII results.

```
.option post_version=2001
```

To use binary values (with double precision) in the output file, include the following in the input file:

```
*******************************************
.option post (or post=1) post_version=2001
*******************************************
```

For more accurate simulation results, comment this format.

.OPTION PROBE

SYNTAX:

.OPTION PROBE=x

DESCRIPTION:

Limits post-analysis output to only variables specified in **.PROBE**, **.PRINT**, **.PLOT**, and **.GRAPH** statements. By default, HSPICE outputs all voltages and power supply currents, in addition to variables listed in **.PROBE/.PRINT/.PLOT/.GRAPH** statements. PROBE significantly decreases the size of simulation output files.

SEE ALSO:

.GRAPH
.PLOT
.PRINT
.PROBE

.OPTION PSF

SYNTAX:

.OPTION PSF=x

DESCRIPTION:

Specifies whether HSPICE outputs binary or ASCII data, when you run an HSPICE simulation from Cadence Analog Artist. Supported on Sun Solaris 2.5/2.7/2.8, HPUX 10.20 and 11.20, IBM AIX 4.3 platforms only. This option is not available on the PC RedHat/Linux 7.2 platform.

The value of *x* can be 1 or 2.

- If x is 2, HSPICE produces ASCII output.
- If **.OPTION** ARTIST PSF = 1, HSPICE produces binary output.

SEE ALSO:

.OPTION ARTIST

.OPTION PURETP

SYNTAX:

.OPTION PURETP=x

DESCRIPTION:

Integration method to use, for reversal time point. The default is 0. If you set puretp=1, then if HSPICE finds non-convergence, it uses TRAP (instead of B.E) for the reversed time point. Use this option, with the method=TRAP statement, to help some oscillating circuits to oscillate, if the default simulation process cannot satisfy the result.

.OPTION PUTMEAS

SYNTAX:

.OPTION PUTMEAS=0|1

DESCRIPTION:

The **.OPTION** PUTMEAS option controls the output variables, listed in the **.MEASURE** statement.

- 0: Does not save variable values, which are listed in the **.MEASURE** statement, into the corresponding output file (such as *.tr#*, *.ac#* or *.sw#*). This option decreases the size of the output file.
- 1: Default. Saves variable values, which are listed in the **.MEASURE** statement, into the corresponding output file (such as *.tr#*, *.ac#* or *.sw#*). This option is similar to the output of HSPICE 2000.4.

SEE ALSO:

.MEASURE

.OPTION RELH

SYNTAX:

.OPTION RELH=x

DESCRIPTION:

Relative current tolerance, through voltage-defined branches (voltage sources and inductors). Use RELH to check current convergence, but only if the value of the ABSH control option is greater than zero. Default is 0.05.

SEE ALSO:

.OPTION ABSH

.OPTION RELI

SYNTAX:

.OPTION RELI=x

DESCRIPTION:

Sets the relative error/tolerance change, from iteration to iteration. This parameter determines convergence for all currents, in diode, BJT, and JFET devices. (RELMOS sets tolerance for MOSFETs). This is the change in current, from the value calculated at the previous timepoint.

- Default = 0.01 for KCLTEST = 0.
- Default = 1e-6 for KCLTEST = 1.

SEE ALSO:

.OPTION RELMOS
.OPTION KCLTEST

.OPTION RELMOS

SYNTAX:

.OPTION RELMOS=x

DESCRIPTION:

Sets the relative error tolerance (percent) for drain-to-source current, from iteration-to-iteration. This parameter determines convergence for currents in MOSFET devices. (RELI sets the tolerance for other active devices.) Sets the change in current, from the value calculated at the previous timepoint. HSPICE uses the RELMOS value, only if the current is greater than the ABSMOS floor value. The default is 0.05.

SEE ALSO:

.OPTION ABSMOS

.OPTION RELQ

SYNTAX:

.OPTION RELQ=x

DESCRIPTION:

Used in the timestep algorithm for local truncation error (LVLTIM = 2). RELQ changes the timestep size. If the capacitor charge calculation (in the present iteration) exceeds that of the past iteration by a percentage greater than the RELQ value, then HSPICE reduces the internal timestep (Delta). The default=0.01.

SEE ALSO:

.OPTION LVLTIM

.OPTION RELTOL

SYNTAX:

.OPTION RELTOL=x

DESCRIPTION:

Relative error tolerance for voltages. Use RELTOL, with the ABSV control option, to determine voltage convergence. Increasing RELTOL increases the relative error. RELTOL is the same as RELV. RELI and RELVDC options default to the RELTOL value. The default is 1e-3.

SEE ALSO:

.OPTION ABSV
.OPTION RELI
.OPTION RELV
.OPTION RELVDC

.OPTION RELV

SYNTAX:

.OPTION RELV=x

DESCRIPTION:

Sets the relative error tolerance for voltages. If voltage or current exceeds the absolute tolerances, a RELV test determines convergence. Increasing RELV increases the relative error. You should generally maintain RELV at its default value. RELV conserves simulator charge. For voltages, RELV is the same as RELTOL. The default is 1e-3.

SEE ALSO:

.OPTION RELTOL

.OPTION RELVAR

SYNTAX:

.OPTION RELVAR=x

DESCRIPTION:

Use this option with ABSVAR, and the DVDT timestep algorithm. RELVAR sets the relative voltage change, for LVLTIM = 1 or 3. If the node voltage at the current time point exceeds the node voltage at the previous time point by RELVAR, then HSPICE reduces the timestep, and calculates a new solution at a new time point. The default is 0.30 (30%).

SEE ALSO:

.OPTION ABSVAR
.OPTION DVDT
.OPTION LVLTIM

.OPTION RELVDC

SYNTAX:

.OPTION RELVDC=x

DESCRIPTION:

Sets the relative error tolerance for voltages. If voltages or currents exceed their absolute tolerances, the RELVDC test determines convergence. Increasing RELVDC increases the relative error. You should generally maintain RELVDC at its default value. RELVDC conserves simulator charge. Default is RELTOL (RELTOL default = 1e-3).

SEE ALSO:

.OPTION RELTOL

.OPTION RESMIN

SYNTAX:

.OPTION RESMIN=x

DESCRIPTION:

Minimum resistance for all resistors, including parasitic and inductive resistances. The default is 1e-5 (ohm), and the range is 1e-15 to 10 ohm.

.OPTION RISETIME

SYNTAX:

.OPTION RISETIME=x

DESCRIPTION:

Smallest risetime of a signal. Use this option only in transmission line models. In the U Element, this equation determines the number of lumps:

$$\mathrm{MIN}\left[20, 1 + \left(\frac{\mathrm{TDeff}}{\mathrm{RISETIME}}\right) \cdot 20\right]$$

TDeff is the end-to-end delay in a transmission line. The W Element uses RISETIME, only if Rs or Gd is non-zero. In such cases, RISETIME determines the maximum signal frequency.

.OPTION RMAX

SYNTAX:

.OPTION RMAX=x

DESCRIPTION:

Sets the TSTEP multiplier, which controls the maximum value (DELMAX) for the Delta of the internal timestep:

DELMAX = TSTEP x RMAX

- Default = 5, if DVDT = 4 and LVLTIM = 1.
- Otherwise, the default = 2.

The maximum value is 1e+9, the minimum value is 1e-9. The recommended maximum value is 1e+5.

SEE ALSO:

.OPTION DVDT
.OPTION LVLTIM

.OPTION RMIN

SYNTAX:

.OPTION RMIN=x

DESCRIPTION:

Sets the minimum value of Delta (internal timestep). An internal timestep smaller than RMINxTSTEP, terminates the transient analysis, and reports an internal timestep too small error. If the circuit does not converge in IMAX iterations, Delta decreases by the amount you set in the FT option. The default = 1.0e-9.

SEE ALSO:

.OPTION FT
.OPTION IMAX

.OPTION SCALE

SYNTAX:

.OPTION SCALE=x

DESCRIPTION:

Element scaling factor, in HSPICE. Scales parameters in element cards, by their value. Default=1.

.OPTION SCALM

SYNTAX:

.OPTION SCALM=x

DESCRIPTION:

Model scaling factor, in HSPICE. Scales model parameters by their value. The default is 1. See the *HSPICE Elements and Device Models Manual*, for parameters this option scales.

.OPTION SDA

SYNTAX:

.OPTION SDA=x

DESCRIPTION:

SDA = 2 produces a Cadence WSF (ASCII format) post-analysis file, for Opus™. This option requires a specific license. The SDA is the same as the CDS option.

.OPTION SEARCH

SYNTAX:

.OPTION SEARCH = '*directory_path*'

EXAMPLE:

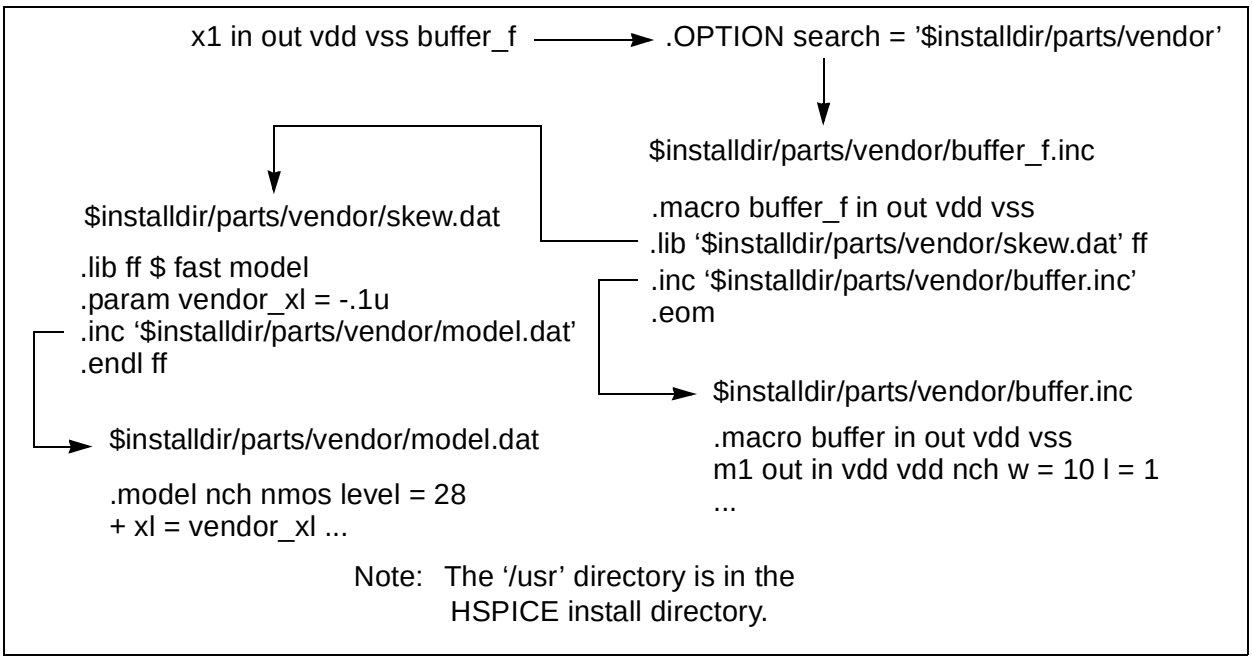
```
.OPTION SEARCH = '$installdir/parts/vendor'
```

DESCRIPTION:

Use the **.OPTION** SEARCH statement to automatically access a library.

This example searches for models in the *vendor* subdirectory, under the <`$installdir>/parts` installation directory (see Figure 3-1). The *parts/* directory contains the DDL subdirectories.

Figure 3-1 Vendor Library Usage



.OPTION SEED

SYNTAX:

.OPTION SEED=x

DESCRIPTION:

Starting seed for random-number generator in HSPICE Monte Carlo analysis. The minimum value is 1; the maximum value is 259200.

.OPTION SLOPETOL

SYNTAX:

.OPTION SLOPETOL=x

DESCRIPTION:

Minimum value, for breakpoint table entries in a piecewise linear (PWL) analysis. If the difference in the slopes of two consecutive PWL segments is less than the SLOPETOL value, HSPICE ignores the breakpoint, for the point between the segments. The default is 0.5.

.OPTION SPARSE

SYNTAX:

.OPTION SPARSE=x

DESCRIPTION:

The SPARSE option is the same as PIVOT.

SEE ALSO:

.OPTION PIVOT

.OPTION SPICE

SYNTAX:

.OPTION SPICE=x

EXAMPLE 1:

Example of general parameters, used with **.OPTION** SPICE:

```
TNOM = 27 DEFNRD = 1 DEFNRS = 1 INGOLD = 2
ACOUT = 0 DC
PIVOT PIVTOL = IE-13 PIVREL = 1E-3
RELTOL = 1E-3
ITL1 = 100
ABSMOS = 1E-6 RELMOS = 1E-3 ABSTOL = 1E-12
VNTOL = 1E-6
ABSVDC = 1E-6 RELVDC = 1E-3 RELI = 1E-3
```

EXAMPLE 2:

Example of transient parameters, used with **.OPTION** SPICE:

```
DCAP = 1 RELQ = 1E-3 CHGTOL-1E-14 ITL3 = 4
ITL4 = 10
ITL5 = 5000 FS = 0.125 FT = 0.125
```

EXAMPLE 3:

Example of model parameters, used with **.OPTION** SPICE:

```
For BJT: MJS = 0
For MOSFET, CAPOP = 0
LD = 0 if not user-specifiedUTRA = 0 not used by
SPICE for LEVEL = 2NSUB must be specifiedNLEV = 0
for SPICE noise equation
```

DESCRIPTION:

Makes HSPICE compatible with Berkeley SPICE. If you set this option, HSPICE uses the options and model parameters explained in the examples.

.OPTION STATFL

SYNTAX:

.OPTION STATFL=x

DESCRIPTION:

Controls whether HSPICE creates a *.st0* file.

- STATFL=0 (default) outputs a *.st0* file.
- STATFL=1 suppresses the *.st0* file.

.OPTION TIMERES

SYNTAX:

.OPTION TIMERES=x

DESCRIPTION:

Minimum separation between breakpoint values, for the breakpoint table. If two breakpoints are closer together (in time) than the TIMERES value, HSPICE enters only one of them in the breakpoint table. The default is 1 ps.

.OPTION TNOM

SYNTAX:

.OPTION TNOM=x

DESCRIPTION:

Reference temperature for HSPICE simulation. At this temperature, component derating is zero. The default is 25 degrees Celsius. If you enable **.OPTION** SPICE, the default is 27 degrees Celsius.

SEE ALSO:

.OPTION SPICE

.OPTION TRCON

SYNTAX:

.OPTION TRCON=x

DESCRIPTION:

Controls the automatic convergence (autoconvergence) and automatic speedup (autospeedup) processes in HSPICE. HSPICE also uses autoconvergence in DC analysis, if the Newton-Raphson (N-R) method fails to converge.

For some large non-linear circuits with large TSTOP/TSTEP values, analysis might run for an excessively long time. In this case, HSPICE might automatically set a new and bigger RMAX value, to speed up the analysis for primary reference. In most cases, however, HSPICE does not activate this type of autospeedup process.

For autospeedup to occur, all three of the following conditions must occur:

- N1 (Number of Nodes) > 1,000
- N2 (TSTOP/TSTEP) >= 10,000
- N3 (Total Number of Diode, BJTs, JFETs and MOSFETs) > 300

Autospeedup is most likely to occur if the circuit also meets either of the following conditions:

- N2 >= 1e+8, and N3 > 500, or
- N2 >= 2e+5, and N3 > 1e+4

If HSPICE does activate autospeedup, you might need to disable it. To do this, set TRCON=-1, and increase TSTEP or RMAX (or both), to balance accuracy and speed.

You can assign the following values to the TRCON option:

- TRCON=1 (the default) enables both autoconvergence and autospeedup.
- TRCON= 0 enables autospeedup only.
- TRCON =-1 disables both autoconvergence and autospeedup.

If the circuit fails to converge using the trapezoidal (TRAP) numerical integration method (for example, because of trapezoidal oscillation), HSPICE uses the GEAR method and LTE timestep algorithm, to run the transient analysis again from time=0. This process is called autoconvergence.

Autoconvergence sets options to their default values before the second try:

```
METHOD=GEAR, LVLTIM=2, MBYPASS=1.0,
BYPASS=0.0, SLOPETOL=0.5, BYTOL=
min{mbypas*vntol and reltol}
```

RMAX=2.0 if it was 5.0 in the first run; otherwise RMAX does not change.

SEE ALSO:

.OPTION BYPASS
.OPTION BYTOL
.OPTION MBYPASS
.OPTION RMAX
.OPTION SLOPETOL

.OPTION TRTOL

SYNTAX:

.OPTION TRTOL=x

DESCRIPTION:

Used in the timestep algorithm for local truncation error (LVLTIM = 2). HSPICE multiplies TRTOL by the internal timestep, which the timestep algorithm for the local truncation error generates. TRTOL reduces simulation time, and maintains accuracy. It estimates the amount of error introduced, when the algorithm truncates the Taylor series expansion. This error reflects the minimum time-step, to reduce simulation time and maintain accuracy. The range of TRTOL is 0.01 to 100; typical values are 1 to 10. If you set TRTOL to 1 (the minimum value), HSPICE uses a very small timestep. As you increase the TRTOL setting, the timestep size increases. The default is 7.0.

SEE ALSO:

.OPTION LVLTIM

.OPTION UNWRAP

SYNTAX:

.OPTION UNWRAP

DESCRIPTION:

Displays phase results for AC analysis, in unwrapped form (with a continuous phase plot).HSPICE uses these results to accurately calculate group delay. It also uses unwrapped phase results to compute group delay, even if you do not set UNWRAP.

.OPTION VERIFY

SYNTAX:

.OPTION VERIFY

DESCRIPTION:

The VERIFY option is an alias for the LIST option.

SEE ALSO:

.OPTION LIST

.OPTION VFLOOR

SYNTAX:

.OPTION VFLOOR=x

DESCRIPTION:

Minimum voltage to print in output listing. All voltages lower than VFLOOR, print as 0. Affects only the output listing: VNTOL (ABSV) sets minimum voltage to use in a simulation.

SEE ALSO:

.OPTION VNTOL

.OPTION VNTOL

SYNTAX:

.OPTION VNTOL=x

DESCRIPTION:

The VNTOL option is the same as the ABSV option.

SEE ALSO:

.OPTION ABSV

.OPTION WARNLIMIT

SYNTAX:

.OPTION WARNLIMIT=x

DESCRIPTION:

Limits how many times certain warnings appear in the output listing. This reduces the output listing file size. *x* is the maximum number of warnings for each warning type. This limit applies to the following warning messages:

- MOSFET has negative conductance.
- Node conductance is zero.
- Saturation current is too small.
- Inductance or capacitance is too large.

The default is 1.

.OPTION WL

SYNTAX:

.OPTION WL=x

DESCRIPTION:

Reverses the specified order, in the VSIZE MOS element. Default order is length-width; changes the order to width-length. The default is 0.

.OPTION XDTEMP

SYNTAX:

.OPTION XDTEMP=*value*

EXAMPLE:

```
.OPTION XDTEMP
X1 2 0 SUB1 DTEMP=2
.SUBCKT SUB1 A B
R1 A B 1K DTEMP=3
C1 A B 1P
X2 A B sub2 DTEMP=4
.ENDS

.SUBCKT SUB2 A B
R2 A B 1K
.ENDS
```

In this example:

1. X1 sets a temperature difference (2 degrees Celsius) between the elements within the subcircuit SUB1.
2. X2 (a subcircuit instance of X1) sets a temperature difference by the DTEMP value of both X1 and X2 (2+4=6 degrees Celsius) between the elements within the SUB2 subcircuit. Finally, the DTEMP value of each element in this example is:

```
Elements DTEMP Value (Celsius)
X1 2
X1.R1 2+3 =5
X1.C1 2
X2 2+4=6
X2.R2 6
```

DESCRIPTION:

The **.OPTION** XDTEMP statement defines how HSPICE interprets the DTEMP parameter, where value is either:

- 0 (the default), indicating a user-defined parameter, or
- 1 indicates a temperature difference parameter.

If you set **.OPTION** XDTEMP to 1, HSPICE adds the DTEMP value in the subcircuit call statement to all elements within the subcircuit, that use the DTEMP keyword syntax.

The DTEMP parameter is cumulative throughout the design hierarchy.

.OPTION ZUKEN

SYNTAX:

.OPTION ZUKEN=x

DESCRIPTION:

This option enables or disables the Zuken interface.

- If x is 2, enables the Zuken interactive interface.
- If x is 1 (default), disables this interface.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z