Chapter 4

Specifying Simulation Output

Use output format statements and variables to display steady state, frequency, and time domain simulation results. These variables also permit you to use behavioral circuit analysis, modeling, and simulation techniques. Display electrical specifications such as rise time, slew rate, amplifier gain, and current density using the output format features.

This chapter discusses the following topics:

- Using Output Statements
- Selecting Simulation Parameters
- Displaying Simulation Results

Using Output Statements

Star-Hspice output statements are contained in the input netlist file and include .PRINT, .PLOT, .GRAPH, .PROBE, and .MEASURE. Each statement specifies the output variables and type of simulation result to be displayed—for example, .DC, .AC, or .TRAN. The .PRINT statement prints numeric analysis results. The .PLOT statement generates low resolution printer plots in the output listing file. The .GRAPH statement generates high resolution plots for supported devices such as HP LaserJet and PostScript printers without using AvanWaves. The .PROBE statement (together with .OPTION PROBE) allows output variables to be saved in all the interface files with no additional output in the listing file. The .MEASURE statement prints numeric results of measured electrical specifications for specific analyses. All output variables referenced in .PRINT, .PLOT, .GRAPH, .PROBE, and .MEASURE statements are put into the interface files for AvanWaves. AvanWaves allows high resolution, post simulation, and interactive terminal or printer display of waveforms.

Output Files

Star-Hspice produces various types of output files, as listed in the following table.

Table 4-1: Star-Hspice Output Files and Suffixes

Output File Type	Extension
Output listing	*.lis*, or user-specified
Transient analysis results	*.tr#* †
Transient analysis measurement results	*.mt#*
DC analysis results	*.sw#* †
DC analysis measurement results	*.ms#*
AC analysis results	*.ac#* †
AC analysis measurement results	*.ma#*

Table 4-1: Star-Hspice Output Files and Suffixes

Output File Type	Extension
Hardcopy graph data (from *meta.cfg* PRTDEFAULT)	*.gr#* ††
Digital output	*.a2d*
FFT analysis graph data	*.ft#*†††
Subcircuit cross-listing	*.pa#*
Output status	*.st#*
Operating point node voltages (initial conditions)	*.ic*

is either a sweep number or a hardcopy file number.
† Only created if a .POST statement is used to generate graphical data.
†† Requires a .GRAPH statement or a pointer to a file exists in the meta.cfg file. This file is not generated by the PC version of Star-Hspice.
††† Only created if a .FFT statement is used.

The files listed in Table 4-1: are described below.

Output listing can appear as *output_file* (no file extension), *output_file.lis*, or have a user-specified file extension, depending upon which format is used to start the simulation (see *Chapter 1, Introducing Star-Hspice*). *Output_file* is the output file specification, less extension. This file includes the following information:

- Name and version of simulator used
- Avant! message block
- Input file name
- User name
- License details
- Copy of the input netlist file
- Node count
- Operating point parameters
- Details of volt drop, current, and power for each source and subcircuit
- Low resolution plots originating from the .PLOT statement

- Results of .PRINT statement
- Results of .OPTIONS statements

Transient analysis results are placed in *output_file. tr#*, where # is specified as 0-9 or a-z following the -n argument. This file contains a list of transient analysis numerical results. It is the result of an input file .TRAN statement together with an .OPTION POST statement to create a post-analysis file. The output file is in proprietary binary format if POST = 0 or 1, or ASCII format if POST = 2.

Transient analysis measurement results are written to *output_file.mt#*. This output file is the result of an input file .MEASURE TRAN statement.

DC analysis results appear in *output_file.sw#*, which is produced as a result of a .DC statement. This file contains the results of the applied stepped or swept DC parameters defined in that statement. The results may include noise, distortion, or network analysis.

DC analysis measurement results are given in the file *output_file. ms#* when a .MEASURE DC statement exists in the input file.

AC analysis results are placed in *output_file.ac#*. These results contain a listing of output variables as a function of frequency, according to user specification following the .AC statement.

AC analysis measurement results appear in *output_file.ma#* when a .MEASURE AC statement exists in the input file.

Hardcopy graph data are placed in *output_file.gr#*, which is produced as a result of a .GRAPH statement. It is in the form of a printer file, typically in Adobe PostScript or HP PCL format. This facility is not available in the PC version of Star-Hspice.

Digital output contains data converted to digital form by the U element A2D conversion option.

FFT analysis graph data contains the graphical data needed to display the FFT analysis waveforms.

Subcircuit cross-listing is automatically generated and written into *output_file.pa#* when the input netlist includes subcircuits. It relates the subcircuit node names in subcircuit call statements to the node names used in the corresponding subcircuit definitions.

Output status is named with the output file specification, with a *.st#* extension, and contains the following runtime reports:

- Start and end times for each CPU phase
- Options settings with warnings for obsolete options
- Status of preprocessing checks for licensing, input syntax, models, and circuit topology
- Convergence strategies used by Star-Hspice on difficult circuits

The information in this file is useful in diagnosing problems, particularly when communicating with Avant! Customer Support.

Operating point node voltages are DC operating point initial conditions stored by the .SAVE statement.

Output Variables

The output format statements require special output variables to print or plot analysis results for nodal voltages and branch currents. There are five groups of output variables: DC and transient analysis, AC analysis, element template, .MEASURE statement, and parametric analysis.

DC and transient analysis displays individual nodal voltages, branch currents, and element power dissipation.

AC analysis displays imaginary and real components of a nodal voltage or branch current, as well as the phase of a nodal voltage or branch current. AC analysis results also print impedance parameters and input and output noise.

Element template analysis displays element-specific nodal voltages, branch currents, element parameters, and the derivatives of the element’s node voltage, current, or charge.

The .MEASURE statement variables are user-defined. They represent the electrical specifications measured in a .MEASURE statement analysis.

Parametric analysis variables are mathematically defined expressions operating on user-specified nodal voltages, branch currents, element template variables, or other parameters. You can perform behavioral analysis of simulation results using these variables. See “Using Algebraic Expressions” on page 10-4 for information about parameters in Star-Hspice.

For element or node output variables defined as parameters, if the parameter name is longer than 16 characters, Star-Hspice substitutes a 0 for the variable, issues a warning, and continues the analysis. The value of the result is 0. For example, in the following statement the parameter name “xptgref.xbug.mxi18” is replaced by 0 because it is longer than 16 characters. This results in a value of 0 for the result “ace”:

```
.MEASURE TRAN ace AVG
+ PAR(‘2*(il(xptgref.xbuf.mxi18))’) from 0 to 100
```

Star-Hspice prints a warning that a value of zero is used for a nonexistent output variable, since it does not recognize the long name as a valid output variable name.

.OPTION POST for High Resolution Graphics

Use an .OPTION POST statement to use AvanWaves to display high resolution plots of simulation results on a graphics terminal or a high resolution laser printer. Use the .OPTION POST to provide output without specifying other parameters. POST has defaults that supply most parameters with usable data.

.OPTION ACCT Summary of Job Statistics

A detailed accounting report is generated using the ACCT option:

where:

.OPTION ACCT   enables reporting

.OPTION ACCT=1 (default)
is the same as ACCT with no argument

.OPTION ACCT=2
enables reporting plus matrix statistic reporting

Example

The following output appears at the end of the output listing.

```
******  job statistics summary tnom=  25.000 temp=  25.000
# nodes = 15 # elements= 29 # real*8 mem avail/used=
333333/   13454
# diodes= 0 # bjts = 0 # jfets = 0 # mosfets = 24

    analysis         time      # points  tot. iter  conv.iter

    op point         0.24         1        11
    transient        5.45        161       265       103
rev=    1
    pass1            0.08
    readin           0.12
    errchk           0.05
    setup            0.04
    output           0.00
the following time statistics are already included in the
analysis time
    load             5.22
    solver           0.16
# external nodes =   15 # internal nodes =    0
# branch currents=    5 total matrix size=   20
    pivot based and non pivotting solution times
      non pivotting:  ----  decompose       0.08  solve 0.08
    matrix size(   109) = initial size(   105) + fill(     4)
    words copied=      111124
          total cpu time          6.02 seconds
              job started at  11:54:11  21-sep92
              job ended   at  11:54:36  21-sep92
```

The definitions for the items in the above listing follow:

# BJTS	Number of bipolar transistors in the circuit
# ELEMENTS	Total number of elements
# JFETS	Number of JFETs in the circuit
# MOSFETS	Number of MOSFETs in the circuit
# NODES	Total number of nodes

# POINTS	Number of transient points specified by the user on the .TRAN statement. JTRFLG is usually at least 50 unless the option DELMAX is set.
CONV.ITER	Number of points that the simulator needed to take in order to preserve the accuracy specified by the tolerances
DC	DC operating point analysis time and number of iterations required. The option ITL1 sets the maximum number of iterations.
ERRCHK	Part of the input processing
MEM +	Amount of workspace available and used for the simulation
AVAILUSED	Measured in 64-bit (8-byte) words
OUTPUT	Time required to process all prints and plots
LOAD	Constructs the matrix equation
SOLVER	Solves equations
PASS1	Part of the input processing
READIN	Specifies the input reader that takes the user data file and any additional library files, and generates an internal representation of the information
REV	Number of times the simulator had to cut time (reversals). This is a measure of difficulty.
SETUP	Constructs a sparse matrix pointer system
TOTAL JOB TIME	Total amount of CPU time required to process the simulation. This is not the length of actual (clock) time that was taken, and may differ slightly from run to run, even if the runs are identical.

The ratio of TOT.ITER to CONV.ITER is the best measure of simulator efficiency. The theoretical ratio is 2:1. In this example the ratio was 2.57:1. SPICE generally has a ratio of 3:1 to 7:1.

In transient, the ratio of CONV.ITER to # POINTS is the measure of the number of points evaluated to the number of points printed. If this ratio is greater than about 4, the convergence and timestep control tolerances might be too tight for the simulation.

Changing the File Descriptor Limit

A simulation that has a large number of .ALTER statements might fail due to the limit on the number of file descriptors. For example, for a Sun workstation, the default number of file descriptors is 64, and a design with more than 50 .ALTER statements is liable to fail with the following error message:

```
error could not open output spool file /tmp/tmp.nnn
a critical system resource is inaccessible or exhausted
```

To prevent this on a Sun workstation, enter the following operating system command before you start the simulation:

limit descriptors 128

For platforms other than Sun workstations, see your system administrator for help with increasing the number of files you can open concurrently.

Selecting Simulation Parameters

This section discusses how to define specific parameters so that the simulation provides the appropriate output. Define simulation parameters using the .OPTION and .MEASURE statements and specific variable element definitions.

DC and Transient Output Variables

Some types of output variables for DC and transient analysis are:

- Voltage differences between specified nodes (or one specified node and ground)
- Current output for an independent voltage source
- Current output for any element
- Element templates containing the values of user-input variables, state variables, element charges, capacitance currents, capacitances, and derivatives for the various types of devices

The codes you can use to specify the element templates for output are summarized in *Element Template Output, page -35 in , Specifying Simulation Output*.

Nodal Voltage Output

Syntax

```
V (n1<,n2>)
```

n1, n2 defines the nodes between which the voltage difference (n1-n2) is to be printed or plotted. When n2 is omitted, the voltage difference between n1 and ground (node 0) is given.

Current Output: Voltage Sources

Syntax

```
I (<Xzzz.Xyyy>.Vxxx)
```

Xyyy	subcircuit element name, used to call the subcircuit (if any) within which the independent voltage source is defined
Xzzz	subcircuit element name, used to call the subcircuit (if any) that defines Xyyy
Vxxx	independent voltage source element name

Examples

```
.PLOT TRAN I(VIN)
.PRINT DC I(X1.VSRC)
.PLOT DC I(XSUB.XSUBSUB.VY)
```

Current Output: Element Branches

Syntax

```
In (<Xzzz.Xyyy>.Wwww)
```

where:	specifies:
n	node position number in the element statement. For example, if the element contains four nodes, I3 denotes the branch current output for the third node; if n is not specified, the first node is assumed.
Xzzz	subcircuit element name, used to call the subcircuit (if any) within which Xyyy is defined
Xyyy	subcircuit element name, used to call the subcircuit (if any) within which Wwww is defined
Wwww	element name

Examples

```
I1(R1)
```

This example specifies the current through the first node of resistor R1.

```
I4(X1.M1)
```

The above example specifies the current through the fourth node (the substrate node) of the MOSFET M1, which is defined in subcircuit X1.

```
I2(Q1)
```

The last example specifies the current through the second node (the base node) of the bipolar transistor Q1.

Define each branch circuit by a single element statement. Star-Hspice evaluates branch currents by inserting a zero-volt power supply in series with branch elements.

If Star-Hspice cannot interpret a .PRINT or .PLOT statement containing a branch current, a warning is generated.

Branch current direction for the elements in Figures 4-1 through 4-6 is defined in terms of arrow notation (current direction) and node position number (terminal type).

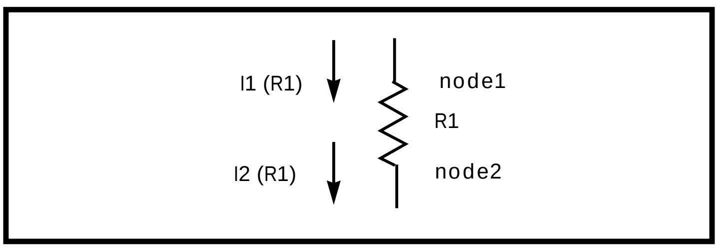

Figure 4-1: Resistor (node1, node2)

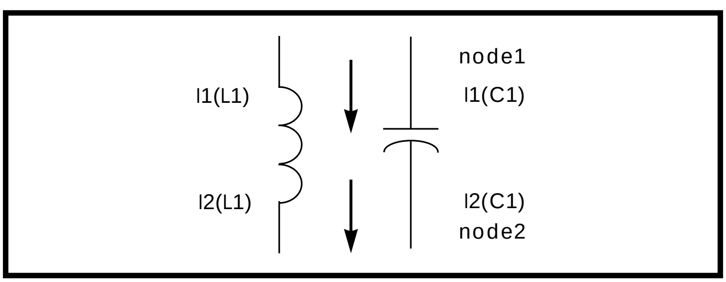

Figure 4-2: Capacitor (node1, node2); Inductor (node 1, node2)

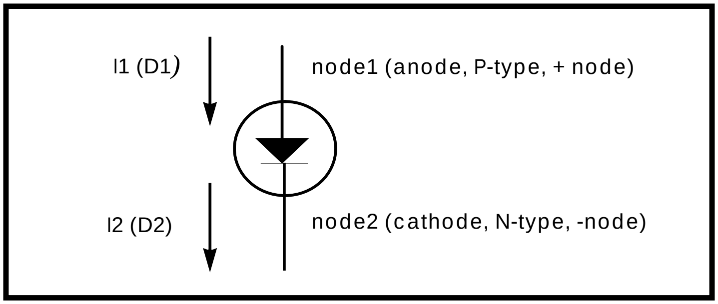

Figure 4-3: Diode (node1, node2)

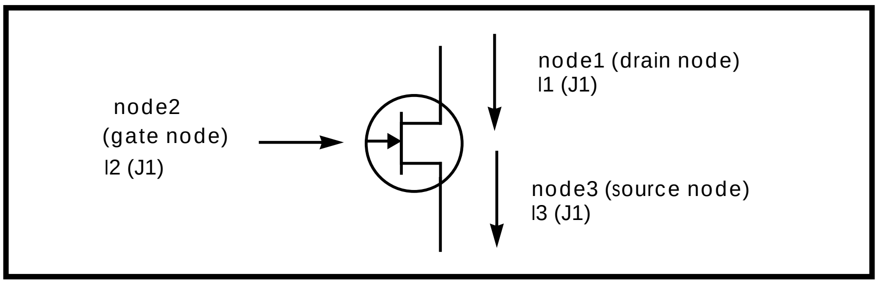

Figure 4-4: JFET (node1, node2, node3) - n-channel

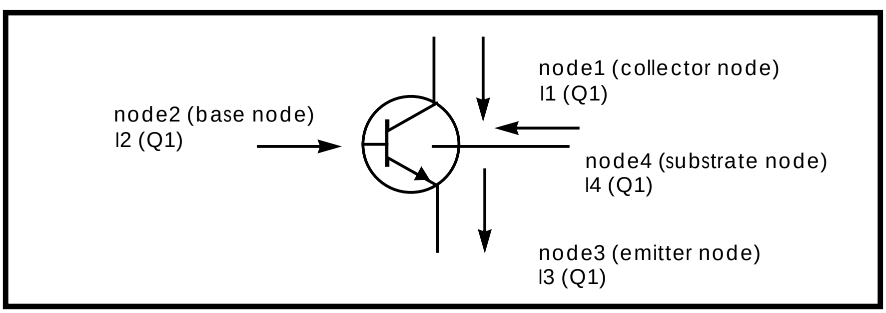

Figure 4-5: BJT (node1, node2, node3, node4) - npn

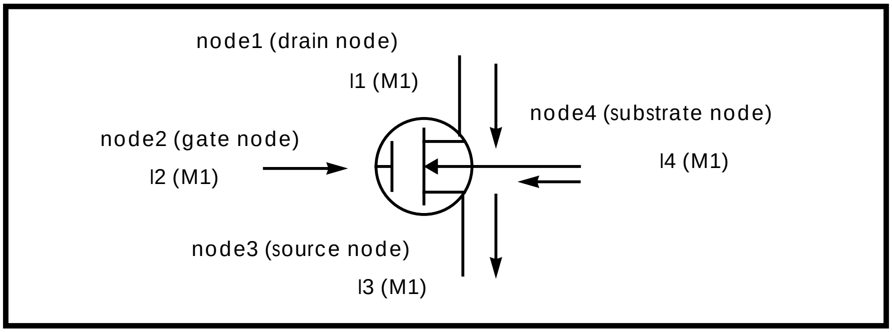

Figure 4-6: MOSFET (node1, node2, node3, node4) - n-channel

AC Analysis Output

This section describes the output for AC analysis.

Group Time Delay

The group time delay, TD, is associated with AC analysis and is defined as the negative derivative of phase, in radians, with respect to radian frequency. In Star-Hspice, the difference method is used to compute TD, as follows

$$TD = -\frac{1}{360} \cdot \frac{(phase2 - phase1)}{(f2 - f1)}$$

where phase1 and phase2 are the phases, in degrees, of the specified signal at the frequencies f1 and f2, in Hertz. See “Group Time Delay Output” on page 9-12 for more information about printing and plotting group time delays.

Also see “AC Analysis Output” on page 4-14 for descriptions and examples of output variables for AC analysis.

Noise and Distortion Analysis Output Variables

This section describes the variables used for noise and distortion analysis.

Syntax

```
ovar <(z)>
```

where:

ovar	noise and distortion analysis parameter. It can be either ONOISE (output noise), or INOISE (equivalent input noise) or any of the distortion analysis parameters (HD2, HD3, SIM2, DIM2, DIM3).
z	output type (only for distortion). If z is omitted, the magnitude of the output variable is output.

Examples

```
.PRINT DISTO HD2(M) HD2(DB)
```

Prints the magnitude and decibel values of the second harmonic distortion component through the load resistor specified in the .DISTO statement (not shown).

```
.PLOT NOISE INOISE ONOISE
```

Note: The noise and distortion output variable may be specified along with other AC output variables in the .PRINT AC or .PLOT AC statements.

Power Output

For power calculations, Star-Hspice computes dissipated or stored power in each passive element (R, L, C), and source (V, I, G, E, F, and H) by multiplying the voltage across an element and its corresponding branch current. However, for semiconductor devices, Star-Hspice calculates only the dissipated power. The power stored in the device junction or parasitic capacitances is excluded from the device power computation. Equations for calculating the power dissipated in different types of devices are shown in the following sections.

Star-Hspice also computes the total power dissipated in the circuit, which is the sum of the power dissipated in the devices, resistors, independent current sources, and all the dependent sources. For hierarchical designs, Star-Hspice computes the power dissipation for each subcircuit as well.

Note: For the total power (dissipated power + stored power), it is possible to add up the power of each independent source (voltage and current sources).

Print or Plot Power

Output the instantaneous element power and the total power dissipation using a .PRINT or .PLOT statement.

Syntax

```
.PRINT <DC | TRAN> P(element_or_subcircuit_name)POWER
```

Power calculation is associated only with transient and DC sweep analyses. The .MEASURE statement can be used to compute the average, rms, minimum, maximum, and peak-to-peak value of the power. The POWER keyword invokes the total power dissipation output.

Examples

```
.PRINT TRAN    P(M1)       P(VIN)      P(CLOAD)    POWER
.PRINT TRAN    P(Q1)       P(DIO)      P(J10)      POWER
.PRINT TRAN    POWER       $ Total transient analysis power
* dissipation
.PLOT DC POWER       P(IIN)      P(RLOAD)    P(R1)
.PLOT DC POWER       P(V1)       P(RLOAD)    P(VS)
.PRINT TRAN P(Xf1) P(Xf1.Xh1)
```

Diode Power Dissipation

$$Pd = Vpp' \cdot (Ido + Icap) + Vp'n \cdot Ido$$

Pd	power dissipated in diode
Ido	DC component of the diode current
Icap	capacitive component of the diode current
Vp'n	voltage across the junction
Vpp'	voltage across the series resistance RS

BJT Power Dissipation

Vertical

$$Pd = Vc'e' \cdot Ico + Vb'e' \cdot Ibo + Vcc' \cdot Ictot + Vee' \cdot Ietot + Vsc' \cdot Iso - Vcc' \cdot Istot$$

Lateral

$$Pd = Vc'e' \cdot Ico + Vb'e' \cdot Ibo + Vcc' \cdot Ictot + Vbb' \cdot Ibtot + Vee' \cdot Ietot + Vsb' \cdot Iso - Vbb' \cdot Istot$$

Ibo	DC component of the base current
Ico	DC component of the collector current
Iso	DC component of the substrate current
Pd	power dissipated in BJT
Ibtot	total base current (excluding the substrate current)
Ictot	total collector current (excluding the substrate current)
Ietot	total emitter current
Istot	total substrate current
Vb'e'	voltage across the base-emitter junction
Vbb'	voltage across the series base resistance RB
Vc'e'	voltage across the collector-emitter terminals
Vcc'	voltage across the series collector resistance RC

Vee'	voltage across the series emitter resistance RE
Vsb'	voltage across the substrate-base junction
Vsc'	voltage across the substrate-collector junction

JFET Power Dissipation

$$Pd = Vd's' \cdot Ido + Vgd' \cdot Igdo + Vgs' \cdot Igso + Vs's \cdot (Ido + Igso + Icgs) + Vdd' \cdot (Ido - Igdo - Icgd)$$

Icgd	capacitive component of the gate-drain junction current
Icgs	capacitive component of the gate-source junction current
Ido	DC component of the drain current
Igdo	DC component of the gate-drain junction current
Igso	DC component of the gate-source junction current
Pd	power dissipated in JFET
Vd's'	voltage across the internal drain-source terminals
Vdd'	voltage across the series drain resistance RD
Vgd'	voltage across the gate-drain junction
Vgs'	voltage across the gate-source junction
Vs's	voltage across the series source resistance RS

MOSFET Power Dissipation

$$Pd = Vd's' \cdot Ido + Vbd' \cdot Ibdo + Vbs' \cdot Ibso + Vs's \cdot (Ido + Ibso + Icbs + Icgs) + Vdd' \cdot (Ido - Ibdo - Icbd - Icgd)$$

Ibdo	DC component of the bulk-drain junction current
Ibso	DC component of the bulk-source junction current
Icbd	capacitive component of the bulk-drain junction current
Icbs	capacitive component of the bulk-source junction current
Icgd	capacitive component of the gate-drain current
Icgs	capacitive component of the gate-source current

Ido	DC component of the drain current
Pd	power dissipated in the MOSFET
Vbd'	voltage across the bulk-drain junction
Vbs'	voltage across the bulk-source junction
Vd's'	voltage across the internal drain-source terminals
Vdd'	voltage across the series drain resistance RD
Vs's	voltage across the series source resistance RS

.MEASURE Statement

Use the .MEASURE statement to modify information and define the results of successive simulations.

The .MEASURE statement prints user-defined electrical specifications of a circuit and is used extensively in optimization. The specifications include propagation, delay, rise time, fall time, peak-to-peak voltage, minimum and maximum voltage over a specified period, and a number of other user-defined variables. With the error mode and GOAL parameter, .MEASURE is also used extensively for optimization of circuit component values and curve fitting measured data to model parameters.

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

Fundamental measurement modes are:

- Rise, fall, and delay
- Average, RMS, min, max, peak-to-peak, and integral
- Find-when
- Equation evaluation
- Derivative evaluation
- Integral evaluation
- Relative error

When a .MEASURE statement fails to execute, Star-Hspice writes 0.0e0 in the *.mt#* file as the .MEASURE result, and writes “FAILED” in the output listing file.

Rise, Fall, and Delay

This format is used to measure independent variable (time, frequency, or any parameter or temperature) differential measurements such as rise time, fall time, slew rate, and any measurement that requires the determination of independent variable values. The format specifies substatements TRIG and TARG. These two statements specify the beginning and ending of a voltage or current amplitude measurement.

The rise, fall, and delay measurement mode computes the time, voltage, or frequency between a trigger value and a target value. Examples for transient analysis include rise/fall time, propagation delay, and slew rate measurement. Applications for AC analysis are the measurement of the bandwidth of an amplifier or the frequency at which a certain gain is achieved.

Syntax

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

where:

MEASURE specifies measurements. You can abbreviate to MEAS.

result is the name given the measured value in the Star-Hspice output. The item measured is the independent variable beginning at the trigger and ending at the target: for transient analysis it is time; for AC analysis it is frequency; for DC analysis it is the DC sweep variable. If the target is reached before the trigger is activated, the resulting value is negative. **Note:** The terms “DC”, “TRAN”, and “AC” are illegal for *result* name.

TRIG…, TARG … identifies the beginning of trigger and target specifications, respectively.

<DC|AC|TRAN>	specifies the analysis type of the measurement. If omitted, the last analysis mode requested is assumed.
GOAL	specifies the desired measure value in optimization. The error is calculated by $ERRfun = (GOAL - result) / GOAL$.
MINVAL	If the absolute value of GOAL is less than MINVAL, the GOAL value is replaced by MINVAL in the denominator of the ERRfun expression. Default=1.0e-12.
WEIGHT	The calculated error is multiplied by the weight value. Used in optimization. Default=1.0.

TRIG (Trigger) Syntax

```
TRIG trig_var VAL=trig_val <TD=time_delay> <CROSS=c> <RISE=r>
+   <FALL=f>
```

or

```
TRIG AT=val
```

TARG (Target) Syntax

```
TARG targ_var VAL=targ_val <TD=time_delay> <CROSS=c | LAST>
+  <RISE=r | LAST> <FALL=f | LAST>
```

where:

TRIG	indicates the beginning of the trigger specification
trig_val	is the value of *trig_var* at which the counter for crossing, rises, or falls is incremented by one
trig_var	specifies the name of the output variable, which determines the logical beginning of measurement. If the target is reached before the trigger is activated, .MEASURE reports a negative value.
TARG	indicates the beginning of the target signal specification

targ_val	specifies the value of the *targ_var* at which the counter for crossing, rises, or falls is incremented by one
targ_var	the name of the output variable whose propagation delay is determined with respect to the *trig_var*
time_delay	the amount of simulation time that must elapse before the measurement is enabled. The number of crossings, rises, or falls is counted only after *time_delay* value. The default trigger delay is zero.
CROSS=c *RISE=r* *FALL=f*	The numbers indicate which occurrence of a CROSS, FALL, or RISE event causes a measurement to be performed. For RISE=r, the WHEN condition is met and measurement is performed when the designated signal has risen *r* rise times. For FALL =f, measurement is performed when the designated signal has fallen *f* fall times. A crossing is either a rise or a fall, so for CROSS=c, measurement is performed when the designated signal has achieved a total of *c* crossing times, as a result of either rising or falling. For TARG, the last event is specified with the LAST keyword.
LAST	Measurement is performed when the last CROSS, FALL, or RISE event occurs. For CROSS = LAST, measurement is performed the last time the WHEN condition is true for either a rising or falling signal. For FALL = LAST, measurement is performed the last time the WHEN condition is true for a falling signal. For RISE = LAST, measurement is performed the last time the WHEN condition is true for a rising signal. LAST is a reserved word and cannot be chosen as a parameter name in the above .MEASURE statements.
AT=val	a special case for trigger specification. The “val” is the time for TRAN analysis, the frequency for AC analysis, or the parameter for DC analysis, at which measurement is to start.

Examples

```
.MEASURE TRAN tdlay TRIG V(1) VAL=2.5 TD=10n RISE=2
+                   TARG V(2) VAL=2.5        FALL=2
```

This example specifies that a propagation delay measurement is taken between nodes 1 and 2 for a transient analysis. The delay is measured from the second rising edge of the voltage at node 1 to the second falling edge of node 2. The measurement is specified to begin when the second rising voltage at node 1 is 2.5 V and to end when the second falling voltage at node 2 reaches 2.5 V. The TD=10n parameter does not allow the crossings to be counted until after 10 ns has elapsed. The results are printed as tdlay=<value>.

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

```
.MEASURE pwidth TRIG AT=10n TARG V(IN) VAL=2.5 CROSS=3
```

The last example uses the short form of TRIG. AT=10n specifies that the time measurement is to begin at time t=10 ns in the transient analysis. The TARG parameters specify that the time measurement is to end when V(IN)=2.5 V on the third crossing. The variable *pwidth* is the printed output variable.

Note: If the .TRAN statement is used in conjunction with a .MEASURE statement, using a nonzero START time in the .TRAN statement can result in incorrect .MEASURE results. Do not use nonzero START times in .TRAN statements when .MEASURE is also being used.

Average, RMS, MIN, MAX, and Peak-To-Peak Measurements

The average (AVE), RMS, MIN, MAX, and peak-to-peak (PP) measurement modes report functions of the output variable rather than the analysis value. Average calculates the area under the output variable divided by the periods of interest. RMS takes the square root of the area under the output variable square divided 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. Integral provides the integral of an output variable over a specified period.

Syntax

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

where:

<DC|AC|TRAN> specifies the analysis type of the measurement. If omitted, the last analysis mode requested is assumed.

FROM specifies the initial value for the “func” calculation. For transient analysis, value is in units of time.

TO specifies the end of the “func” calculation.

GOAL specifies the desired .MEASURE value. It is used in optimization. The error is calculated by
$ERRfun = (GOAL - result) / GOAL$

MINVAL If the absolute value of GOAL is less than MINVAL, the GOAL value is replaced by MINVAL in the denominator of the ERRfun expression. Default=1.0e-12.

func indicates the type of the measure statement, one of the following:

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_var^2 curve divided by the period of interest

INTEG integral: reports the integral of *out_var* over the specified interval

result	the name given the measured value in the Star-Hspice output. The value is a function of the variable specified (*out_var*) and func.
out_var	the name of any output variable whose function (“func”) is to be measured in the simulation.
WEIGHT	The calculated error is multiplied by the weight value. Default=1.0.

Examples

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

The example above calculates the average nodal voltage value for node 10 during the transient sweep from the time 10 ns to 55 ns and prints out the result as “avgval”.

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

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

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

FIND and WHEN Functions

The FIND and WHEN functions allow any independent variables (time, frequency, parameter), any dependent variables (voltage or current, for example), or the derivative of any dependent variables to be measured when some specific event occurs. These measure statements are useful in unity gain frequency or phase measurements, as well as for measuring the time, frequency, or any parameter value when two signals cross each other, or when a signal crosses a constant value. The measurement starts after a specified time delay, TD. It is possible to find a specific event by setting RISE, FALL, or CROSS to a value (or parameter) or LAST for last event. LAST is a reserved word and cannot be chosen as a parameter name in the above measure statements. See *.MEASURE Statement, page -19 in , Specifying Simulation Output* for the definitions of parameters on measure statement.

The syntax is:

```
.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>
```

or

```
.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>
```

or

```
.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>
```

or

```
.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>
```

or

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

Parameter Definitions

CROSS=c *RISE=r* *FALL=f*	The numbers indicate which occurrence of a CROSS, FALL, or RISE event causes a measurement to be performed. For RISE=r, the WHEN condition is met and measurement is performed when the designated signal has risen *r* rise times. For FALL =f, measurement is performed when the designated signal has fallen *f* fall times. A crossing is either a rise or a fall, so for CROSS=c, measurement is performed 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 of the measurement. If omitted, the last analysis type requested is assumed.
FIND	selects the FIND function

GOAL	specifies the desired .MEASURE value. It is used in optimization. The error is calculated by $ERRfun = (GOAL - result)/GOAL$.
LAST	Measurement is performed when the last CROSS, FALL, or RISE event occurs. For CROSS = LAST, measurement is performed the last time the WHEN condition is true for either a rising or falling signal. For FALL = LAST, measurement is performed the last time the WHEN condition is true for a falling signal. For RISE = LAST, measurement is performed the last time the WHEN condition is true for a rising signal. LAST is a reserved word and cannot be chosen as a parameter name in the above .MEASURE statements.
MINVAL	If the absolute value of GOAL is less than MINVAL, the GOAL value is replaced by MINVAL in the denominator of the ERRfun expression. Default=1.0e-12.
out_var(1,2,3)	variables used to establish conditions at which measurement is to take place
result	the name given the measured value in the Star-Hspice output
TD	identifies the time at which measurement is to start
WEIGHT	the calculated error is multiplied by the weight value. Default=1.0.
WHEN	selects the WHEN function

Equation Evaluation

Use this 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.

Syntax

```
.MEASURE <DC|TRAN|AC> result PARAM='equation'
```

```
+ <GOAL=val> <MINVAL=val>
```

DERIVATIVE Function

The DERIVATIVE function provides the derivative of an output variable at a given time or frequency or for any sweep variable, depending on the type of analysis. It also provides the derivative of a specified output variable when some specific event occurs.

Syntax

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

or

```
.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>
```

or

```
.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>
```

where:

AT=val — the value of *out_var* at which the derivative is to be found

CROSS=c
RISE=r
FALL=f — The numbers indicate which occurrence of a CROSS, FALL, or RISE event causes a measurement to be performed. For RISE=r, the WHEN condition is met and measurement is performed when the designated signal has risen *r* rise times. For FALL =f, measurement is performed when the designated signal has fallen *f* fall times. A crossing is either a rise or a fall, so for CROSS=c, measurement is performed 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 measured. If omitted, the last analysis mode requested is assumed.

DERIVATIVE	selects the derivative function. May be abbreviated to DERIV.
GOAL	specifies the desired .MEASURE value. It is used in optimization. The error is calculated by $ERRfun = (GOAL - result) / GOAL$.
LAST	Measurement is performed when the last CROSS, FALL, or RISE event occurs. For CROSS = LAST, measurement is performed the last time the WHEN condition is true for either a rising or falling signal. For FALL = LAST, measurement is performed that last time the WHEN condition is true for a falling signal. For RISE = LAST, measurement is performed the last time the WHEN condition is true for a rising signal. LAST is a reserved word and cannot be chosen as a parameter name in the above .MEASURE statements.
MINVAL	If the absolute value of GOAL is less than MINVAL, the GOAL value is replaced by MINVAL in the denominator of the ERRfun expression. Default=1.0e-12.
out_var	the variable for which the derivative is to be found
result	the name given the measured value in the Star-Hspice output
TD	identifies the time at which measurement is to start
var(2,3)	variables used to establish conditions at which measurement is to take place
WEIGHT	The calculated error between result and GOAL is multiplied by the weight value. Default=1.0.
WHEN	selects the WHEN function

Examples

The following example calculates the derivative of V(out) at 25 ns:

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

The following example calculates the derivative of v(1) when v(1) is equal to 0.9*vdd:

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

The following example calculates the derivative of VP(output)/360.0 when the frequency is 10 kHz:

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

INTEGRAL Function

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

Syntax

```
.MEASURE <DC|TRAN|AC> result INTEGRAL out_var1 <FROM=val1> <TO=val2>
+ <GOAL=goalval> <MINVAL=minval> <WEIGHT=weightval>
```

where:

<DC|AC|TRAN>	specifies the analysis type of the measurement. If omitted, the last analysis mode requested is assumed.
result	the name given the measured value in the Star-Hspice output
INTEGRAL	selects the integral function. It can be abbreviated to INTEG.
FROM	the lower limit of integration
TO	the upper limit of integration
GOAL	specifies the desired .MEASURE value. It is used in optimization. The error is calculated by $ERRfun = (GOAL - result)/GOAL$
MINVAL	If the absolute value of GOAL is less than MINVAL, the GOAL value is replaced by MINVAL in the denominator of the ERRfun expression. Default=1.0e-12.
WEIGHT	the calculated error between result and GOAL is multiplied by the weight value. Default=1.0.

Example

The following example calculates the integral of I(cload) from t=10 ns to t = 100 ns:

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

ERROR Function

The relative error function reports the relative difference of two output variables. This format is often used in optimization and curve fitting of measured data. The relative error format specifies the variable to be measured and calculated from the .PARAM variables. The relative error between the two is calculated using 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..

Syntax

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

where:

<DC|AC|TRAN>	specifies the analysis type of the measurement. If omitted, the last analysis mode requested is assumed.
result	the name given the measured result in the output
ERRfun	ERRfun indicates which error function to use: ERR, ERR1, ERR2, or ERR3.
meas_var	the name of any output variable or parameter in the data statement. M denotes the *meas_var* in the error equation.
calc_var	the name of the simulated output variable or parameter in the .MEASURE statement to be compared with *meas_var*. C denotes the *calc_var* in the error equation.
IGNOR | YMIN	If the absolute value of *meas_var* is less than IGNOR value, then this point is not considered in the ERRfun calculation. 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	The calculated error is multiplied by the weight value. Default=1.0.
YMAX	If the absolute value of *meas_var* is greater than the YMAX value, then this point is not considered in the ERRfun calculation. Default=1.0e+15.
TO	specifies the end of the ERRfun calculation. Defaults to the last value of the sweep variable.
MINVAL	If the absolute value of *meas_var* is less than MINVAL, the *meas_var* value is replaced by MINVAL in the denominator of the ERRfun expression. Default=1.0e-12.

ERR

ERR sums the squares of (M-C)/max (M, MINVAL) for each point, divides by the number of points, and then takes the square root of the result. M (meas_var) and C (calc_var) are the measured and calculated values of the device or circuit response, respectively. NPTS is the number of data points.

$$ERR = \left[\frac{1}{NPTS} \cdot \sum_{i=1}^{NPTS} \left(\frac{M_i - C_i}{\max(MINVAL, M_i)} \right)^2 \right]^{1/2}$$

ERR1

ERR1 computes the relative error at each point. For NPTS points, there are NPTS ERR1 error function calculations. For device characterization, the ERR1 approach has been found to be more efficient than the other error functions (ERR, ERR2, ERR3).

$$ERR1_i = \frac{M_i - C_i}{\max(MINVAL, M_i)} ,i=1,NPTS$$

Star-Hspice does not print out each calculated ERR1 value. When the ERR1 option is set, it returns an ERR value calculated as follows:

$$ERR = \left[\frac{1}{NPTS} \cdot \sum_{i=1}^{NPTS} ERR1_i^2\right]^{1/2}$$

ERR2

This option computes the absolute relative error at each point. For NPTS points, there are NPTS error function calls.

$$ERR2_i = \left|\frac{M_i - C_i}{\max(MINVAL, M_i)}\right| ,i=1,NPTS$$

The returned value printed for ERR2 is

$$ERR = \frac{1}{NPTS} \cdot \sum_{i=1}^{NPTS} ERR2_i$$

ERR3

$$ERR3_i = \frac{\pm\log\left|\frac{M_i}{C_i}\right|}{\left|\log[\max(MINVAL, |M_i|)]\right|} ,i=1,NPTS$$

The + sign corresponds to a positive M/C ratio. The – sign corresponds to a negative M/C ratio.

Note: If the measured value M is less than MINVAL, the MINVAL is used instead. Also, if the absolute value of M is less than the IGNOR | YMIN value or greater than the YMAX value, then this point is not considered in the error calculation.

To prevent parameter values given in .MEASURE statements from overwriting parameter assignments in other statements, Star-Hspice keeps track of parameter types. If the same parameter name is used in both a .MEASURE statement and a .PARAM statement at the same hierarchical level, Star-Hspice terminates with an error. No error occurs if the parameter assignments are at different hierarchical levels. PRINT statements that occur at different levels do not print hierarchical information for the parameter name headings.

The following example illustrates how Star-Hspice handles .MEASURE statement parameters.

```
...
.MEASURE tran length TRIG v(clk) VAL=1.4 TD=11ns RISE=1
+ TARGv(neq) VAL=1.4 TD=11ns RISE=1
.SUBCKT path out in width=0.9u length=600u
+ rm1 in m1 m2mg w='width' l='length/6'
...
.ENDS
```

In the above listing, the ‘length’ in the resistor statement

```
rm1 in m1 m2mg w='width' l='length/6'
```

does not inherit its value from the length in the .MEASURE statement

```
.MEASURE tran length ...
```

since they are of different types. The correct value of l in rm1 should be

```
l = length/6 = 100u
```

instead of a value derived from the measured value in transient analysis.

Element Template Output

Element templates are used in .PRINT, .PLOT, .PROBE, and .GRAPH statements for output of user-input parameters, state variables, stored charges, capacitor currents, capacitances, and derivatives of variables. The Star-Hspice element templates are listed in this section.

Format of Element Template Output

The form is:

```
Elname:Property
```

Elname name of the element

Property property name of an element, such as a user-input parameter, state variable, stored charge, capacitance current, capacitance, or derivative of a variable

The alias is:

```
LVnn(Elname)
```

or

```
LXnn(Elname)
```

LV form to obtain output of user-input parameters, and state variables

LX form to obtain output of stored charges, capacitor currents, capacitances, and derivatives of variables

nn code number for the desired parameter, given in the tables in this section

Elname name of the element

Examples

```
.PLOT TRAN V(1,12) I(X2.VSIN) I2(Q3) DI01:GD
.PRINT TRAN X2.M1:CGGBO M1:CGDBO X2.M1:CGSBO
```

Element Template Listings

Resistor

Name	Alias	Description
G	LV1	conductance at analysis temperature
R	LV2	resistance at reference temperature
TC1	LV3	first temperature coefficient
TC2	LV4	second temperature coefficient

Capacitor

Name	Alias	Description
CEFF	LV1	computed effective capacitance
IC	LV2	initial condition
Q	LX0	charge stored in capacitor
CURR	LX1	current flowing through capacitor
VOLT	LX2	voltage across capacitor
–	LX3	capacitance (not used in Star-Hspice releases after 95.3)

Inductor

Name	Alias	Description
LEFF	LV1	computed effective inductance
IC	LV2	initial condition
FLUX	LX0	flux in the inductor
VOLT	LX1	voltage across inductor

Inductor

Name	Alias	Description
CURR	LX2	current flowing through inductor
–	LX4	inductance (not used in Star-Hspice releases after 95.3)

Mutual Inductor

Name	Alias	Description
K	LV1	mutual inductance

Voltage-Controlled Current Source

Name	Alias	Description
CURR	LX0	current through the source, if VCCS
R	LX0	resistance value, if VCR
C	LX0	capacitance value, if VCCAP
CV	LX1	controlling voltage
CQ	LX1	capacitance charge, if VCCAP
DI	LX2	derivative of source current with respect to control voltage
ICAP	LX2	capacitance current, if VCCAP
VCAP	LX3	voltage across capacitance, if VCCAP

Voltage-Controlled Voltage Source

Name	Alias	Description
VOLT	LX0	source voltage
CURR	LX1	current through source

Voltage-Controlled Voltage Source

Name	Alias	Description
CV	LX2	controlling voltage
DV	LX3	derivative of source voltage with respect to control current

Current-Controlled Current Source

Name	Alias	Description
CURR	LX0	current through source
CI	LX1	controlling current
DI	LX2	derivative of source current with respect to control current

Current-Controlled Voltage Source

Name	Alias	Description
VOLT	LX0	source voltage
CURR	LX1	source current
CI	LX2	controlling current
DV	LX3	derivative of source voltage with respect to control current

Independent Voltage Source

Name	Alias	Description
VOLT	LV1	DC/transient voltage
VOLTM	LV2	AC voltage magnitude
VOLTP	LV3	AC voltage phase

Independent Current Source

Name	Alias	Description
CURR	LV1	DC/transient current
CURRM	LV2	AC current magnitude
CURRP	LV3	AC current phase

Diode

Name	Alias	Description
AREA	LV1	diode area factor
AREAX	LV23	area after scaling
IC	LV2	initial voltage across diode
VD	LX0	voltage across diode (VD), excluding RS (series resistance)
IDC	LX1	DC current through diode (ID), excluding RS. Total diode current is the sum of IDC and ICAP
GD	LX2	equivalent conductance (GD)
QD	LX3	charge of diode capacitor (QD)
ICAP	LX4	current through diode capacitor.
		Total diode current is the sum of IDC and ICAP.
C	LX5	total diode capacitance
PID	LX7	photocurrent in diode

BJT

Name	Alias	Description
AREA	LV1	area factor

BJT

Name	Alias	Description
ICVBE	LV2	initial condition for base-emitter voltage (VBE)
ICVCE	LV3	initial condition for collector-emitter voltage (VCE)
MULT	LV4	number of multiple BJTs
FT	LV5	FT (Unity gain bandwidth)
ISUB	LV6	substrate current
GSUB	LV7	substrate conductance
LOGIC	LV8	LOG 10 (IC)
LOGIB	LV9	LOG 10 (IB)
BETA	LV10	BETA
LOGBETAI	LV11	LOG 10 (BETA) current
ICTOL	LV12	collector current tolerance
IBTOL	LV13	base current tolerance
RB	LV14	base resistance
GRE	LV15	emitter conductance, 1/RE
GRC	LV16	collector conductance, 1/RC
PIBC	LV18	photocurrent, base-collector
PIBE	LV19	photocurrent, base-emitter
VBE	LX0	VBE
VBC	LX1	base-collector voltage (VBC)
CCO	LX2	collector current (CCO)
CBO	LX3	base current (CBO)
GPI	LX4	g_{π} = ib / vbe, constant vbc
GU	LX5	g_{μ} = ib / vbc, constant vbe
GM	LX6	g_m = ic / vbe+ ic / vbe, constant vce

BJT

Name	Alias	Description
G0	LX7	$g_0 =$ ic / vce constant vbe
QBE	LX8	base-emitter charge (QBE)
CQBE	LX9	base-emitter charge current (CQBE)
QBC	LX10	base-collector charge (QBC)
CQBC	LX11	base-collector charge current (CQBC)
QCS	LX12	current-substrate charge (QCS)
CQCS	LX13	current-substrate charge current (CQCS)
QBX	LX14	base-internal base charge (QBX)
CQBX	LX15	base-internal base charge current (CQBX)
GXO	LX16	1/Rbeff Internal conductance (GXO)
CEXBC	LX17	base-collector equivalent current (CEXBC)
–	LX18	base-collector conductance (GEQCBO) (not used in Star-Hspice releases after 95.3)
CAP_BE	LX19	cbe capacitance ($C\Pi$)
CAP_IBC	LX20	cbc internal base-collector capacitance ($C\mu$)
CAP_SCB	LX21	csc substrate-collector capacitance for vertical transistors csb substrate-base capacitance for lateral transistors
CAP_XBC	LX22	cbcx external base-collector capacitance
CMCMO	LX23	(TF*IBE) / vbc
VSUB	LX24	substrate voltage

JFET

Name	Alias	Description
AREA	LV1	JFET area factor
VDS	LV2	initial condition for drain-source voltage
VGS	LV3	initial condition for gate-source voltage
PIGD	LV16	photocurrent, gate-drain in JFET
PIGS	LV17	photocurrent, gate-source in JFET
VGS	LX0	VGS
VGD	LX1	gate-drain voltage (VGD)
CGSO	LX2	gate-to-source (CGSO)
CDO	LX3	drain current (CDO)
CGDO	LX4	gate-to-drain current (CGDO)
GMO	LX5	transconductance (GMO)
GDSO	LX6	drain-source transconductance (GDSO)
GGSO	LX7	gate-source transconductance (GGSO)
GGDO	LX8	gate-drain transconductance (GGDO)
QGS	LX9	gate-source charge (QGS)
CQGS	LX10	gate-source charge current (CQGS)
QGD	LX11	gate-drain charge (QGD)
CQGD	LX12	gate-drain charge current (CQGD)
CAP_GS	LX13	gate-source capacitance
CAP_GD	LX14	gate-drain capacitance
–	LX15	body-source voltage (not used in Star-Hspice releases after 95.3)
QDS	LX16	drain-source charge (QDS)
CQDS	LX17	drain-source charge current (CQDS)
GMBS	LX18	drain-body (backgate) transconductance (GMBS)

MOSFET

Name	Alias	Description
L	LV1	channel length (L)
W	LV2	channel width (W)
AD	LV3	area of the drain diode (AD)
AS	LV4	area of the source diode (AS)
ICVDS	LV5	initial condition for drain-source voltage (VDS)
ICVGS	LV6	initial condition for gate-source voltage (VGS)
ICVBS	LV7	initial condition for bulk-source voltage (VBS)
–	LV8	device polarity: 1=forward, -1=reverse (not used in Star-Hspice releases after 95.3)
VTH	LV9	threshold voltage (bias dependent)
VDSAT	LV10	saturation voltage (VDSAT)
PD	LV11	drain diode periphery (PD)
PS	LV12	source diode periphery (PS)
RDS	LV13	drain resistance (squares) (RDS)
RSS	LV14	source resistance (squares) (RSS)
XQC	LV15	charge sharing coefficient (XQC)
GDEFF	LV16	effective drain conductance (1/RDeff)
GSEFF	LV17	effective source conductance (1/RSeff)
IDBS	LV18	drain-bulk saturation current at -1 volt bias
ISBS	LV19	source-bulk saturation current at -1 volt bias
VDBEFF	LV20	effective drain bulk voltage
BETAEFF	LV21	BETA effective
GAMMAEFF	LV22	GAMMA effective
DELTAL	LV23	ΔL (MOS6 amount of channel length modulation) (only valid for Levels 1, 2, 3 and 6)

MOSFET

Name	Alias	Description
UBEFF	LV24	UB effective (only valid for Levels 1, 2, 3 and 6)
VG	LV25	VG drive (only valid for Levels 1, 2, 3 and 6)
VFBEFF	LV26	VFB effective
–	LV31	drain current tolerance (not used in Star-Hspice releases after 95.3)
IDSTOL	LV32	source diode current tolerance
IDDTOL	LV33	drain diode current tolerance
COVLGS	LV36	gate-source overlap capacitance
COVLGD	LV37	gate-drain overlap capacitance
COVLGB	LV38	gate-bulk overlap capacitance
VBS	LX1	bulk-source voltage (VBS)
VGS	LX2	gate-source voltage (VGS)
VDS	LX3	drain-source voltage (VDS)
CDO	LX4	DC drain current (CDO)
CBSO	LX5	DC source-bulk diode current (CBSO)
CBDO	LX6	DC drain-bulk diode current (CBDO)
GMO	LX7	DC gate transconductance (GMO)
GDSO	LX8	DC drain-source conductance (GDSO)
GMBSO	LX9	DC substrate transconductance (GMBSO)
GBDO	LX10	conductance of the drain diode (GBDO)
GBSO	LX11	conductance of the source diode (GBSO)
Meyer and Charge Conservation Model Parameters		
QB	LX12	bulk charge (QB)
CQB	LX13	bulk charge current (CQB)

MOSFET

Name	Alias	Description
QG	LX14	gate charge (QG)
CQG	LX15	gate charge current (CQG)
QD	LX16	channel charge (QD)
CQD	LX17	channel charge current (CQD)
CGGBO	LX18	$CGGBO = \partial Qg / \partial Vg$ = CGS + CGD + CGB
CGDBO	LX19	$CGDBO = \partial Qg / \partial Vd$, (for Meyer CGD=-CGDBO)
CGSBO	LX20	$CGSBO = \partial Qg / \partial Vs$, (for Meyer CGS=-CGSBO)
CBGBO	LX21	$CBGBO = \partial Qb / \partial Vg$, (for Meyer CGB=-CBGBO)
CBDBO	LX22	$CBDBO = \partial Qb / \partial Vd$
CBSBO	LX23	$CBSBO = \partial Qb / \partial Vs$
QBD	LX24	drain-bulk charge (QBD)
–	LX25	drain-bulk charge current (CQBD) (not used in Star-Hspice releases after 95.3)
QBS	LX26	source-bulk charge (QBS)
–	LX27	source-bulk charge current (CQBS) (not used in Star-Hspice releases after 95.3)
CAP_BS	LX28	bulk-source capacitance
CAP_BD	LX29	bulk-drain capacitance
CQS	LX31	channel charge current (CQS)
CDGBO	LX32	$CDGBO = \partial Qd / \partial Vg$
CDDBO	LX33	$CDDBO = \partial Qd / \partial Vd$
CDSBO	LX34	$CDSBO = \partial Qd / \partial Vs$

Saturable Core Element

Name	Alias	Description
MU	LX0	dynamic permeability (mu) Weber/(amp-turn-meter)
H	LX1	magnetizing force (H) Ampere-turns/meter
B	LX2	magnetic flux density (B) Webers/meter^2

Saturable Core Winding

Name	Alias	Description
LEFF	LV1	effective winding inductance (Henry)
IC	LV2	initial Condition
FLUX	LX0	flux through winding (Weber-turn)
VOLT	LX1	voltage across winding (Volt)

Displaying Simulation Results

The following section describes the statements used to display simulation results for your specific requirements.

.PRINT Statement

The .PRINT statement specifies output variables for which values are printed.

The maximum number of variables in a single .PRINT statement is 32. You can use additional .PRINT statements for more output variables.

To simplify parsing of the output listings, a single “x” printed in the first column indicates the beginning of the .PRINT output data, and a single “y” in the first column indicates the end of the .PRINT output data.

Syntax

```
.PRINT antype ov1 <ov2 … ov32>
```

where

antype specifies the type of analysis for outputs. Antype is one of the following types: DC, AC, TRAN, NOISE, or DISTO.

ov1 … specifies output variables to be print. These are voltage, current, or element template variables from a DC, AC, TRAN, NOISE, or DISTO analysis.

Examples

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

This example prints out the results of a transient analysis for the nodal voltage named 4 and the current through the voltage source named VIN. The ratio of the nodal voltage at node “OUT” and node “IN” is also printed.

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

VM(4,2) specifies that the AC magnitude of the voltage difference (or the difference of the voltage magnitudes, depending on the value of the ACOUT option) between nodes 4 and 2 is printed. VR(7) specifies that the real part of the AC voltage between nodes 7 and ground is printed. VP(8,3) specifies that 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 depending on the value of ACOUT options) is printed. II(R1) specifies that the imaginary part of the current through R1 is printed.

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

The above example specifies that the magnitude of the input impedance, the phase of the output admittance, and several S and Z parameters are to be printed. This statement would accompany a network analysis using the .AC and .NET analysis statements.

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

This example specifies that the DC analysis results are tprinted for several different nodal voltages and currents through the resistor named R1, the voltage source named VSRC, and the drain- to-source current of the MOSFET named M1.

```
.PRINT NOISE INOISE
```

In this example the equivalent input noise is printed.

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

This example prints the magnitude of the third-order harmonic distortion and the decibel value of the intermodulation distortion sum through the load resistor specified in the .DISTO statement.

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

In this statement, specifications of NOISE, DISTO, and AC output variables are included on the same .PRINT statements.

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

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

Note: Star-Hspice ignores .PRINT statement references to nonexistent netlist part names and prints those names in a warning message.

Print Control Options

The number of output variables printed on a single line of output is a function of the number of columns, set by the option CO. Typical values are CO=80 for narrow printouts and CO=132 for wide printouts. CO=80 is the default. The maximum number of output variables allowed is 5 per 80-column output and 8 per 132-column output with twelve characters per column. Star-Hspice automatically creates additional print statements and tables for all output variables beyond the number specified by the CO option. Variable values are printed in engineering notation by default:

```
F = 1e-15          M = 1e-3
P = 1e-12          K = 1e3
N = 1e-9           X = 1e6
U = 1e-6           G = 1e9
```

In contrast to the exponential format, the engineering notation provides two to three extra significant digits and aligns columns to facilitate comparison. To obtain output in exponential format, specify INGOLD = 1 or 2 with an .OPTION statement.

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
1.23e-01

Subcircuit Output Printing

The following examples demonstrate how to print or plot voltages of nodes in subcircuit definitions using .PRINT or .PLOT.

Note: .PROBE, .PLOT, or .GRAPH may be substituted for .PRINT in the following example.

Example 1

```
.GLOBAL vdd vss
X1 1 2 3 nor2
X2 3 4 5 nor2
.SUBCKT nor2 A B Y
  .PRINT v(B) v(N1)$ Print statement 1
  M1 N1 A vdd vdd pch w=6u l=0.8u
  M2 Y B N1 vdd pch w=6u l=0.8u
  M3 Y A vss vss nch w=3u l=0.8u
  M4 Y B vss vss nch w=3u l=0.8u
.ENDS
```

Print statement 1 invokes a printout of the voltage on input node B and internal node N1 for every instance of the nor2 subcircuit.

```
  .PRINT v(1) v(X1.A)$ Print statement 2
```

The print statement above specifies two ways of printing the voltage on input A of instance X1

```
  .PRINT v(3) v(X1.Y) v(X2.A)$ Print statement 3
```

This print statement specifies three different ways of printing the voltage at output Y of instance X1. (input A of instance X2).

```
  .PRINT v(X2.N1)$ Print statement 4
```

The print statement above prints out the voltage on the internal node N1 of instance X2.

```
  .PRINT  i(X1.M1)$ Print statement 5
```

The print statement above prints out the drain-to-source current through MOSFET M1 in instance X1.

Example 2

```
X1 5 6 YYY
 .SUBCKT YYY 15 16
 X2 16 36 ZZZ
 R1 15 25 1
 R2 25 16 1
 .ENDS
.SUBCKT ZZZ 16 36
 C1 16 0 10P
 R3 36 56 10K
 C2 56 0 1P
 .ENDS
.PRINT  V(X1.25) V(X1.X2.56) V(6)
```

The .PRINT statement voltages are:

V(X1.25) local node to subcircuit definition YYY, called by subcircuit X1

V(X1.X2.56) local node to subcircuit definition ZZZ, called by subcircuit X2, which was called by X1

V(6) represents the voltage of node 16 in instance X1 of subcircuit YYY

This example prints analysis results for the voltage at node 56 within the subcircuits X2 and X1. The full path name X1.X2.56 specifies that node 56 is within subcircuit X2 that is within subcircuit X1.

.WIDTH Statement

The syntax is:

```
.WIDTH OUT={80 |132}
```

where *OUT* is the output print width

Example

```
.WIDTH OUT=132  $ SPICE compatible style
```

```
.OPTION CO=132  $ preferred style
```

Permissible values for OUT are 80 and 132. OUT can also be set with option CO.

.PLOT Statement

The .PLOT statement plots output values of one or more variables in a selected analysis. Each .PLOT statement defines the contents of one plot, which can have 1 to 32 output variables.

When no plot limits are specified, Star-Hspice automatically determines the minimum and maximum values of each output variable being plotted and scales each plot to fit common limits. To cause Star-Hspice to set limits for certain variables, set the plot limits to (0,0) for those variables.

To make Star-Hspice find plot limits for each plot individually, select .OPTION PLIM to create a different axis for each plot variable. The PLIM option is similar to the plot limit algorithm in SPICE2G.6. In the latter case, each plot can have limits different from any other plot. The overlap of two or more traces on a plot is indicated by a number from 2 through 9.

When more than one output variable appears on the same plot, the first variable specified is printed as well as plotted. If a printout of more than one variable is desired, include another .PLOT statement.

The number of .PLOT statements you can specify for each type of analysis is unlimited. Plot width is set by the option CO (columns out). For a CO setting of 80, a 50-column plot is produced. If CO is 132, a 100-column plot is produced.

Syntax

```
.PLOT antype ov1 <(plo1,phi1)> … <ov32>
+ <(plo32,phi32)>
```

where:

antype	the type of analysis for the specified plots. Analysis types are: DC, AC, TRAN, NOISE, or DISTO.

ov1 … output variables to plot. These are voltage, current, or element template variables from a DC, AC, TRAN, NOISE, or DISTO analysis. See the following sections for syntax.

plo1,phi1 … lower and upper plot limits. Each output variable is plotted using the first set of plot limits following the output variable. Output variables following a plot limit should have a new plot limit. For example, to plot all output variables with the same scale, specify one set of plot limits at the end of the PLOT statement. Setting the plot limits to (0,0) causes Star-Hspice to set the plot limits.

Examples

In the following example, PAR invokes the plot of the ratio of the collector current and the base current of the transistor Q1.

```
.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
```

The second of the two examples above uses the VDB output variable to plot the AC analysis results of the node named 5 in decibels. Also, NOISE results may be requested along with the other variables in the AC plot.

```
.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 example above, Star-Hspice sets the plot limits for V(1) and V(2), while 0 and 5 volts are specified as the plot limits for V(3) and V(4).

.PROBE Statement

The .PROBE statement saves output variables into the interface and graph data files. Star-Hspice usually saves all voltages and supply currents in addition to the output variables. Set .OPTION PROBE to save output variables only. Use the .PROBE statement to specify which quantities are to be printed in the output listing.

If you are only interested in the output data file and do not want tabular or plot data in your listing file, set .OPTION PROBE and use the .PROBE statement to specify which values you want saved in the output listing.

Syntax

```
.PROBE antype ov1 … <ov32>
```

antype the type of analysis for the specified plots. Analysis types are: DC, AC, TRAN, NOISE, or DISTO.

ov1 … output variables to be plotted. These are voltage, current, or element template variables from a DC, AC, TRAN, NOISE, or DISTO analysis. The limit for the number of output variables in a single .PROBE statement is 32. Additional .PROBE statements may be used to deal with more output variables.

Example

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

.GRAPH Statement

Note: The .GRAPH statement is not provided in the PC version of Star-Hspice.

The .GRAPH statement allows high resolution plotting of simulation results. This statement is similar to the .PLOT statement with the addition of an optional model. When a model is specified, you can add or change graphing properties for the graph. The .GRAPH statement generates a *.gr#* graph data file and sends this file directly to the default high resolution graphical device (specified by PRTDEFAULT in the *meta.cfg* configuration file).

Each .GRAPH statement creates a new *.gr#* file, where # ranges first from 0 to 9, and then from a to z. The maximum number of graph files that can exist is 36. If more than 36 .GRAPH statements are used, the graph files are overwritten starting with the *.gr0* file.

Syntax

```
.GRAPH antype <MODEL=mname> <unam1=> ov1,
+ <unam2=>ov2, … <unam32=> ov32 (plo,phi)
```

where

antype — the type of analysis for the specified plots. Analysis types are: DC, AC, TRAN, NOISE, or DISTO.

mname — the plot modelname referenced by the .GRAPH statement. The .GRAPH statement and its plot name allow high resolution plots to be made from Star-Hspice directly.

unam1… — user-defined output names, which correspond to output variables ov1…ov32 (*unam1* to *unam32* respectively), are used as labels instead of output variables for a high resolution graphic output.

ov1 …ov2 — output variables to be printed, 32 maximum. They can be voltage, current, or element template variables from a different type analysis. Algebraic expressions also are used as output variables, but they must be defined inside the PAR( ) statement.

plo, phi — lower and upper plot limits. Set the plot limits only at the end of the .GRAPH statement.

.MODEL Statement for .GRAPH

This section describes the model statement for .GRAPH.

Syntax

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

mname the plot model name referenced by the .GRAPH statements

PLOT the keyword for a .GRAPH statement model

pnam1=val1… Each .GRAPH statement model includes a variety of model parameters. If no model parameters are specified, Star-Hspice takes the default values of the model parameters described in the following table. Pnam*n* is one of the model parameters of a .GRAPH statement, and val*n* is the value of

Model Parameters

Name(Alias)	Default	Description
FREQ	0.0	plots symbol frequency. Value 0 suppresses plot symbol generation; a value of n generates a plot symbol every n points.
MONO	0.0	monotonic option. MONO=1 automatically resets x-axis if any change in x direction.
TIC	0.0	shows tick marks
XGRID, YGRID	0.0	setting to 1.0 turns on the axis grid lines
XMIN, XMAX	0.0	If XMIN is not equal to XMAX, then XMIN and XMAX determines the x-axis plot limits. If XMIN equals XMAX, or if XMIN and XMAX are not set, then the limits are automatically set. These limits apply to the actual x-axis variable value regardless of the XSCAL type.
XSCAL	1.0	scale for the x-axis. Two common axis scales are: Linear(LIN) (XSCAL=1) Logarithm(LOG) (XSCAL=2)

Name(Alias)	Default	Description
YMIN,YMAX	0.0	If YMIN is not equal to YMAX, then YMIN and YMAX determines the y-axis plot limits. The y-axis limits specified in the .GRAPH statement override YMIN and YMAX in the model. If limits are not specified then they are automatically set. These limits apply to the actual y-axis variable value regardless of the YSCAL type.
YSCAL	1.0	scale for the y-axis. Two common axis scales are: Linear(LIN) (YSCAL=1) Logarithm(LOG) (YSCAL=2)

Examples

```
.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
```