WinSpice3 User's Manual

4 August, 2002

Mike Smith

Based on Spice 3 User Manual

by

T.Quarles, A.R.Newton, D.O.Pederson, A.Sangiovanni-Vincentelli

Department of Electrical Engineering and Computer Sciences

University of California

Berkeley, Ca., 94720

WinSpice3 User Manual

Table Of Contents

1 INTRODUCTION

WinSpice3 is a general-purpose circuit simulation program for non-linear DC, non-linear transient, and linear AC analyses. Circuits may contain resistors, capacitors, inductors, mutual inductors, independent voltage and current sources, four types of dependent sources, lossless and lossy transmission lines (two separate implementations), switches, uniform distributed RC lines, and the five most common semiconductor devices: diodes, BJTs, JFETs, MESFETs, and MOSFETs.

WinSpice3 is based on **Spice3F4**[1] which in turn was developed from **SPICE2G.6**. While **WinSpice3** is being developed to include new features, it continues to support those capabilities and models which remain in extensive use in the **SPICE2** program.

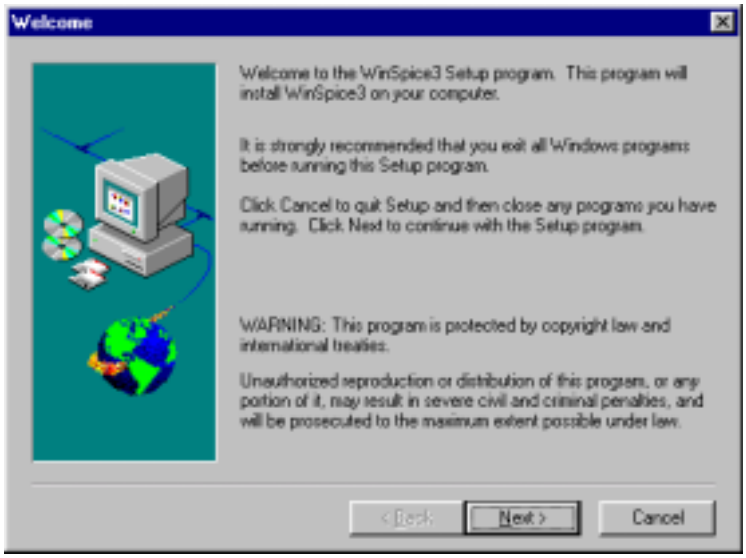
1.1 Installation

WinSpice3 is supplied as a self-extracting .ZIP file called SPICE3.EXE. When executed, the setup files for WinSpice are placed in the directory defined by the TEMP environment string by default as shown below.

If you want to unzip to a different directory, edit the folder path.

Now, click on the Unzip button to unpack the setup files. Navigate to the folder containing the unzipped files and run setup.exe by double clicking on its icon. The dialogue shown below should appear.

[1] Spice3F4 was developed by the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley.



Make sure you read the **readme.txt** file which might contain some useful information and give details of how to contact the author (it is amazing how many people don't read ANY of the documents – when in doubt RTFM!!).

Note that **WinSpice3** adds no files to the Windows directories!

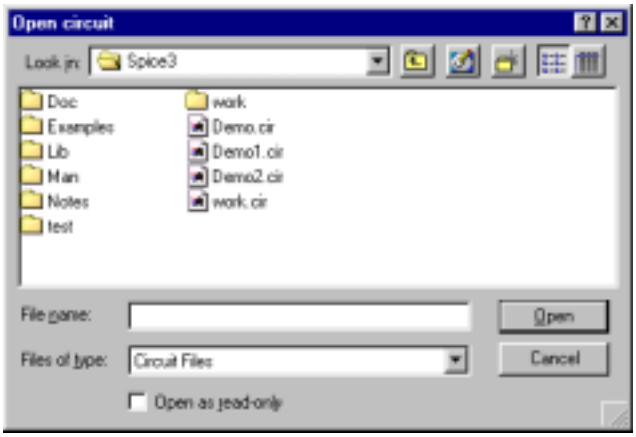
1.2 Running WinSpice3

Click on 'Start', point to 'Programs' and find the WinSpice3 popout. Click on 'wspice3' to run the program. The following window (or something like it) will appear:-

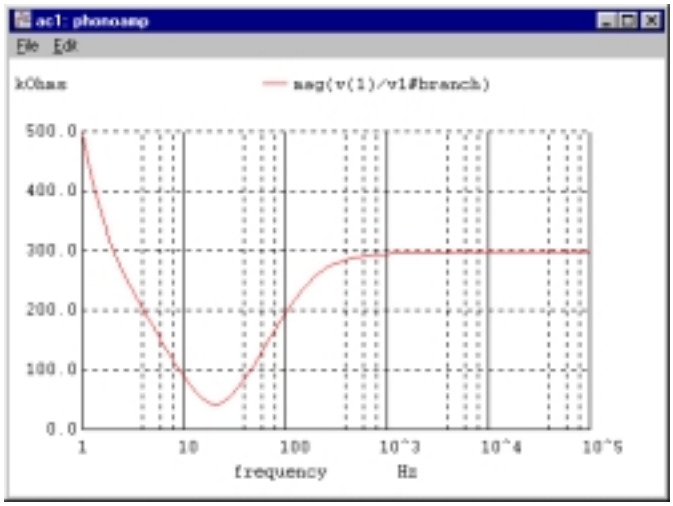
This window emulates a terminal window as is seen in versions of Spice3 running on Unix machines.

At this point, **WinSpice3** will accept numerous commands typed in at the keyboard (see section 6.9 for details of the commands supported). The command interpreter is based on the Unix C-shell and it is possible to write complex programs with it. For example, the **setplot** command (see section 6.9.42) is implemented using such a 'script' (look in the lib\script directory in the directory **WinSpice3** is installed in for the script).

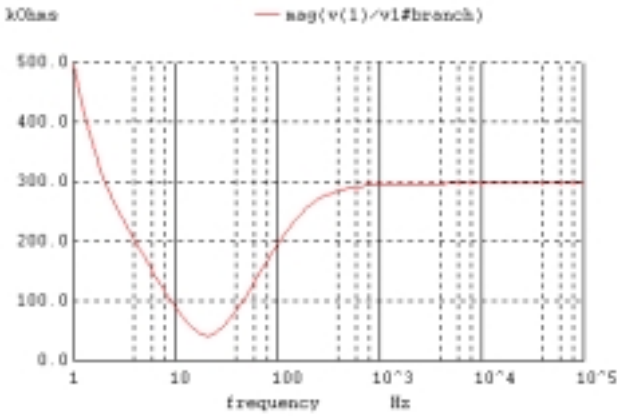
However, since you haven't read that far yet, the quickest way of running a simulation is to open one of the circuit files in the examples directory. To do this, click 'File', 'Open'. The dialogue box shown below will appear.



Double click on 'Examples' and then double click on 'Phonoamp.cir'. As soon as the file is loaded, it begins simulating the circuit and generating plot windows as it goes. Make one of the plot windows the active window.



The plot can be resized by dragging the window border. The plot can be printed to the default printer by clicking on 'File', 'Print'. The plot can be copied to the clipboard by clicking 'Edit', 'Copy' and then pasting the plot into a document e.g.



You can also zoom into an area of a plot by clicking on the graph and dragging the mouse to select the required area. A zoomed-in graph will be displayed when you release the mouse button.

A tutorial, in the form of a Word document, is also provided and you should run this tutorial to understand some of the basic concepts of **WinSpice3**.

1.3 Uninstalling WinSpice3

Use the 'Add/Remove Programs' applet in the Control Panel to uninstall the program.

1.4 Command Line Options

```
wspice3 [-n][-b][-i][-r rawfile] [input file ...]
```

Options are:

-n (or **-N**)
Don't try to source the file **.spiceinit** upon start-up. Normally **WinSpice3** tries to find the file in the current directory, and if it is not found then in the directory containing the **WinSpice3** program.

-b
Batch mode. Simulates the input file and writes the results to a rawfile. After the circuit has been simulated, WinSpice will exit.

-i
Interactive mode (default). WinSpice simulates the input file and continues running. It then monitors the state of the input file. If it changes in any way, WinSpice will reload the circuit.

-r rawfile
Specifies the name of the output rawfile. This causes WinSpice to output results directly to the file.

Further arguments to **WinSpice3** are taken to be **SPICE3** input files, which are read and saved (if running in batch mode then they are run immediately). **WinSpice3** accepts most **SPICE2** input files, and output ASCII plots, Fourier analyses, and node printouts as specified in .plot, .four, and .print cards. If an out parameter is given on a .width card, the effect is the same as **set width = ...** Since **WinSpice3** ASCII plots do not use multiple ranges, however, if vectors together on a .plot card have different ranges they do not provide as much information as they would in **SPICE2**. The output of **WinSpice3** is also much less verbose than **SPICE2**, in that the only data printed is that requested by the above cards.

2 TYPES OF ANALYSIS

2.1 DC Analysis

The DC analysis portion of SPICE determines the DC operating point of the circuit with inductors shorted and capacitors opened. The DC analysis options are specified on the .DC, .TF, and .OP control lines. A DC analysis is automatically performed prior to a transient analysis to determine the transient initial conditions, and prior to an AC small-signal analysis to determine the linearized, small-signal models for non-linear devices. If requested, the DC small-signal value of a transfer function (ratio of output variable to input source), input resistance, and output resistance is also computed as a part of the DC solution. The DC analysis can also be used to generate DC transfer curves: a specified independent voltage or current source is stepped over a user-specified range and the DC output variables are stored for each sequential source value.

2.2 AC Small-Signal Analysis

The AC small-signal portion of **WinSpice3** computes the AC output variables as a function of frequency. The program first computes the DC operating point of the circuit and determines linearized, small-signal models for all of the non-linear devices in the circuit. The resultant linear circuit is then analysed over a user-specified range of frequencies. The desired output of an AC small-signal analysis is usually a transfer function (voltage gain, transimpedance, etc.). If the circuit has only one AC input, it is convenient to set that input to unity and zero phase, so that output variables have the same value as the transfer function of the output variable with respect to the input.

2.3 Transient Analysis

The transient analysis portion of **WinSpice3** computes the transient output variables as a function of time over a user-specified time interval. The initial conditions are automatically determined by a DC analysis. All sources which are not time dependent (for example, power supplies) are set to their DC value. The transient time interval is specified on a .TRAN control line.

2.4 Pole-Zero Analysis

The pole-zero analysis portion of **WinSpice3** computes the poles and/or zeros in the small-signal AC transfer function. The program first computes the DC operating point and then determines the linearized, small-signal models for all the non-linear devices in the circuit. This circuit is then used to find the poles and zeros of the transfer function.

Two types of transfer functions are allowed: one of the form (output voltage)/(input voltage) and the other of the form (output voltage)/(input current). These two types of transfer functions cover all the cases and one can find the poles/zeros of functions like input/output impedance and voltage gain. The input and output ports are specified as two pairs of nodes.

The pole-zero analysis works with resistors, capacitors, inductors, linear-controlled sources, independent sources, BJTs, MOSFETs, JFETs and diodes. Transmission lines are not supported.

The method used in the analysis is a sub-optimal numerical search. For large circuits it may take a considerable time or fail to find all poles and zeros. For some circuits, the method becomes "lost" and finds an excessive number of poles or zeros.

2.5 Small-Signal Distortion Analysis

The distortion analysis portion of **WinSpice3** computes steady-state harmonic and intermodulation products for small input signal magnitudes. If signals of a single frequency are specified as the input to the circuit, the complex values of the second and third harmonics are determined at every point in the circuit. If there are signals of two frequencies input to the circuit, the analysis finds out the complex values of the

circuit variables at the sum and difference of the input frequencies, and at the difference of the smaller frequency from the second harmonic of the larger frequency.

Distortion analysis is supported for the following non-linear devices: diodes (DIO), BJT, JFET, MOSFETs (levels 1, 2, 3, 4/BSIM1, 5/BSIM2, and 6) and MESFETs. All linear devices are automatically supported by distortion analysis. If there are switches present in the circuit, the analysis continues to be accurate provided the switches do not change state under the small excitations used for distortion calculations.

2.6 Sensitivity Analysis

WinSpice3 will calculate either the DC operating-point sensitivity or the AC small-signal sensitivity of an output variable with respect to all circuit variables, including model parameters. **WinSpice3** calculates the difference in an output variable (either a node voltage or a branch current) by perturbing each parameter of each device independently. Since the method is a numerical approximation, the results may demonstrate second order affects in highly sensitive parameters, or may fail to show very low but non-zero sensitivity. Further, since each variable is perturbed by a small fraction of its value, zero-valued parameters are not analysed (this has the benefit of reducing what is usually a very large amount of data).

2.7 Noise Analysis

The noise analysis portion of **WinSpice3** does analysis device-generated noise for the given circuit. When provided with an input source and an output port, the analysis calculates the noise contributions of each device (and each noise generator within the device) to the output port voltage. It also calculates the input noise to the circuit, equivalent to the output noise referred to the specified input source. This is done for every frequency point in a specified range - the calculated value of the noise corresponds to the spectral density of the circuit variable viewed as a stationary gaussian stochastic process.

After calculating the spectral densities, noise analysis integrates these values over the specified frequency range to arrive at the total noise voltage/current (over this frequency range). This calculated value corresponds to the variance of the circuit variable viewed as a stationary gaussian process.

2.8 Analysis At Different Temperatures

All input data for **WinSpice3** is assumed to have been measured at a nominal temperature of 27 C, which can be changed by use of the TNOM parameter on the .OPTION control line. This value can further be overridden for any device which models temperature effects by specifying the TNOM parameter on the model itself. The circuit simulation is performed at a temperature of 27 C, unless overridden by a TEMP parameter on the .OPTION control line. Individual instances may further override the circuit temperature through the specification of a TEMP parameter on the instance.

Temperature dependent support is provided for resistors, capacitors, diodes, JFETs, BJTs, and level 1, 2, and 3 MOSFETs. BSIM (levels 4 and 5) MOSFETs have an alternate temperature dependency scheme that adjusts all of the model parameters before input to SPICE. For details of the BSIM temperature adjustment, see [6] and [7].

Temperature appears explicitly in the exponential terms of the BJT and diode model equations. In addition, saturation currents have built-in temperature dependence. The temperature dependence of the saturation current in the BJT models is determined by:

$$I_S(T_1) = I_S(T_0)\left(\frac{T_1}{T_0}\right)^{XTI} \exp\left(-\frac{qE_g}{kT1}\left(1-\frac{T_1}{T_0}\right)\right)$$

where k is Boltzmann's constant, q is the electronic charge, E_g is the energy gap which is a model parameter, and XTI is the saturation current temperature exponent (also a model parameter, and usually equal to 3).

The temperature dependence of forward and reverse beta is according to the formula:

$$\beta(T_1) = \beta(T_0)\left(\frac{T_1}{T_0}\right)^{XTB}$$

where T_1 and T_0 are in degrees Kelvin, and XTB is a user-supplied model parameter. Temperature effects on beta are carried out by appropriate adjustment to the values of β_F, I_{SE}, β_R, and I_{SC} (**WinSpice3** model parameters BF, ISE, BR, and ISC, respectively).

Temperature dependence of the saturation current in the junction diode model is determined by:

$$I_S(T_1) = I_S(T_0)\left(\frac{T_1}{T_0}\right)^{\frac{XTI}{N}} \exp\left(-\frac{qE_g}{NkT_1}\left(1-\frac{T_1}{T_0}\right)\right)$$

where N is the emission coefficient, which is a model parameter, and the other symbols have the same meaning as above. Note that for Schottky barrier diodes, the value of the saturation current temperature exponent, XTI, is usually 2.

Temperature appears explicitly in the value of junction potential, ϕ (in **WinSpice3** PHI), for all the device models. The temperature dependence is determined by:

$$\Phi(T) = \frac{kT}{q}\log_e\left(\frac{N_a N_d}{N_i(T)^2}\right)$$

where k is Boltzmann's constant, q is the electronic charge, N_a is the acceptor impurity density, N_d is the donor impurity density, N_i is the intrinsic carrier concentration, and E_g is the energy gap.

Temperature appears explicitly in the value of surface mobility, μ_0 (or UO), for the MOSFET model. The temperature dependence is determined by:

$$\mu_0(T) = \frac{\mu_0(T_0)}{\left(\frac{T}{T_0}\right)^{1.5}}$$

The effects of temperature on resistors is modelled by the formula:

$$R(T) = R(T_0)\left[1 + TC_1(T-T_0) + TC_2(T-T_0)^2\right]$$

where T is the circuit temperature, T_0 is the nominal temperature, and TC_1 and TC_2 are the first- and second-order temperature coefficients.

3 CIRCUIT DESCRIPTION

3.1 General Structure And Conventions

The circuit to be analysed is described to WinSpice3 by a set of element lines, which define the circuit topology and element values, and a set of control lines, which define the model parameters and the run controls. The first line in the input file must be the title, and the last line must be ".END". The order of the remaining lines is arbitrary (except, of course, that continuation lines must immediately follow the line being continued).

An element line that contains the element name, the circuit nodes to which the element is connected, and the values of the parameters that determine the electrical characteristics of the element specify each element in the circuit. The first letter of the element name specifies the element type. The format for the SPICE element types is given in what follows. The strings XXXXXXX, YYYYYYY, and ZZZZZZZ denote arbitrary alphanumeric strings. For example, a resistor name must begin with the letter R and can contain one or more characters. Hence, R, R1, RSE, ROUT, and R3AC2ZY are valid resistor names. Details of each type of device are supplied in a following section.

Fields on a line are separated by one or more blanks, a comma, an equal ('=') sign, or a left or right parenthesis; extra spaces are ignored. A line may be continued by entering a '+' (plus) in column 1 of the following line; **WinSpice3** continues reading beginning with column 2.

A name field must begin with a letter (A through Z) and cannot contain any delimiters.

A number field may be an integer field (12, -44), a floating point field (3.14159), either an integer or floating point number followed by an integer exponent (1e-14, 2.65e3), or either an integer or a floating point number followed by one of the following scale factors:

$T = 10^{12}$	$G = 10^{9}$	$Meg = 10^{6}$	$K = 10^{3}$	$mil = 25.4^{-6}$
$m = 10^{-3}$	u (or M) $= 10^{-6}$	$N = 10^{-9}$	$p = 10^{-12}$	$f = 10^{-15}$

Letters immediately following a number that are not scale factors are ignored, and letters immediately following a scale factor are ignored. Hence, 10, 10V, 10Volts, and 10Hz all represent the same number, and M, MA, MSec, and MMhos all represent the same scale factor. Note that 1000, 1000.0, 1000Hz, 1e3, 1.0e3, 1KHz, and 1K all represent the same number.

Nodes names may be arbitrary character strings. The datum (ground) node must be named '0'. Note the difference in **WinSpice3** where the nodes are treated as character strings and not evaluated as numbers, thus '0' and '00' are distinct nodes in **WinSpice3** but not in **SPICE2**. The circuit cannot contain a loop of voltage sources and/or inductors and cannot contain a cut-set of current sources and/or capacitors.

Each node in the circuit must have a DC path to ground.

Every node must have at least two connections except for transmission line nodes (to permit unterminated transmission lines) and MOSFET substrate nodes (which have two internal connections anyway).

3.2 Title Line, Comment Lines And .END Line

3.2.1 Title Line

Examples:

```
POWER AMPLIFIER CIRCUIT
TEST OF CAM CELL
```

The title line must be the first in the input file. Its contents are printed verbatim as the heading for each section of output.

3.2.2 .END Line

Examples:

```
.END
```

The "End" line must always be the last in the input file. Note that the period is an integral part of the name.

3.2.3 Comments

General Form:

```
* <any comment>
```

Examples:

```
* RF=1K      Gain should be 100
* Check open-loop gain and phase margin
```

The asterisk in the first column indicates that this line is a comment line. Comment lines may be placed anywhere in the circuit description. Note that **WinSpice3** also considers any line with leading white space to be a comment.

3.3 .MODEL: Device Models

General form:

```
.MODEL MNAME TYPE(PNAME1=PVAL1 PNAME2=PVAL2 ... )
```

Examples:

```
.MODEL MOD1 NPN (BF=50 IS=1E-13 VBF=50)
```

Most simple circuit elements typically require only a few parameter values. However, some devices (semiconductor devices in particular) that are included in **WinSpice3** require many parameter values. Often, many devices in a circuit are defined by the same set of device model parameters. For these reasons, a set of device model parameters is defined on a separate .MODEL line and assigned a unique model name. The device element lines in **WinSpice3** then refer to the model name.

For these more complex device types, each device element line contains the device name, the nodes to which the device is connected, and the device model name. In addition, other optional parameters may be specified for some devices: geometric factors and an initial condition (see the following section on Transistors and Diodes for more details).

MNAME in the above is the model name, and type is one of the following types:

R	Semiconductor resistor model
C	Semiconductor capacitor model
SW VSWITCH	Voltage controlled switch
CSW ISWITCH	Current controlled switch
URC	Uniform distributed RC model
LTRA	Lossy transmission line model
D	Diode model
NPN	NPN BJT model
PNP	PNP BJT model
NJF	N-channel JFET model
PJF	P-channel JFET model
NMOS	N-channel MOSFET model
PMOS	P-channel MOSFET model
NMF	N-channel MESFET model
PMF	P-channel MESFET model

Parameter values are defined by appending the parameter name followed by an equal sign and the parameter value. Model parameters that are not given a value are assigned the default values given below for each model type. Models, model parameters, and default values are listed in the next section along with the description of device element lines.

3.4 Subcircuits

A subcircuit that consists of **WinSpice3** elements can be defined and referenced in a fashion similar to device models. The subcircuit is defined in the input file by a grouping of element lines; the program then automatically inserts the group of elements wherever the subcircuit is referenced. There is no limit on the size or complexity of subcircuits, and subcircuits may contain other subcircuits. An example of subcircuit usage is given in Appendix A.

3.4.1 .SUBCKT Line

General form:

```
.SUBCKT subnam N1 <N2 N3 ...>
```

Examples:

```
.SUBCKT OPAMP 1 2 3 4
```

A circuit definition is begun with a .SUBCKT line. SUBNAM is the subcircuit name, and N1, N2, are the external nodes, which cannot be zero. The group of element lines which immediately follow the .SUBCKT line define the subcircuit. The last line in a subcircuit definition is the .ENDS line (see section 3.4.2).

Control lines may not appear within a subcircuit definition. However, subcircuit definitions may contain anything else, including other subcircuit definitions, device models, and subcircuit calls (see section 3.4.4).

Note that any device models or subcircuit definitions included as part of a subcircuit definition are strictly local (i.e., such models and definitions are not known outside the subcircuit definition). Also, any element nodes not included on the .SUBCKT line are strictly local, with the exception of 0 (ground) which is always global.

Other nodes can be made global by using the .GLOBAL directive (see section 3.4.3).

3.4.2 .ENDS Line

General form:

```
.ENDS <SUBNAM>
```

Examples:

```
.ENDS OPAMP
```

The .ENDS line must be the last one for any subcircuit definition. The subcircuit name, if included, indicates which subcircuit definition is being terminated. If omitted, all subcircuits being defined are terminated. The name is needed only when nested subcircuit definitions are being made.

3.4.3 .GLOBAL Line

General form:

```
.GLOBAL N1 <N2 N3 ...>
```

Examples:

```
.GLOBAL 1 2 3 9
```

This line defines a set of global nodes. These nodes are not affected by subcircuit expansion.

3.4.4 Xxxxx: Subcircuit Calls

General form:

```
XYYYYYYY N1 <N2 N3 ...> SUBNAM
```

Examples:

```
X1 2 4 17 3 1 MULTI
```

Subcircuits are used in SPICE by specifying pseudo-elements beginning with the letter X, followed by the circuit nodes to be used in expanding the subcircuit.

3.5 Combining Files

3.5.1 .INCLUDE Lines

General form:

```
.INCLUDE filename
.INCLUDE “filename with spaces.cir”
```

Examples:

```
.INCLUDE \users\spice\common\wattmeter.cir
.INCLUDE “\users\spice files\wattmeter.cir”
```

Frequently, portions of circuit descriptions will be reused in several input files, particularly with common models and subcircuits. In any SPICE input file, the **.include** line may be used to copy some other file as if

that second file appeared in place of the ".include" line in the original file. There is no restriction on the file name imposed by SPICE beyond those imposed by the local operating system.

If the filename or path contain spaces, double quote marks must be used.

3.5.2 .LIB Lines

General form:

```
.LIB filename
.LIB “filenamewith spaces”
```

Examples:

```
.LIB \users\spice\common\bipolar.lib
```

This is an extension, not found in the Berkeley version of SPICE3, that provides backward compatibility with PSPICE.

The .LIB line is similar to the .INCLUDE line except that the specified file is assumed to contain .MODEL and .SUBCKT definitions. WinSpice3 searches for any undefined models or subcircuits in the specified file and extracts the required definitions and pastes them into the circuit. The main difference is that because it only extracts parts of the specified file and does not include the whole file in your circuit, the .LIB line uses far less memory.

The input file can have any extension, but by convention has the extension .lib.

If the filename or path contain spaces, double quote marks must be used.

As an example, consider the following call to a BC338AP BJT.

```
Q1 10 15 20 BC338AP
```

The model name, BC338AP, is the name of the library entry that contains the description of the transistor. The model is located in the library ZMODELS.LIB provided with WinSpice.

The .LIB ZMODELS.LIB statement would retrieve the model from the BJTN library.

```
SAMPLE NETLIST
.LIB ZMODELS.LIB
.DC VCE 0 15 .5 IB 100U 1M 100U
.PRINT DC I(VC)
IB 0 1
Q1 2 1 0 BC338AP
VC 3 2
VCE 3 0
.END
```

When the simulation is run, the model library ZMODELS.LIB will be searched for the BC338AP model statement which will be inserted into the final netlist.

```
SAMPLE NETLIST
.MODEL BC338AP NPN IS=3.941445E-14 BF=175 VAF=109.45 NF=1 IKF=.8
+ISE=7.4025E-15 NE=1.3 BR=20.5 VAR=14.25 NR=.974 IKR=.1 ISC=3.157E-13
+NC=1.2 RB=1.1 RE=.1259 RC=.0539 CJE=63E-12 TF=.75E-9 CJC=15.8E-12
+TR=85E-9 VJC=.505 MJC=.39
.PRINT DC I(VC)
IB 0 1
Q1 2 1 0 BC338AP
VC 3 2
VCE 3 0
.END
```

The netlist is loaded and the specified libraries are searched for unresolved subcircuit or model references in the order in which they appear in the netlist. Each library will be searched repeatedly until no additional references can be resolved in that library. The process is then repeated for succeeding libraries. The

program runs until a pass is made with no unresolved references. Libraries may cause additional unresolved references to occur if your subcircuits call other subcircuits or models. It is best to resolve those references within the same library.

In the example above, *INCLUDE may be substituted for .LIB with the same results (see section 3.6.1).

3.6 Extended Syntax

By extending the normal SPICE syntax, several new capabilities have been added to the standard WinSpice capabilities. These include the ability to:

- Call models and subcircuits from library files
- Pass parameters to the main circuit and to subcircuits
- Define and substitute expressions for keywords

These syntax extensions are made compatible with IsSpice and other Berkeley compatible SPICE versions by processing the input netlist through a series of pre-processors. These pre-processors are INCLUDE, DEFINE and PARAM.

3.6.1 *INCLUDE

General form:

```
*INCLUDE filename
*INCLUDE "filename with spaces.cir"
```

Examples:

```
*INCLUDE c:\spice\common\wattmeter.cir
*INCLUDE "c:\spice files\wattmeter.cir"
```

*INCLUDE acts as a combination of .INCLUDE (section 3.5.1) and .LIB (section 3.5.2) and is included in WinSpice for compatibility with commercial SPICE programs.

If 'filename' has a .LIB extension, *INCLUDE acts like the .LIB described earlier. It searches stored model library files (ASCII) for all subcircuits and device models that are not already in your input netlist. The appropriate models and subcircuits are automatically appended to the **WinSpice3** netlist.

Otherwise, *INCLUDE statement acts like .INCLUDE also described earlier and causes an entire file to be inserted into the netlist.

3.6.2 *DEFINE

General form:

```
*DEFINE variable name = <text string>
.DEFINE variable name = <text string>
```

Examples:

```
*DEFINE DUT=MPSA42
```

The *DEFINE feature is similar to the same feature found in commercial SPICE programs like IsSpice and is provided in WinSpice for compatibility as well as being a useful feature. It allows complicated expressions and statements to be defined by single keywords. These keywords can then be used throughout the netlist to decrease typing time and ease circuit debugging. Define statements may be placed anywhere in the netlist and will cause user-defined expressions to be substituted for keywords. *DEFINE may also be used in lower case, e.g. '*define', and can also be written '.define' as shown above.

*DEFINE allows a text string to be replaced with another text string within the netlist. This function can be used to easily change model names that are used numerous times, or to easily shorten long phrases. In the example above, every occurrence of the string ``DUT'' will be replaced by its substitute text string ``MPSA42''. The expression ``substitute text string'' may contain any characters. The substituted text is comprised of all the characters following the ``='' equals sign up until a carriage return is encountered.

*DEFINE statements are erased as they are performed, in order to eliminate duplicate substitutions. **WinSpice3** first scans the netlist for all *DEFINE lines and removes them from the netlist. It then scans the netlist again and makes the substitutions.

When using *DEFINE, the following rules and limitations should be noted:-

- *DEFINE statements are only processed in a forward direction. Define statements are usually placed at the beginning of the netlist in order to apply them to all subsequent entries.
- Be careful of what you are substituting. The variable name must be unique so that inadvertent substitutions are avoided.
- The variable name cannot start with a number.
- The *DEFINE statement cannot longer than one line long.

As an example, with the following netlist:-

```
*DEFINE WIDTH=5U
M1 1 2 3 4 WIDTH
M2 7 8 9 10 WIDTH
M20 34 45 23 12 WIDTH
```

When the netlist is loaded into WinSpice3, the *DEFINE lines are read and removed from the netlist. Then the netlist is scanned and substitutions made to give the following result:-

```
M1 1 2 3 4 5U
M2 7 8 9 10 5U
M20 34 45 23 12 5U
```

Note that this is a pure text substitution. Unlike the .PARAM substitutions (see section 3.6.3), if the substitution text contains an expression then this is not evaluated as the substitution is made.

3.6.3 .PARAM

General form:

```
.PARAM name1 = value1, ... namen = valuen
.PARAM name1 = { expression1 } ...  + namen = { expressionn }
```

Examples:

```
.PARAM VCC = 12V, VEE = -12V
.PARAM Freq=10K, Period={1/FREQ}, TRISE = {period/100}
.PARAM PI = 3.14159, TWO_PI = {2 * 3.14159}
.PARAM TEST = 1, Phase = 90
.PARAM K1 = {10 * Sin(Test) / 1 + TEST/180}
.PARAM K2 = {TEST < 1 ? PI : Exp(Test^2) * 5K}
```

The PARAM function is used to pass parameters into the main circuit and to subcircuits. They may then be used as is or inserted into mathematical expressions. The mathematical expressions will then be evaluated using the passed parameters and replaced with a resultant value.

Note that individual parameter definitions on a .PARAM line can be separated by a space or a comma. Hence

```
.PARAM VCC = 12V, VEE = -12V
```

and

```
.PARAM VCC = 12V VEE = -12V
```

are treated in the same way. Spaces within the values is not allowed because it will confuse **WinSpice3**.

3.6.3.1 Parameter Passing

Many electronic devices can be represented through the use of equations which are based on known or measured values. It would be helpful if these equations could be incorporated into a SPICE model and the model's behaviour controlled by supplying the dependent variables. This is exactly what parameter passing accomplishes.

Parameters can be passed from a .PARAM statement to the main circuit or to subcircuits via the X subcircuit call line. Parameters can also be passed directly from a subcircuit call line (X line) into a subcircuit. In both cases, parameters passed into a subcircuit can be further passed to another subcircuit down the hierarchy. Parameters can be used alone or as part of an expression.

Example, Parameter Passing To The Main Circuit:

```
.PARAM T1=1U T2=5U
V1 1 0 Pulse 0 1 0 {T1} {2*T1} {T2} {3*T2}
```

After parameters are passed and evaluated

```
V1 1 0 Pulse 0 1 0 1U 2U 5U 15U
```

Example, Parameter Passing To Subcircuits:

```
X1 1 2 3 4 XFMR {RATIO=3}

.SUBCKT XFMR 1 2 3 4
RP 1 2 1MEG
E1 5 4 1 2 {RATIO} ; parameterised expression in curly braces
F1 1 2 VM {RATIO * 2}
RS 6 3 1U
VM 5 6
.ENDS
```

Subcircuit after parameters are passed and evaluated

```
.SUBCKT XFMR 1 2 3 4
RP 1 2 1MEG
E1 5 4 1 2 3
F1 1 2 VM 6
RS 6 3 1U
VM 5 6
.ENDS
```

In the example, you can see that the subcircuit model for the transformer, XFMR, can represent many different transformers by merely changing the value of RATIO. Therefore, it is not necessary to construct a different subcircuit for every turns ratio. The turns ratio can be set at runtime and the PARAM function will take care of passing the parameter and calculating the correct values.

The .PARAM statement defines the value of a parameter.

- A parameter name can be used in place of most numeric values in the circuit description or passed into a subcircuit.
- Parameters can be constants, or expressions involving other parameters.
- .PARAM expressions may also take on the same form and features of analog behavioural element expressions including Inline equations and If-Then-Else statements.
- Name cannot begin with a number.
- The parameter values must be either constants or expressions.

- Curly braces are optional for constants or single parameters, but mandatory for all expressions.
- Expression can contain constants, parameters, or mathematical operators similar to the B element (see section 4.2.4).
- The .PARAM statements are order independent but parameter values must be completely defined such that all expressions can be evaluated to a resultant numeric value.
- A .PARAM statement can be used inside a subcircuit definition to establish local subcircuit parameters.

Parameters and parameterised equations can be used in just about any facet of the design including but not limited to: all numeric element properties (including transmission lines and polynomials), analysis statements (.AC, Tran, ICL), and independent sources (PWL, etc.).

WinSpice3 parameter passing syntax attempts to be compatible with the PSpice PARAMS:, .PARAM, and parameterised expression syntax.

The PARAM function evaluates expressions in the main circuit or in subcircuits using .PARAM statement variables, passed parameters or default parameters. Expressions may be as complex or as simple as desired. Several rules follow;

- Parameters defined in the main circuit file are applied to all subcircuits. Parameters defined in a subcircuit apply only within the subcircuit definition. Passed parameters override all other parameters of the same name.
- Expressions support the same operators and syntax used in the B element including mathematical and if-then-else expressions detailed in section 4.2.4. The resulting value is inserted using engineering notation.
- Recursive parameter values are not allowed, for example
  .Param N = N+1.
- You may pass unused parameters, however, each parameter which is used within an expression must be assigned a value or have a default value.
- Parameters passed into subcircuits must be accounted for with .PARAM statement(s), put on the subcircuit call line in curly braces or appear in the subcircuit's default listings.
- Expressions to be evaluated in the .PARAM statements, in a part's property field, or in a subcircuit listing must also be placed inside curly braces.
- Default parameters are placed on the .SUBCKT definition line. All of the parameters should have defaults.
- Parameters are available only within the subcircuit definition in which they appear. If a .PARAM is defined in the main netlist it is available in all subcircuits.
- Passed parameters will take precedence over default parameters.

Parameters or Expressions using parameters must appear within curly braces `` { } ``in order to be evaluated. For example;

```
.Subckt sub 1 2 PARAMS: Rval=1
Rval 1 2 {Rval}
.ends
```

In the above subcircuit the variable Rval within the curly braces will be substituted with a value of 1. The reference designator will be unaffected.

```
.Subckt sub 1 2 PARAMS: PARAM1=2u
X1 1 2 3 NextSub PARAMS: PARAM1 = {PARAM1}
.ENDS
```

In the above subcircuit, the variable PARAM1 within the curly braces will be substituted with 2u. The parameter PARAM1 for subcircuit NextSub will not be modified. Likewise,

```
.Subckt sub 1 2 PARAMS: PARAM1=2u
X1 1 2 3 NextSub {PARAM1 = {PARAM1}}
.ENDS
```

should produce the same results.

Local subcircuit parameters(PARAMS: or .PARAM) supersede global parameters (.PARAM parameters defined in the main netlist) of the same name. Expressions in the main circuit are treated the same as expressions in subcircuits.

In B elements parameterised expressions can be used inside of the behavioural equations. This allows you to mix parameters with circuit quantities like voltages, currents, and device power dissipations. For example:-

```
B1 1 0 V = {Tr}*v(tm1) + {Ts - Tr}*v(tm2)
```

3.6.3.2 Passing Parameters To Subcircuits

Subcircuit calling statement syntax:

Xname N1 N2 ... N# subname + {P1=val1 or expr1 ... Pj=valj or exprn}

where P1 through Pj are parameters passed to the subcircuit.

There are two ways to pass parameters into a subcircuit.

- Parameters may be defined with a .PARAM statement inside the .SUBCKT netlist.
- By stating the parameters on the subcircuit call line (X line).

The following forms are all valid.

```
x1 1 2 3 Subname {var1=expr var2=val2 ... varn = valn }
x1 1 2 3 Subname {var1=val1 var2=expr ... + varn = valn}
x1 1 2 3 Subname
+ {var1=val1 var2=val2 ... varn = valn }
x1 1 2 3 Subname PARAMS: var1=val1 ... varn = valn
x1 1 2 3 Subname PARAMS: var1=val1 var2=val2 ...
+ varn = {expr}
x1 1 2 3 Subname
+ PARAMS: var1=val1 var2={expr}... varn = valn
```

Note: A parameter can be a single parameter or a parameterised expression. However, the parameters must be previously defined in a .PARAM statement or in the subcircuit that the subcircuit call line is used in, so that a value can be passed to the subcircuit.

Parameters can be passed through multiple levels of a subcircuit's hierarchy. For example,

```
.PARAM Varmain = 1, Varmain2=1
.SUBCKT Subname 1 2 3
x1 1 2 3 Subname2 {var1=varmain }
.ENDS
.SUBCKT Subname2 1 2 3
x1 1 2 3 Subname3 {var2=var1 var3=varmain2}
.ENDS
```

Any number of variables can be accommodated.

3.6.3.3 Default Subcircuit Parameters

Default subcircuit parameters can be predefined on the subcircuit definition line. If a value is passed in by the calling X line, it will override the default value. Defaults can appear in curly braces on the .Subckt line or after the ``PARAMS:'' keyword.

Syntax:

```
.SUBCKT subname N1 ... N# {DP1=val1 ... DPj=valj}
.SUBCKT subname N1 ... N# {DP1=expr}
.SUBCKT subname N1 ... N# PARAMS: DP1=val1 ... DPj=valj
.SUBCKT subname N1 ... N# PARAMS: DP1=expr
```

where D1 through Dj are default parameters, val# is a valid SPICE number, and expr is a valid expression. Curly braces around an expression in the default list are optional.

PARAM: keywords can be placed anywhere in the subcircuit definition. However, all text after PARAM: up to the end of the line will be read in a parameters.

As an example, we will consider a semiconductor resistor subcircuit model. The subcircuit call is;

```
X1 1 2 RSUB {WIDTH=10U RPERSQ=1KOHMS}
```

The subcircuit contains;

```
.SUBCKT RSUB 1 2 {WIDTH=2U}
R1 1 2 {RPERSQ * (WIDTH^2)/1E-12}
.ENDS
```

The subcircuit call, X1, calls the subcircuit and passes two parameters, WIDTH and RPERSQ, into the subcircuit. The resistance value R1 will be calculated based on the equation which is shown next. All of the extended syntax is transformed into **WinSpice3** syntax by evaluating the expression(s) and then replacing each one with a value. For example;

```
X1 1 2 RSUB#0
.SUBCKT RSUB#0 1 2
R1 1 2 100.00K
.ENDS
```

After a simulation is run, the subcircuit names will have a sharp sign and a number appended to them in order to make them unique. If two RSUBs are called with different sets of parameters, then two different subcircuit representations will be created automatically. For example:

```
X1 1 2 RSUB {WIDTH=50U RPERSQ=100OHMS}
X2 3 4 RSUB {WIDTH=10U RPERSQ=1KOHMS}
```

will produce:

```
X1 1 2 RSUB#0
X2 3 4 RSUB#1
.SUBCKT RSUB#0 1 2
R1 1 2 250.00K
.ENDS
.SUBCKT RSUB#1 1 2
R1 1 2 100.00K
.ENDS
```

Each subcircuit call with a different parameter list will automatically create a new subcircuit. If all subcircuit calls use the same parameter list, only one subcircuit will be generated for all calls.

4 CIRCUIT ELEMENTS AND MODELS

Data fields that are enclosed in less-than and greater-than signs ('< >') are optional. All indicated punctuation (parentheses, equal signs, etc.) is optional but indicate the presence of any delimiter. Further, future implementations may require the punctuation as stated. A consistent style adhering to the punctuation shown here makes the input easier to understand. With respect to branch voltages and currents, **WinSpice3** uniformly uses the associated reference convention (current flows in the direction of voltage drop).

4.1 Elementary Devices

4.1.1 Rxxxx: Resistors

4.1.1.1 Simple Resistors

General form:

```
RXXXXXXX N1 N2 VALUE
```

Examples:

```
R1 1 2 100
RC1 12 17 1K
```

N1 and N2 are the two element nodes. VALUE is the resistance (in ohms) and may be positive or negative but not zero.

4.1.1.2 Semiconductor Resistors

General form:

```
RXXXXXXX N1 N2 <VALUE> <MNAME> <L=LENGTH> <W=WIDTH> <TEMP=T>
```

Examples:

```
RLOAD 2 10 10K
RMOD 3 7 RMODEL L=10u W=1u
```

This is the more general form of the resistor presented in section 4.1.1.1, and allows the modelling of temperature effects and for the calculation of the actual resistance value from strictly geometric information and the specifications of the process.

If VALUE is specified, it overrides the geometric information and defines the resistance. If MNAME is specified, then the resistance may be calculated from the process information in the model MNAME and the given LENGTH and WIDTH. If VALUE is not specified, then MNAME and LENGTH must be specified. If WIDTH is not specified, then it is taken from the default width given in the model. The (optional) TEMP value is the temperature at which this device is to operate, and overrides the temperature specification on the .OPTION control line.

4.1.1.3 Semiconductor Resistor Model (R)

The resistor model consists of process-related device data that allow the resistance to be calculated from geometric information and to be corrected for temperature.

The parameters available are:

name	parameter	units	default	example
TC1	first order temperature coefficient	Ω/°C	0.0	-
TC2	second order temperature coefficient.	Ω/°C^2	0.0	-
RSH	sheet resistance	Ω/square	-	50
DEFW	default width	meters	1e-6	2e-6
NARROW	narrowing due to side etching	meters	0.0	1e-7
TNOM	parameter measurement temperature	°C	27	50

The sheet resistance is used with the narrowing parameter and L and W from the resistor device to determine the nominal resistance by the formula

$$R = RSH\frac{L - NARROW}{W - NARROW}$$

DEFW is used to supply a default value for W if one is not specified for the device. If either RSH or L is not specified, then the standard default resistance value of 1k Z is used.

TNOM is used to override the circuit-wide value given on the .OPTIONS control line where the parameters of this model have been measured at a different temperature. After the nominal resistance is calculated, it is adjusted for temperature by the formula:

$$R(T) = R(T_0)\left[1 + TC_1(T - T_0) + TC_2(T - T_0)^2\right]$$

4.1.2 Cxxxx: Capacitors

4.1.2.1 Simple Capacitors

General form:

```
CXXXXXXX N+ N- VALUE <IC=INCOND>
```

Examples:

```
CBYP 13 0 1UF
COSC 17 23 10U IC=3V
```

N+ and N- are the positive and negative element nodes, respectively. VALUE is the capacitance in Farads.

The (optional) initial condition is the initial (time- zero) value of capacitor voltage (in Volts). Note that the initial conditions (if any) apply 'only' if the UIC option is specified on the .TRAN control line.

NOTE: unlike Spice2, non-linear capacitors using POLY are not directly supported by WinSpice3. However, they can be simulated using non-linear current and voltage sources. Voltage and temperature dependent capacitance can be simulated using the capacitor model described in section 4.1.2.3.

4.1.2.2 Semiconductor Capacitors

General form:

```
CXXXXXXX N1 N2 <VALUE> <MNAME> <L=LENGTH> <W=WIDTH> <IC=VAL>
```

Examples:

```
CLOAD 2 10 10P
CMOD 3 7 CMODEL L=10u W=1u
```

This is the more general form of the Capacitor presented in section 4.1.2.1, and allows for the calculation of the actual capacitance value from strictly geometric information and the specifications of the process.

If VALUE is specified, it defines the capacitance. If MNAME is specified, then the capacitance is calculated from the process information in the model MNAME and the given LENGTH and WIDTH. If VALUE is not specified, then MNAME and LENGTH must be specified. If WIDTH is not specified, then it is taken from the default width given in the model. Either VALUE or MNAME, LENGTH, and WIDTH may be specified, but not both sets.

4.1.2.3 Semiconductor Capacitor Model (C)

The capacitor model contains process information that may be used to compute the capacitance from strictly geometric information.

name	parameter	units	default	example
TNOM	parameter measurement temperature	°C	27	50
TC1	first order temperature coefficient	Ω/°C	0.0	-
TC2	second order temperature coefficient.	$\Omega/°C^2$	0.0	-
VC1	first order voltage coefficient	$volt^{-1}$	0.0	-
VC2	second order voltage coefficient.	$volt^{-2}$	0.0	-
CJ	junction bottom capacitance	$F/meters^2$	-	5e-5
CJSW	junction side wall capacitance	F/meters	-	2e-11
DEFW	default device width	meters	1e-6	2e-6
NARROW	narrowing due to side etching	meters	0.0	1e-7

The capacitor has a capacitance computed as

$$CAP = CJ(LENGTH - NARROW)(WIDTH - NARROW) + 2CJSW(LENGTH + WIDTH - 2NARROW)$$

TNOM is used to override the circuit-wide value given on the .OPTIONS control line where the parameters of this model have been measured at a different temperature.

After the nominal capacitance is calculated above, it is adjusted for temperature and voltage nonlinearity by the formula:-

$$C_{eff} = CAP(1 + VC_1 \cdot V_{cap} + VC_2 \cdot V_{cap}{}^2)(1 + TC_1(T - T_{nom}) + TC_2(T - T_{nom})^2)$$

4.1.3 Lxxxx: Inductors

General form:

```
LYYYYYYY N+ N- VALUE <IC=INCOND>
```

Examples:

```
LLINK 42 69 1UH
LSHUNT 23 51 10U IC=15.7MA
```

N+ and N- are the positive and negative element nodes, respectively. VALUE is the inductance in Henries.

The (optional) initial condition is the initial (time-zero) value of inductor current (in Amps) that flows from N+, through the inductor, to N-. Note that the initial conditions (if any) apply only if the UIC option is specified on the .TRAN analysis line.

NOTE: unlike Spice2, non-linear inductors are not directly supported by WinSpice3. However, they can be simulated using non-linear current and voltage sources.

4.1.4 Kxxxx: Coupled (Mutual) Inductors

General form:

```
KXXXXXXX LYYYYYYY LZZZZZZZ VALUE
```

Examples:

```
K43 LAA LBB 0.999
KXFRMR L1 L2 0.87
```

LYYYYYYY and LZZZZZZZ are the names of the two coupled inductors, and VALUE is the coefficient of coupling, K, which must be greater than 0 and less than or equal to 1. Using the 'dot' convention, place a 'dot' on the first node of each inductor.

4.1.5 Sxxxx and Wxxxx: Switches

4.1.5.1 Sxxxx: Voltage Controlled Switch

General form:

```
SXXXXXXX N+ N- NC+ NC- MODEL <ON><OFF>
```

Examples:

```
s1 1 2 3 4 switch1 ON
s2 5 6 3 0 sm2 off
Switch1 1 2 10 0 smodel1
```

Nodes 1 and 2 are the nodes between which the switch terminals are connected. The model name is mandatory while the initial conditions are optional. Nodes 3 and 4 are the positive and negative controlling nodes respectively.

4.1.5.2 Wxxxx: Current Controlled Switch

General form:

```
WYYYYYYY N+ N- VNAM MODEL <ON><OFF>
```

Examples:

```
w1 1 2 vclock switchmod1
W2 3 0 vramp sm1 ON
wreset 5 6 vclck lossyswitch OFF
```

Nodes 1 and 2 are the nodes between which the switch terminals are connected. The model name is mandatory while the initial conditions are optional. The controlling current is that through the specified voltage source. The direction of positive controlling current flow is from the positive node, through the source, to the negative node.

4.1.5.3 Switch Model (SW/CSW)

General form:

```
.MODEL MNAME TYPE(PNAME1=PVAL1 PNAME2=PVAL2 ... )
```

Examples:

```
.MODEL SMOD SW(RON=5M ROFF=10E9 VT=1.0 VH=0.1)
.MODEL SMOD VSWITCH(RON=5M ROFF=10E9 VON=1.1 VOFF=0.9)
.MODEL SMOD CSW(RON=5M ROFF=10E9 IT=0.5MA IH=0.5MA)
.MODEL SMOD ISWITCH(RON=5M ROFF=10E9 ION=1.0MA IOFF=0)
```

The VSWITCH and ISWITCH forms of the model show above are provided for compatibility with PSPICE.

The switch model allows an almost ideal switch to be described in **WinSpice3**. The switch is not quite ideal, in that the resistance can not change from 0 to infinity, but must always have a finite positive value. By proper selection of the on and off resistances, they can be effectively zero and infinity in comparison to other circuit elements.

The parameters available are:

name	parameter	units	default	switch
VT	threshold voltage	Volts	0.0	S
VH	hysteresis voltage	Volts	0.0	S
VON	threshold voltage	Volts	0.0	S
VOFF	threshold voltage	Volts	0.0	S
IT	threshold current	Amps	0.0	W
IH	hysteresis current	Amps	0.0	W
ION	threshold current	Amps	0.0	W
IOFF	threshold current	Amps	0.0	W
RON	on resistance	Ω	1.0	both
ROFF	off resistance	Ω	1/GMIN *	both

*(See the .OPTIONS control line for a description of GMIN, its default value results in an off-resistance of 1.0e+12 ohms.)

For the voltage controlled switch, the switch is in the ON state if

$$V_{ctrl} > (VT + VH)$$

if VT and VH are defined or

$$V_{ctrl} > (VON)$$

if VON is defined. It is in the OFF state if

$$V_{ctrl} < (VT - VH)$$

if VT and VH are defined or

$$V_{ctrl} < (VOFF)$$

if VOFF is defined.

For the current controlled switch, the switch is in the ON state if

$$I_{ctrl} > (IT + IH)$$

if IT and IH are defined or

$$I_{ctrl} > (ION)$$

if ION is defined.

and is in the OFF state if

$$I_{ctrl} < (IT - IH)$$

if IT and IH are defined or

$$I_{ctrl} < (IOFF)$$

if IOFF is defined.

The use of an ideal element that is highly non-linear such as a switch can cause large discontinuities to occur in the circuit node voltages. A rapid change such as that associated with a switch changing state can cause numerical roundoff or tolerance problems leading to erroneous results or timestep difficulties. The user of switches can improve the situation by taking the following steps:

First, it is wise to set ideal switch impedances just high or low enough to be negligible with respect to other circuit elements. Using switch impedances that are close to "ideal" in all cases aggravates the problem of discontinuities mentioned above. Of course, when modelling real devices such as MOSFETs, the on resistance should be adjusted to a realistic level depending on the size of the device being modelled.

If a wide range of ON to OFF resistance must be used in the switches (ROFF/RON > 1e+12), then the tolerance on errors allowed during transient analysis should be decreased by using the .OPTIONS control line and specifying TRTOL to be less than the default value of 7.0. When switches are placed around capacitors, then the option CHGTOL should also be reduced. Suggested values for these two options are 1.0 and 1e-16 respectively. These changes inform **WinSpice3** to be more careful around the switch points so that no errors are made due to the rapid change in the circuit.

4.2 Voltage And Current Sources

4.2.1 Ixxxx and Vxxxx: Independent Sources

General form:

```
VXXXXXXX N+ N- <<DC> DC/TRAN VALUE> <AC <ACMAG <ACPHASE>>>
+   <DISTOF1 <F1MAG <F1PHASE>>> <DISTOF2 <F2MAG <F2PHASE>>>
IYYYYYYY N+ N- <<DC> DC/TRAN VALUE> <AC <ACMAG <ACPHASE>>>
+   <DISTOF1 <F1MAG <F1PHASE>>> <DISTOF2 <F2MAG <F2PHASE>>>
```

Examples:

```
VCC 10 0 DC 6
VIN 13 2 0.001 AC 1 SIN(0 1 1MEG)
ISRC 23 21 AC 0.333 45.0 SFFM(0 1 10K 5 1K)
VMEAS 12 9
VCARRIER 1 0 DISTOF1 0.1 -90.0
VMODULATOR 2 0 DISTOF2 0.01
IIN1 1 5 AC 1 DISTOF1 DISTOF2 0.001
```

N+ and N- are the positive and negative nodes, respectively. Note that voltage sources need not be grounded. Positive current is assumed to flow from the positive node, through the source, to the negative node. A current source of positive value forces current to flow out of the N+ node, through the source, and into the N- node. Voltage sources, in addition to being used for circuit excitation, are the 'ammeters' for **WinSpice3**, that is, zero valued voltage sources may be inserted into the circuit for the purpose of measuring current. They of course have no effect on circuit operation since they represent short-circuits.

DC/TRAN is the DC and transient analysis value of the source. If the source value is zero both for DC and transient analyses, this value may be omitted. If the source value is time-invariant (e.g., a power supply), then the value may optionally be preceded by the letters DC.

ACMAG is the AC magnitude and ACPHASE is the AC phase. The source is set to this value in the AC analysis. If ACMAG is omitted following the keyword AC, a value of unity is assumed. If ACPHASE is omitted, a value of zero is assumed. If the source is not an AC small-signal input, the keyword AC and the AC values are omitted.

DISTOF1 and DISTOF2 are the keywords that specify that the independent source has distortion inputs at the frequencies F1 and F2 respectively (see the description of the .DISTO control line). An optional magnitude and phase may follow the keywords. The default values of the magnitude and phase are 1.0 and 0.0 respectively.

Any independent source can be assigned a time-dependent value for transient. If a source is assigned a time-dependent value, the time-zero value is used for DC analysis. There are five independent source functions: pulse (PULSE), exponential (EXP), sinusoidal (SIN), piece-wise linear (PWL), and single-frequency FM (SFFM). If parameters other than source values are omitted or set to zero, the default values shown are assumed. In the descriptions below, TSTEP is the printing increment and TSTOP is the final time (see the .TRAN control line for explanation – section 5.3.9).

See sections 10.26 and 10.13 for parameters that can be altered using the ‘alter’ command (see 6.9.3).

4.2.1.1 PULSE(): Pulse

General form:

```
PULSE(V1 V2 TD TR TF PW PER)
```

Examples:

```
VIN 3 0 PULSE(-1 1 2NS 2NS 2NS 50NS 100NS)
```

parameter	default value	units
V1 (initial value)		Volts or Amps
V2 (pulsed value)		Volts or Amps
TD (delay time)	0.0	seconds
TR (rise time)	TSTEP	seconds
TF (fall time)	TSTEP	seconds
PW (pulse width)	TSTOP	seconds
PER(period)	TSTOP	seconds

The following table describes a single pulse so specified:

Time	value
0	V1
TD	V1
TD+TR	V2
TD+TR+PW	V2
TD+TR+PW+TF	V1
TSTOP	V1

Intermediate points are determined by linear interpolation.

4.2.1.2 SIN(): Sinusoidal

General form:

```
SIN(VO VA FREQ TD THETA)
```

Examples:

```
VIN 3 0 SIN(0 1 100MEG 1NS 1E10)
```

parameters	default value	units
VO (offset)		Volts or Amps
VA (amplitude)		Volts or Amps
FREQ (frequency)	1/TSTOP	Hz
TD (delay)	0.0	seconds
THETA (damping factor)	0.0	1/seconds

The following table describes the shape of the waveform:

time	value
0 to TD	VO
TD to TSTOP	$VO + VAe^{-(t-TD)THETA}\sin\left(2\pi FREQ(t+TD)\right)$

4.2.1.3 EXP(): Exponential

General Form:

```
EXP(V1 V2 TD1 TAU1 TD2 TAU2)
```

Examples:

```
VIN 3 0 EXP(-4 -1 2NS 30NS 60NS 40NS)
```

parameter	default value	units
V1 (initial value)		Volts or Amps
V2 (pulsed value)		Volts or Amps
TD1 (rise delay time)	0.0	seconds
TAU1 (rise time constant)	TSTEP	seconds
TD2 (fall delay time)	TD1+TSTEP	seconds
TAU2 (fall time constant)	TSTEP	seconds

The following table describes the shape of the waveform:

time	value
0 to TD1	$V1$
TD1 to TD2	$V1+(V2-V1)\left(1-e^{\frac{-(t-TD1)}{TAU1}}\right)$
TD2 to TSTOP	$V1+(V2-V1)\left(-e^{\frac{-(t-TD1)}{TAU1}}\right)+(V1-V2)\left(1-e^{\frac{-(t-TD2)}{TAU2}}\right)$

4.2.1.4 PWL(): Piece-Wise Linear

General Form:

```
PWL(T1 V1 <T2 V2 T3 V3 T4 V4 ...>)
```

Examples:

```
VCLOCK 7 5 PWL(0 -7 10NS -7 11NS -3 17NS -3 18NS -7 50NS -7)
```

Each pair of values (Ti, Vi) specifies that the value of the source is Vi (in Volts or Amps) at time=Ti. The value of the source at intermediate values of time is determined by using linear interpolation on the input values.

4.2.1.5 SFFM(): Single-Frequency FM

General Form:

```
SFFM(VO VA FC MDI FS)
```

Examples:

```
V1 12 0 SFFM(0 1M 20K 5 1K)
```

parameter	default value	units
VO (offset)		Volts or Amps
VA (amplitude)		Volts or Amps
FC (carrier frequency)	1/TSTOP	Hz
MDI (modulation index)		
FS (signal frequency)	1/TSTOP	Hz

The shape of the waveform is described by the following equation:

$$V(t) = V_0 + V_A \sin\left(2\pi FCt + MDI \sin\left(2\pi FSt\right)\right)$$

4.2.2 Linear Dependent Sources

SPICE allows circuits to contain linear dependent sources characterised by any of the four equations

$i = g\,v \quad v = e\,v \quad i = f\,i \quad v = h\,i$

where g, e, f, and h are constants representing transconductance, voltage gain, current gain, and transresistance, respectively.

4.2.2.1 Gxxxx: Linear Voltage-Controlled Current Sources

General form:

```
GXXXXXXX N+ N- NC+ NC- VALUE
```

Examples:

```
G1 2 0 5 0 0.1MMHO
```

N+ and N- are the positive and negative nodes, respectively. Current flow is from the positive node, through the source, to the negative node. NC+ and NC- are the positive and negative controlling nodes, respectively. VALUE is the transconductance (in mhos).

4.2.2.2 Exxxx: Linear Voltage-Controlled Voltage Sources

General form:

```
EXXXXXXX N+ N- NC+ NC- VALUE
```

Examples:

```
E1 2 3 14 1 2.0
```

N+ is the positive node, and N- is the negative node. NC+ and NC- are the positive and negative controlling nodes, respectively. VALUE is the voltage gain.

4.2.2.3 Fxxxx: Linear Current-Controlled Current Sources

General form:

```
FXXXXXXX N+ N- VNAM VALUE
```

Examples:

```
F1 13 5 VSENS 5
```

N+ and N- are the positive and negative nodes, respectively. Current flow is from the positive node, through the source, to the negative node. VNAM is the name of a voltage source through which the controlling current flows. The direction of positive controlling current flow is from the positive node, through the source, to the negative node of VNAM. VALUE is the current gain.

4.2.2.4 Hxxxx: Linear Current-Controlled Voltage Sources

General form:

```
HXXXXXXX N+ N- VNAM VALUE
```

Examples:

```
HX 5 17 VZ 0.5K
```

N+ and N- are the positive and negative nodes, respectively. VNAM is the name of a voltage source through which the controlling current flows. The direction of positive controlling current flow is from the positive node, through the source, to the negative node of VNAM. VALUE is the transresistance (in ohms).

4.2.3 Non-linear Dependent Sources using POLY()

For compatibility with SPICE2, **WinSpice** allows circuits to contain dependent sources characterised by any of the four equations

i=f(v) v=f(v) i=f(i) v=f(i)

where the functions must be polynomials, and the arguments may be multidimensional. The polynomial functions are specified by a set of coefficients p0, p1, ..., pn. Both the number of dimensions and the number of coefficients are arbitrary. The meaning of the coefficients depends upon the dimension of the polynomial, as shown in the following examples:

Suppose that the function is one-dimensional (that is, a function of one argument). Then the function value fv is determined by the following expression in fa (the function argument):

fv = p0 + (p1*fa) + (p2*fa**2) + (p3*fa**3) + (p4*fa**4) + (p5*fa**5) + ...

Suppose now that the function is two-dimensional, with arguments fa and fb. Then the function value fv is determined by the following expression:

```
fv = p0 + (p1*fa) + (p2*fb) + (p3*fa**2) + (p4*fa*fb)
     + (p5*fb**2)
     + (p6*fa**3) + (p7*fa**2*fb) + (p8*fa*fb**2)
     + (p9*fb**3) + ...
```

Consider now the case of a three-dimensional polynomial function with arguments fa, fb, and fc. Then the function value fv is determined by the following expression:

```
fv = p0  +  (p1*fa)  +  (p2*fb)  +  (p3*fc)  +  (p4*fa**2)
     + (p5*fa*fb)
     + (p6*fa*fc) + (p7*fb**2) + (p8*fb*fc)  +  (p9*fc**2)
     + (p10*fa**3)
     + (p11*fa**2*fb) + (p12*fa**2*fc) + (p13*fa*fb**2)
     + (p14*fa*fb*fc)
     + (p15*fa*fc**2)  +  (p16*fb**3)  +  (p17*fb**2*fc)
     + (p18*fb*fc**2)
     + (p19*fc**3) + (p20*fa**4) + ...
```

Note: if the polynomial is one-dimensional and exactly one coefficient is specified, then SPICE assumes it to be p1 (and p0 = 0.0), in order to facilitate the input of linear controlled sources.

For all four of the dependent sources described below, the initial condition parameter is described as optional. If not specified, WinSpice assumes 0 the initial condition for dependent sources is an initial 'guess' for the value of the controlling variable. The program uses this initial condition to obtain the dc operating point of the circuit. After convergence has been obtained, the program continues iterating to obtain the exact value for the controlling variable. Hence, to reduce the computational effort for the dc operating point, or if the polynomial specifies a strong nonlinearity, a value fairly close to the actual controlling variable should be specified for the initial condition.

4.2.3.1 Voltage-Controlled Current Sources

General form:

```
GXXXXXXX N+ N- <POLY(ND)> NC1+ NC1- ... P0 <P1 ...> <IC=...>
```

Examples:

```
G1 1 0 5 3 0 0.1M
GR 17 3 17 3 0 1M 1.5M IC=2V
GMLT 23 17 POLY(2) 3 5 1 2 0 1M 17M 3.5U IC=2.5, 1.3
```

N+ and N- are the positive and negative nodes, respectively. Current flow is from the positive node, through the source, to the negative node. POLY(ND) only has to be specified if the source is multi-dimensional (one-dimensional is the default). If specified, ND is the number of dimensions, which must be positive. NC1+, NC1-, .are the positive and negative controlling nodes, respectively. One pair of nodes must be specified for each dimension. P0, P1, P2, ..., Pn are the polynomial coefficients. The (optional) initial condition is the initial guess at the value(s) of the controlling voltage(s). If not specified, 0.0 is assumed. The polynomial specifies the source current as a function of the controlling voltage(s). The second example above describes a current source with value

I = 1E-3*V(17,3) + 1.5E-3*V(17,3)**2

note that since the source nodes are the same as the controlling nodes, this source actually models a nonlinear resistor.

4.2.3.2 Voltage-Controlled Voltage Sources

General form:

```
EXXXXXXX N+ N- <POLY(ND)> NC1+ NC1- ... P0 <P1 ...> <IC=...>
```

Examples:

```
E1 3 4 21 17 10.5 2.1 1.75
EX 17 0 POLY(3) 13 0 15 0 17 0 0 1 1 1 IC=1.5,2.0,17.35
```

N+ and N- are the positive and negative nodes, respectively. POLY(ND) only has to be specified if the source is multi-dimensional (one-dimensional is the default). If specified, ND is the number of dimensions, which must be positive. NC1+, NC1-, ... are the positive and negative controlling nodes, respectively. One pair of nodes must be specified for each dimension.

P0, P1, P2, ..., Pn are the polynomial coefficients. The optional initial condition is the initial guess at the value(s) of the controlling voltage(s). If not specified, 0.0 is assumed.

The polynomial specifies the source voltage as a function of the controlling voltage(s). The second example above describes a voltage source with value

V = V(13,0) + V(15,0) + V(17,0)

(in other words, an ideal voltage summer).

4.2.3.3 Current-Controlled Current Sources

General form:

```
FXXXXXXX N+ N- <POLY(ND)> VN1 <VN2 ...> P0 <P1 ...> <IC=...>
```

Examples:

```
F1 12 10 VCC 1MA 1.3M
FXFER 13 20 VSENS 0 1
```

N+ and N- are the positive and negative nodes, respectively. Current flow is from the positive node, through the source, to the negative node. POLY(ND) only has to be specified if the source is multi-dimensional (one-dimensional is the default). If specified, ND is the number of dimensions, which must be positive. VN1, VN2, ... are the names of voltage sources through which the controlling current flows; one name must be specified for each dimension. The direction of positive controlling current flow is from the positive node, through the source, to the negative node of each voltage source. P0, P1, P2, ..., Pn are the polynomial coefficients. The (optional) initial condition is the initial guess at the value(s) of the controlling current(s) (in Amps). If not specified, 0.0 is assumed. The polynomial specifies the source current as a function of the controlling current(s). The first example above describes a current source with value

I = 1E-3 + 1.3E-3*I(VCC)

4.2.3.4 Current-Controlled Voltage Sources

General form:

```
HXXXXXXX N+ N- <POLY(ND)> VN1 <VN2 ...> P0 <P1 ...> <IC=...>
```

Examples:

```
HXY 13 20 POLY(2) VIN1 VIN2 0 0 0 0 1 IC=0.5 1.3
HR 4 17 VX 0 0 1
```

N+ and N- are the positive and negative nodes, respectively. POLY(ND) only has to be specified if the source is multi-dimensional (one-dimensional is the default). If specified, ND is the number of dimensions, which must be positive. VN1, VN2, ... are the names of voltage sources through which the controlling current flows; one name must be specified for each dimension. The direction of positive controlling current flow is from the positive node, through the source, to the negative node of each voltage source. P0, P1, P2, ..., Pn are the polynomial coefficients. The optional initial condition is the initial guess at the value(s) of the controlling current(s) (in Amps). If not specified, 0.0 is assumed. The polynomial specifies the source voltage as a function of the controlling current(s). The first example above describes a voltage source with value

V = I(VIN1)*I(VIN2)

4.2.4 Bxxxx: Non-linear Dependent Sources

General form:

```
BXXXXXXX N+ N- <I=EXPR> <V=EXPR>
```

Examples:

```
B1 0 1 I=cos(v(1))+sin(v(2))
B1 0 1 V=ln(cos(log(v(1,2)^2)))-v(3)^4+v(2)^v(1)
B1 3 4 I=17
B1 3 4 V=exp(pi^i(vdd))
```

N+ is the positive node, and N- is the negative node. The values of the V and I parameters determine the voltages and currents across and through the device, respectively. If I is given then the device is a current source, and if V is given the device is a voltage source. One and only one of these parameters must be given.

The small-signal AC behaviour of the non-linear source is a linear dependent source (or sources) with a proportionality constant equal to the derivative (or derivatives) of the source at the DC operating point.

The expressions given for V and I may be any function of voltages and currents through voltage sources in the system. In an AC analysis, only the DC component of a voltage or current source when the initial operating point was calculated is used. The following functions of real variables are defined:

Function	Description
abs(x)	The absolute value of x.
acos(x)	acos(x) in radians
acosh(x)	acosh(x) in radians
asin(x)	asin(x) in radians
asinh(x)	asinh(x) in radians
atan(x)	atan(x) in radians
atanh(x)	atanh(x) in radians
cos(x)	cos(x) (x in radians)
cosh(x)	cosh(x) (x in radians)
exp(x)	Returns e^x
expl(x,y)	Returns e^x with a limit y. If $e^x < y$ return e^x else return y
ln(x)	log base e of x
log(x)	log base 10 of x
sgn(x)	Signum function. If x >= 0, returns 1, else returns -1.

sin(x)	sin(x) (x in radians)
sinh(x)	sinh(x) (x in radians)
sqrt(x)	Square root of x
tan(x)	tan(x) (x in radians)
u(x)	The unit step function, with a value of one for arguments greater than one and a value of zero for arguments less than zero.
uramp(x)	The integral of the unit step: for an input x, the value is zero if x is less than zero, or if x is greater than zero the value is x.

Table 1: Functions

The u() and uramp() functions are useful in synthesising piece-wise non-linear functions, though convergence may be adversely affected.

The following standard operators are defined (precedence decreasing down the table):-

Operator	**Meaning**
( )	Parentheses
unary - unary +	Negation
^	Exponentiation (x^y -> x^y)
*, /	Multiplication and division
+, -	Addition and subtraction
<, <=, >, >=	
==, !=	Equivalence
&, &&	Logical AND
|, ||	Logical OR
? :	If-Then-Else

Table 2: Operator Precedence

Note here that the If-Then-Else operator found in the C programming language is available here. This greatly increases the capabilities of the B line by allowing very complex equations to be defined. For example:-

```
B1 3 4 V= (v(1) >= 1V) ? 5V : 0V
```

defined a voltage source which outputs 5V if the voltage at node 1 is greater than or equal to 1V and 0V otherwise.

If the argument of log(), ln(), or sqrt() becomes less than zero, the absolute value of the argument is used. If a divisor becomes zero or the argument of log() or ln() becomes zero, an error will result. Other problems may occur when the argument for a function in a partial derivative enters a region where that function is undefined.

To get time into the expression you can integrate the current from a constant current source with a capacitor and use the resulting voltage (don't forget to set the initial voltage across the capacitor). Non-linear resistors, capacitors, and inductors may be synthesised with the non-linear dependent source. Non-linear resistors are obvious. Non-linear capacitors and inductors are implemented with their linear counterparts by a change of variables implemented with the non-linear dependent source. The following subcircuit will implement a non-linear capacitor:

```
.Subckt nlcap   pos neg
* Bx: calculate f(input voltage)
Bx   1    0    v = f(v(pos,neg))
* Cx: linear capacitance
Cx   2    0    1
* Vx: Ammeter to measure current into the capacitor
Vx   2    1    DC 0Volts
* Drive the current through Cx back into the circuit
Fx   pos  neg  Vx 1
.ends
```

Non-linear inductors are similar.

4.3 Transmission Lines

4.3.1 Txxxx: Lossless Transmission Lines

General form:

```
TXXXXXXX N1 N2 N3 N4 Z0=VALUE <TD=VALUE> <F=FREQ <NL=NRMLEN>>
+        <IC=V1, I1, V2, I2>
```

Examples:

```
T1 1 0 2 0 Z0=50 TD=10NS
```

N1 and N2 are the nodes at port 1; N3 and N4 are the nodes at port 2. Z0 is the characteristic impedance. The length of the line may be expressed in either of two forms. The transmission delay, TD, may be specified directly (as TD=10ns, for example). Alternatively, a frequency F may be given, together with NL, the normalised electrical length of the transmission line with respect to the wavelength in the line at the frequency F. If a frequency is specified but NL is omitted, 0.25 is assumed (that is, the frequency is assumed to be the quarter-wave frequency). Note that although both forms for expressing the line length are indicated as optional, one of the two must be specified.

Note that this element models only one propagating mode. If all four nodes are distinct in the actual circuit, then two modes may be excited. To simulate such a situation, two transmission-line elements are required (see the example in Appendix A for further clarification).

The (optional) initial condition specification consists of the voltage and current at each of the transmission line ports. Note that the initial conditions (if any) apply 'only' if the UIC option is specified on the .TRAN control line.

Note that a lossy transmission line (see below) with zero loss may be more accurate than the lossless transmission line due to implementation details.

4.3.2 Oxxxx: Lossy Transmission Lines

General form:

```
OXXXXXXX N1 N2 N3 N4 MNAME
```

Examples:

```
O23 1 0 2 0 LOSSYMOD
OCONNECT 10 5 20 5 INTERCONNECT
```

This is a two-port convolution model for single-conductor lossy transmission lines. N1 and N2 are the nodes at port 1; N3 and N4 are the nodes at port 2. Note that a lossy transmission line with zero loss may be more accurate than the lossless transmission line due to implementation details.

4.3.2.1 Lossy Transmission Line Model (LTRA)

The uniform RLC/RC/LC/RG transmission line model (referred to as the LTRA model henceforth) models a uniform constant-parameter distributed transmission line. The RC and LC cases may also be modelled using the URC and TRA models; however, the newer LTRA model is usually faster and more accurate than the others are. The operation of the LTRA model is based on the convolution of the transmission line's impulse responses with its inputs (see [8]).

The LTRA model takes a number of parameters, some of which must be given and others that are optional.

name	parameter	units/type	default	example
R	resistance/length	Z/unit	0.0	0.2
L	inductance/length	henrys/unit	0.0	9.13e-9
G	conductance/length	mhos/unit	0.0	0.0
C	capacitance/length	farads/unit	0.0	3.65e-12
LEN	length of line		no default	1.0
REL	breakpoint control	arbitrary unit	1	0.5
ABS	breakpoint control		1	5
NOSTEPLIMIT	don't limit timestep to less than line delay	flag	not set	set
NOCONTROL	don't do complex timestep control	flag	not set	set
LININTERP	use linear interpolation	flag	not set	set
MIXEDINTERP	use linear when quadratic seems bad		not set	set
COMPACTREL	special reltol for history compaction	flag	RELTOL	1.0e-3
COMPACTABS	special abstol for history compaction		ABSTOL	1.0e-9
TRUNCNR	use Newton-Raphson method for timestep control	flag	not set	set
TRUNCDONTCUT	don't limit timestep to keep impulse-response errors low	flag	not set	set

The following types of lines have been implemented so far: RLC (uniform transmission line with series loss only), RC (uniform RC line), LC (lossless transmission line), and RG (distributed series resistance and parallel conductance only). Any other combination will yield erroneous results and should not be tried. The length LEN of the line must be specified.

NOSTEPLIMIT is a flag that will remove the default restriction of limiting time-steps to less than the line delay in the RLC case. NOCONTROL is a flag that prevents the default limiting of the time-step based on convolution error criteria in the RLC and RC cases. This speeds up simulation but may in some cases reduce the accuracy of results.

LININTERP is a flag that, when specified, will use linear interpolation instead of the default quadratic interpolation for calculating delayed signals.

MIXEDINTERP is a flag that, when specified, uses a metric for judging whether quadratic interpolation is not applicable and if so uses linear interpolation; otherwise it uses the default quadratic interpolation.

TRUNCDONTCUT is a flag that removes the default cutting of the time-step to limit errors in the actual calculation of impulse-response related quantities.

COMPACTREL and COMPACTABS are quantities that control the compaction of the past history of values stored for convolution. Larger values of these lower accuracy but usually increase simulation speed. These are to be used with the TRYTOCOMPACT option, described in the .OPTIONS section.

TRUNCNR is a flag that turns on the use of Newton-Raphson iterations to determine an appropriate timestep in the timestep control routines. The default is a trial and error procedure by cutting the previous timestep in half.

REL and ABS are quantities that control the setting of breakpoints.

The option most worth experimenting with for increasing the speed of simulation is REL. The default value of 1 is usually safe from the point of view of accuracy but occasionally increases computation time. A value greater than 2 eliminates all breakpoints and may be worth trying depending on the nature of the rest of the circuit, keeping in mind that it might not be safe from the viewpoint of accuracy. Breakpoints may usually be entirely eliminated if it is expected the circuit will not display sharp discontinuities. Values between 0 and 1 are usually not required but may be used for setting many breakpoints.

COMPACTREL may also be experimented with when the option TRYTOCOMPACT is specified in a .OPTIONS card. The legal range is between 0 and 1. Larger values usually decrease the accuracy of the simulation but in some cases improve speed. If TRYTOCOMPACT is not specified on a .OPTIONS card, history compaction is not attempted and accuracy is high. NOCONTROL, TRUNCDONTCUT and NOSTEPLIMIT also tend to increase speed at the expense of accuracy.

4.3.3 Uxxxx: Uniform Distributed RC Lines (Lossy)

General form:

```
UXXXXXXX N1 N2 N3 MNAME L=LEN <N=LUMPS>
```

Examples:

```
U1 1 2 0 URCMOD L=50U
URC2 1 12 2 UMODL l=1MIL N=6
```

N1 and N2 are the two element nodes the RC line connects, while N3 is the node to which the capacitances are connected. MNAME is the model name, LEN is the length of the RC line in meters. LUMPS, if specified, is the number of lumped segments to use in modelling the RC line (see the model description for the action taken if this parameter is omitted).

4.3.3.1 Uniform Distributed RC Model (URC)

The URC model is derived from a model proposed by L. Gertzberrg in 1974. The model is accomplished by a subcircuit type expansion of the URC line into a network of lumped RC segments with internally generated nodes. The RC segments are in a geometric progression, increasing toward the middle of the URC line, with K as proportionality constant. The number of lumped segments used, if not specified for the URC line device, is determined by the following formula:

$$N = \frac{\log\left[F_{\max} \frac{R}{L}\frac{C}{L} 2\pi L^2 \left(\frac{(K-1)}{K} \right)^2 \right]}{\log K}$$

The URC line is made up strictly of resistor and capacitor segments unless the ISPERL parameter is given a non-zero value. In this case the capacitors are replaced with reverse biased diodes. These have a zero-bias junction capacitance equivalent to the capacitance replaced, and with a saturation current of ISPERL amps per meter of transmission line, and an optional series resistance equivalent to RSPERL ohms per meter.

name	parameter	units	default	example	area
K	Propagation Constant	-	2.0	1.2	-
FMAX	Maximum Frequency of interest	Hz	1.0G	6.5Meg	-
RPERL	Resistance per unit length	Ω/m	1000	10	-
CPERL	Capacitance per unit length	F/m	1.0e-15	1pF	-
ISPERL	Saturation Current per unit length	A/m	0	-	-
RSPERL	Diode Resistance per unit length	Ω/m	0	-	-

4.4 Transistors And Diodes

WinSpice3 has built-in models for the semiconductor devices, and the user need specify only the pertinent model parameter values.

The model for the BJT is based on the integral-charge model of Gummel and Poon; however, if the Gummel-Poon parameters are not specified, the model reduces to the simpler Ebers-Moll model. In either case, charge-storage effects, ohmic resistances, and a current-dependent output conductance may be included.

The diode model can be used for either junction diodes or Schottky barrier diodes. The JFET model is based on the FET model of Shichman and Hodges.

Six MOSFET models are implemented: MOS1 is described by a square-law I-V characteristic, MOS2 [1] is an analytical model, while MOS3 [1] is a semi-empirical model; MOS6 [2] is a simple analytic model accurate in the short-channel region; MOS4 [3, 4] and MOS5 [5] are the BSIM (Berkeley Short-channel IGFET Model) and BSIM2. MOS2, MOS3, and MOS4 include second-order effects such as channel-length modulation, sub threshold conduction, scattering-limited velocity saturation, small-size effects, and charge-controlled capacitances.

The area factor used on the diode, BJT, JFET, and MESFET devices determines the number of equivalent parallel devices of a specified model. The affected parameters are marked with an asterisk under the heading 'area' in the model descriptions below. Several geometric factors associated with the channel and the drain and source diffusions can be specified on the MOSFET device line.

Two different forms of initial conditions may be specified for some devices. The first form is included to improve the DC convergence for circuits that contain more than one stable state. If a device is specified OFF, the DC operating point is determined with the terminal voltages for that device set to zero. After convergence is obtained, the program continues to iterate to obtain the exact value for the terminal voltages. If a circuit has more than one DC stable state, the OFF option can be used to force the solution to correspond to a desired state. If a device is specified OFF when in reality the device is conducting, the program still obtains the correct solution (assuming the solutions converge) but more iterations are required since the program must independently converge to two separate solutions. The .NODESET control line serves a similar purpose as the OFF option. The .NODESET option is easier to apply and is the preferred means to aid convergence.

The second form of initial conditions is specified for use with the transient analysis. These are true 'initial conditions' as opposed to the convergence aids above. See the description of the .IC control line and the .TRAN control line for a detailed explanation of initial conditions.

4.4.1 Dxxxx: Junction Diodes

General form:

```
DXXXXXXX N+ N- MNAME <AREA> <OFF> <IC=VD> <TEMP=T>
```

Examples:

```
DBRIDGE 2 10 DIODE1
DCLMP 3 7 DMOD 3.0 IC=0.2
```

N+ and **N-** are the positive and negative nodes, respectively.

MNAME is the model name, **AREA** is the area factor, and **OFF** indicates an (optional) starting condition on the device for DC analysis. If the area factor is omitted, a value of 1.0 is assumed.

The (optional) initial condition specification using **IC=VD** is intended for use with the UIC option on the .TRAN control line, when a transient analysis is desired starting from other than the quiescent operating point.

The (optional) **TEMP** value is the temperature at which this device is to operate, and overrides the temperature specification on the .OPTION control line.

4.4.1.1 Diode Model (D)

The DC characteristics of the diode are determined by the parameters **IS** and **N**. An ohmic resistance, **RS**, is included.

Charge storage effects are modelled by a transit time, **TT**, and a non-linear depletion layer capacitance which is determined by the parameters **CJO**, **VJ**, and **M**.

The temperature dependence of the saturation current is defined by the parameters **EG**, the energy and **XTI**, the saturation current temperature exponent. The nominal temperature at which these parameters were measured is TNOM, which defaults to the circuit-wide value specified on the .OPTIONS control line.

Reverse breakdown is modelled by an exponential increase in the reverse diode current and is determined by the parameters **BV** and **IBV** (both of which are positive numbers).

name	parameter	units	default	example	area
IS	saturation current	A	1.0e-14	1.0e-14	*
RS	ohmic resistance	Ω	0	10	*
N	emission coefficient	-	1	1.0	
TT	transit-time	sec	0	0.1ns	
CJO	zero-bias junction capacitance	F	0	2pF	*
VJ	junction potential	V	1	0.6	
M	grading coefficient	-	0.5	0.5	
EG	activation energy	eV	1.11	1.11 Si 0.69 Sbd 0.67 Ge	
XTI	saturation-current temp. exp	-	3.0	3.0 jn 2.0 Sbd	
KF	flicker noise coefficient	-	0		
AF	flicker noise exponent	-	1		
FC	coefficient for forward-bias depletion capacitance formula	-	0.5		
BV	reverse breakdown voltage	V	infinite	40.0	
IBV	current at breakdown voltage	A	1.0e-3		
TNOM	parameter measurement temperature	°C	27	50	

4.4.2 Qxxxx: Bipolar Junction Transistors (BJTs)

General form:

```
QXXXXXXX NC NB NE <NS> MNAME <AREA> <OFF> <IC=VBE, VCE> <TEMP=T>
```

Examples:

```
Q23 10 24 13 QMOD IC=0.6, 5.0
Q50A 11 26 4 20 MOD1
```

NC, NB, and NE are the collector, base, and emitter nodes, respectively. NS is the (optional) substrate node. If unspecified, ground is used.

MNAME is the model name, AREA is the area factor, and OFF indicates an (optional) initial condition on the device for the DC analysis. If the area factor is omitted, a value of 1.0 is assumed. The (optional) initial condition specification using IC=VBE, VCE is intended for use with the UIC option on the .TRAN control line, when a transient analysis is desired starting from other than the quiescent operating point. See the .IC control line description for a better way to set transient initial conditions.

The (optional) TEMP value is the temperature at which this device is to operate, and overrides the temperature specification on the .OPTION control line.

4.4.2.1 BJT Models (NPN/PNP)

The bipolar junction transistor model in **WinSpice3** is an adaptation of the integral charge control model of Gummel and Poon. This modified Gummel-Poon model extends the original model to include several effects at high bias levels. The model automatically simplifies to the simpler Ebers-Moll model when certain parameters are not specified. The parameter names used in the modified Gummel-Poon model have been chosen to be more easily understood by the program user, and to reflect better both physical and circuit design thinking.

The DC model is defined by the parameters **IS**, **BF**, **NF**, **ISE**, **IKF**, and **NE** which determine the forward current gain characteristics, **IS**, **BR**, **NR**, **ISC**, **IKR**, and **NC** which determine the reverse current gain characteristics, and **VAF** and **VAR** which determine the output conductance for forward and reverse regions.

Three ohmic resistances **RB**, **RC**, and **RE** are included, where **RB** can be highly current dependent. Base charge storage is modelled by forward and reverse transit times, **TF** and **TR**, the forward transit time **TF** being bias dependent if desired.

CJE, **VJE**, and **MJE** determine non-linear depletion layer capacitances for the B-E junction, **CJC**, **VJC**, and **MJC** for the B-C junction and **CJS**, **VJS**, and **MJS** for the C-S (Collector-Substrate) junction.

The temperature dependence of the saturation current, **IS**, is determined by the energy-gap, **EG**, and the saturation current temperature exponent, **XTI**. Additionally, the beta temperature exponent **XTB** in the new model models base current temperature dependence. It is assumed that the values specified were measured at the temperature TNOM, which can be specified on the .OPTIONS control line or overridden by a specification on the .MODEL line.

The BJT parameters used in the modified Gummel-Poon model are listed below. The parameter names used in earlier versions of **SPICE2** are still accepted.

Modified Gummel-Poon BJT Parameters.

name	parameter	units	default	example	area
IS	transport saturation current	A	1.0e-16	1.0e-15	*
BF	ideal maximum forward beta	-	100	100	
NF	forward current emission coefficient	-	1.0	1	
VAF	forward Early voltage	V	infinite	200	
IKF	corner for forward beta high current roll-off	A	infinite	0.01	*
ISE	B-E leakage saturation current	A	0	1.0e-13	*
NE	B-E leakage emission coefficient	-	1.5	2	
BR	ideal maximum reverse beta	-	1	0.1	
NR	reverse current emission coefficient	-	1	1	
VAR	reverse Early voltage	V	infinite	200	
IKR	corner for reverse beta high current roll-off	A	infinite	0.01	*
ISC	B-C leakage saturation current	A	0	1.0e-13	*

name	parameter	units	default	example	area
NC	B-C leakage emission coefficient	-	2	1.5	
RB	zero bias base resistance	Ω	0	100	*
IRB	current where base resistance falls halfway to its min value	A	infinite	0.1	*
RBM	minimum base resistance at high currents	Ω	RB	10	*
RE	emitter resistance	Ω	0	1	*
RC	collector resistance	Ω	0	10	*
CJE	B-E zero-bias depletion capacitance	F	0	2pF	*
VJE	B-E built-in potential	V	0.75	0.6	
MJE	B-E junction exponential factor	-	0.33	0.33	
TF	ideal forward transit time	sec	0	0.1ns	
XTF	coefficient for bias dependence of TF	-	0		
VTF	voltage describing VBC dependence of TF	V	infinite		
ITF	high-current parameter for effect on TF	A	0		*
PTF	excess phase at freq=1.0/(TF*2PI) Hz	deg	0		
CJC	B-C zero-bias depletion capacitance	F	0	2pF	*
VJC	B-C built-in potential	V	0.75	0.5	
MJC	B-C junction exponential factor	-	0.33	0.5	
XCJC	fraction of B-C depletion capacitance connected to internal base node	-	1		
TR	ideal reverse transit time	sec	0	10ns	
CJS	zero-bias collector-substrate capacitance	F	0	2pF	*
VJS	substrate junction built-in potential	V	0.75		
MJS	substrate junction exponential factor	-	0	0.5	
XTB	forward and reverse beta temperature exponent	-	0		
EG	energy gap for temperature effect on IS	eV	1.11		
XTI	temperature exponent for effect on IS	-	3		
KF	flicker-noise coefficient	-	0		

name	parameter	units	default	example	area
AF	flicker-noise exponent	-	1		
FC	coefficient for forward-bias depletion capacitance formula	-	0.5		
TNOM	Parameter measurement temperature	°C	27	50	

4.4.3 Jxxxx: Junction Field-Effect Transistors (JFETs)

General form:

```
JXXXXXXX ND NG NS MNAME <AREA> <OFF> <IC=VDS, VGS> <TEMP=T>
```

Examples:

```
J1 7 2 3 JM1 OFF
```

ND, **NG**, and **NS** are the drain, gate, and source nodes, respectively.

MNAME is the model name, **AREA** is the area factor, and **OFF** indicates an (optional) initial condition on the device for DC analysis. If the area factor is omitted, a value of 1.0 is assumed.

The (optional) initial condition specification, using **IC=VDS, VGS** is intended for use with the UIC option on the .TRAN control line, when a transient analysis is desired starting from other than the quiescent operating point. See the .IC control line for a better way to set initial conditions.

The (optional) **TEMP** value is the temperature at which this device is to operate, and overrides the temperature specification on the .OPTION control line.

4.4.3.1 JFET Models (NJF/PJF)

WinSpice provides two JFET models:-

LEVEL=1 -> Shichman-Hodges

LEVEL=2 -> Parker-Skellern FET model (see [9])

The Level 1 JFET model is derived from the FET model of Shichman and Hodges.

The Level 2 model is an alternative model by Anthony Parker at Macquarie University.

In both models, the DC characteristics are defined by the parameters **VTO** and **BETA**, which determine the variation of drain current with gate voltage, **LAMBDA**, which determines the output conductance, and **IS**, the saturation current of the two gate junctions. Two ohmic resistances, **RD** and **RS**, are included. Charge storage is modelled by non-linear depletion layer capacitances for both gate junctions which vary as the -1/2 power of junction voltage and are defined by the parameters **CGS**, **CGD**, and **PB**.

name	parameter	units	default	example	area
VTO	threshold voltage (V_{TO})	V	-2.0	-2.0	
BETA	transconductance parameter (B)	A/V^2	1.0e-4	1.0e-3	*
LAMBDA	channel-length modulation parameter (λ)	1/V	0	1.0e-4	
RD	drain ohmic resistance	Ω	0	100	*
RS	source ohmic resistance	Ω	0	100	*
CGS	zero-bias G-S junction capacitance (C_{gs})	F	0	5pF	*
CGD	zero-bias G-D junction capacitance (C_{gd})	F	0	1pF	*
PB	gate junction potential	V	1	0.6	
IS	gate junction saturation current (I_S)	A	1.0e-14	1.0e-14	*
B	doping tail parameter	-	1	1.1	
KF	flicker noise coefficient	-	0		
AF	flicker noise exponent	-	1		
FC	coefficient for forward-bias depletion capacitance formula	-	0.5		
TNOM	parameter measurement temperature	°C	27	50	

4.4.4 Mxxxx: MOSFETs

General form:

```
MXXXXXXX ND NG NS NB MNAME <L=VAL> <W=VAL> <AD=VAL> <AS=VAL>
+ <PD=VAL> <PS=VAL> <NRD=VAL> <NRS=VAL> <OFF>
+ <IC=VDS, VGS, VBS> <TEMP=T>
```

Examples:

```
M1 24 2 0 20 TYPE1
M31 2 17 6 10 MODM L=5U W=2U
M1 2 9 3 0 MOD1 L=10U W=5U AD=100P AS=100P PD=40U PS=40U
```

ND, **NG**, **NS**, and **NB** are the drain, gate, source, and bulk (substrate) nodes, respectively.

MNAME is the model name.

L and **W** are the channel length and width, in meters. **AD** and **AS** are the areas of the drain and source diffusions, in $meters^2$. Note that the suffix U specifies microns (1e-6 m) and P sq.-microns (1e-12 m^2).

If any of **L**, **W**, **AD**, or **AS** are not specified, the default values defined by the .OPTION control line variables **DEFL**, **DEFW**, **DEFAD** and **DEFAS** are used (see section 5.1). The use of defaults simplifies input file preparation, as well as the editing required if device geometry's are to be changed.

PD and **PS** are the perimeters of the drain and source junctions, in meters and default to 0.0

NRD and **NRS** designate the equivalent number of squares of the drain and source diffusions; these values multiply the sheet resistance **RSH** specified on the .MODEL control line for an accurate representation of the parasitic series drain and source resistance of each transistor. **NRD** and **NRS** default to 1.0.

OFF indicates an (optional) initial condition on the device for DC analysis. The (optional) initial condition specification using IC=VDS, VGS, VBS is intended for use with the UIC option on the .TRAN control line, when a transient analysis is desired starting from other than the quiescent operating point. See the .IC control line for a better and more convenient way to specify transient initial conditions.

The (optional) **TEMP** value is the temperature at which this device is to operate, and overrides the temperature specification on the .OPTION control line. The temperature specification is ONLY valid for level 1, 2, 3, and 6 MOSFETs, not for level 4 or 5 (BSIM) devices.

4.4.4.1 MOSFET Models (NMOS/PMOS)

SPICE provides four MOSFET device models, which differ in the formulation of the I-V characteristic. The variable LEVEL specifies the model to be used:-

LEVEL=1	Shichman-Hodges
LEVEL=2	MOS2 (as described in [1])
LEVEL=3	MOS3, a semi-empirical model(see [1])
LEVEL=4	BSIM1 (as described in [3])
LEVEL=5	BSIM2 (as described in [5])
LEVEL=6	MOS6 (as described in [6])
LEVEL=8	BSIM3
LEVEL=9	B3SOI
LEVEL=14	BSIM4
LEVEL=44	EKV from Ecole Polytechnique Federale de Lausanne (see `http://legwww.epfl.ch/ekv`)
LEVEL=49	BSIM3 (same as LEVEL=8 for HSPICE compatibility)

Note that three versions of the BSIM3 model are supported by **WinSpice**. This is needed because different versions of BSIM3 are not compatible with each other in term of the model parameters. The version is selected by placing a 'VERSION=x.x.x' option in the .MODEL line as follows:-

VERSION=3.1	BSIM3 v3.1
VERSION=3.2	BSIM3 v3.2
Omitted	BSIM3 v3.2.2

The DC characteristics of the level 1 through level 3 MOSFETs are defined by the device parameters **VTO**, **KP**, **LAMBDA**, **PHI** and **GAMMA**. These parameters are computed by **WinSpice3** if process parameters (**NSUB**, **TOX**, .) are given, but user-specified values always override. **VTO** is positive (negative) for enhancement mode and negative (positive) for depletion mode N-channel (P-channel) devices.

Charge storage is modelled by three constant capacitors, **CGSO**, **CGDO**, and **CGBO** which represent overlap capacitances, by the non-linear thin-oxide capacitance which is distributed among the gate, source, drain, and bulk regions, and by the non-linear depletion-layer capacitances for both substrate junctions divided into bottom and periphery. These vary as the **MJ** and **MJSW** power of junction voltage respectively, and are determined by the parameters **CBD**, **CBS**, **CJ**, **CJSW**, **MJ**, **MJSW** and **PB**. Charge the piecewise linear voltages-dependent capacitance model proposed by Meyer models storage effects. The thin-oxide charge-storage effects are treated slightly different for the LEVEL=1 model. These voltage-dependent capacitances are included only if **TOX** is specified in the input description and they are represented using Meyer's formulation.

There is some overlap among the parameters describing the junctions, e.g. the reverse current can be input either as **IS** (in A) or as **JS** (in A/m^2). Whereas the first is an absolute value the second is multiplied by **AD** and **AS** to give the reverse current of the drain and source junctions respectively. This methodology has been chosen since there is no sense in relating always junction characteristics with **AD** and **AS** entered on the device line; the areas can be defaulted. The same idea applies also to the zero-bias junction capacitances **CBD** and **CBS** (in F) on one hand, and **CJ** (in F/m^2) on the other.

The parasitic drain and source series resistance can be expressed as either **RD** and **RS** (in ohms) or **RSH** (in ohms/sq.), the latter being multiplied by the number of squares **NRD** and **NRS** input on the device line.

A discontinuity in the MOS level 3 model with respect to the **KAPPA** parameter has been detected (see [10]). The supplied fix has been implemented in **WinSpice3**. Since this fix may affect parameter fitting, the option "BADMOS3" may be set to use the old implementation (see the section on simulation variables and the ".OPTIONS" line).

SPICE level 1, 2, 3 and 6 parameters:

Name	parameter	units	default	example
LEVEL	model index	-	1	
VTO	zero-bias threshold voltage (V_{TO})	V	0.0	1.0
KP	transconductance parameter	A/V^2	2.0e-5	3.1e-5
GAMMA	bulk threshold parameter (γ)	$V^{1/2}$	0.0	0.37
PHI	surface potential (ϕ)	V	0.6	0.65
LAMBDA	channel-length modulation (MOS1 and MOS2 only) (λ)	1/V	0.0	0.02
RD	drain ohmic resistance	Ω	0.0	1.0
RS	source ohmic resistance	Ω	0.0	1.0
CBD	zero-bias B-D junction capacitance	F	0.0	20fF
CBS	zero-bias B-S junction capacitance	F	0.0	20fF
IS	bulk junction saturation current (I_S)	A	1.0e-14	1.0e-15
PB	bulk junction potential	V	0.8	0.87
CGSO	gate-source overlap capacitance per meter channel width	F/m	0.0	4.0e-11

Name	parameter	units	default	example
CGDO	gate-drain overlap capacitance per meter channel width	F/m	0.0	4.0e-11
CGBO	gate-bulk overlap capacitance per meter channel length	F/m	0.0	2.0e-10
RSH	drain and source diffusion sheet resistance	Ω/[]	0.0	10.0
CJ	zero-bias bulk junction bottom cap per sq.-meter of junction area	F/m^2	0.0	2.0e-4
MJ	bulk junction bottom grading coefficient.	-	0.5	0.5
CJSW	zero-bias bulk junction sidewall cap. per meter of junction perimeter	F/m	0.0	1.0e-9
MJSW	bulk junction sidewall grading coefficient.	-	0.50 (level1) 0.33 (level2, 3)	
JS	bulk junction saturation current per sq.-meter of junction area	A/m^2		1.0e-8
TOX	Oxide thickness	meter	1.0e-7	1.0e-7
NSUB	Substrate doping	$1/cm^3$	0.0	4.0e15
NSS	Surface state density	$1/cm^2$	0.0	1.0e10
NFS	fast surface state density	$1/cm^2$	0.0	1.0e10
TPG	type of gate material: +1 opp. to substrate -1 same as substrate 0 Al gate	-	1.0	
XJ	Metallurgical junction depth	meter	0.0	1µ
LD	lateral diffusion	meter	0.0	0.8µ
UO	surface mobility	cm^2/Vs	600	700
UCRIT	critical field for mobility degradation (MOS2 only)	V/cm	1.0e4	1.0e4
UEXP	critical field exponent in mobility degradation (MOS2 only)	-	0.0	0.1
UTRA	Transverse field coefficient (mobility) (deleted for MOS2)	-	0.0	0.3
VMAX	Maximum drift velocity of carriers	m/s	0.0	5.0e4

Name	parameter	units	default	example
NEFF	total channel-charge (fixed and mobile) coefficient (MOS2 only)	-	1.0	5.0
KF	flicker noise coefficient	-	0.0	1.0e-26
AF	flicker noise exponent	-	1.0	1.2
FC	Coefficient for forward-bias depletion capacitance formula	-	0.5	
DELTA	width effect on threshold voltage (MOS2 and MOS3)	-	0.0	1.0
THETA	mobility modulation (MOS3 only)	1/V	0.0	0.1
ETA	static feedback (MOS3 only)	-	0.0	1.0
KAPPA	Saturation field factor (MOS3 only)	-	0.2	0.5
TNOM	Parameter measurement temperature	°C	27	50

The level 4 and level 5 (BSIM1 and BSIM2) parameters are all values obtained from process characterisation, and can be generated automatically. J. Pierret [4] describes a means of generating a 'process' file, and the program Proc2Mod provided with **WinSpice3** converts this file into a sequence of BSIM1 ".MODEL" lines suitable for inclusion in a **WinSpice3** input file. Parameters marked below with an * in the l/w column also have corresponding parameters with a length and width dependency. For example, VFB is the basic parameter with units of Volts, and LVFB and WVFB also exist and have units of Volt-micrometer. The formula

$$P = P_0 + \frac{P_L}{L_{effective}} + \frac{P_W}{W_{effective}}$$

is used to evaluate the parameter for the actual device specified with

$$L_{effective} = L_{input} - DL$$

and

$$W_{effective} = W_{input} - DW$$

Note that unlike the other models in **WinSpice3**, the BSIM model is designed for use with a process characterisation system that provides all the parameters, thus there are no defaults for the parameters, and leaving one out is considered an error. For an example set of parameters and the format of a process file, see the SPICE2 implementation notes [3].

For more information on BSIM2, see reference [5].

SPICE BSIM (level 4) parameters.

name	parameter	units	l/w
VFB	flat-band voltage	V	*
PHI	surface inversion potential	V	*
K1	body effect coefficient	$V^{1/2}$	*
K2	drain/source depletion charge-sharing coefficient	-	*
ETA	zero-bias drain-induced barrier-lowering coefficient	-	*
MUZ	zero-bias mobility	cm^2/V-s	
DL	shortening of channel	µm	
DW	narrowing of channel	µm	
U0	zero-bias transverse-field mobility degradation coefficient	V^{-1}	*
U1	zero-bias velocity saturation coefficient	µm/V	*
X2MZ	sens. of mobility to substrate bias at V_{ds}=0	cm^2/V^2-s	*
X2E	sens. of drain-induced barrier lowering effect to substrate bias	V^{-1}	*
X3E	sens. of drain-induced barrier lowering effect to drain bias at $V_{ds}=V_{dd}$	V^{-1}	*
X2U0	sens. of transverse field mobility degradation effect to substrate bias	V^{-2}	*
X2U1	sens. of velocity saturation effect to substrate bias	μmV^{-2}	*
MUS	mobility at zero substrate bias and at $V_{ds}=V_{dd}$	cm^2/V^2-s	
X2MS	sens. of mobility to substrate bias at $V_{ds}=V_{dd}$	cm^2/V^2-s	*
X3MS	sens. of mobility to drain bias at $V_{ds}=V_{dd}$	cm^2/V^2-s	*
X3U1	sens. of velocity saturation effect on drain bias at $V_{ds}=V_{dd}$	µmV	*
TOX	gate oxide thickness	µm	
TEMP	temperature at which parameters were measured	°C	
VDD	measurement bias range	V	
CGDO	gate-drain overlap capacitance per meter channel width	F/m	
CGSO	gate-source overlap capacitance per meter channel width	F/m	
CGBO	gate-bulk overlap capacitance per meter channel length	F/m	

name	parameter	units	l/w
XPART	gate-oxide capacitance-charge model flag	-	
N0	zero-bias sub threshold slope coefficient	-	*
NB	sens. of sub threshold slope to substrate bias	-	*
ND	sens. of sub threshold slope to drain bias	-	*
RSH	drain and source diffusion sheet resistance	Ω/[]	
JS	source drain junction current density	A/m^2	
PB	built in potential of source drain junction	V	
MJ	Grading coefficient of source drain junction	-	
PBSW	built in potential of source, drain junction sidewall	V	
MJSW	grading coefficient of source drain junction sidewall	-	
CJ	Source drain junction capacitance per unit area	F/m^2	
CJSW	source drain junction sidewall capacitance per unit length	F/m	
WDF	source drain junction default width	m	
DELL	Source drain junction length reduction	m	

XPART = 0 selects a 40/60 drain/source charge partition in saturation, while XPART=1 selects a 0/100 drain/source charge partition.

ND, **NG**, and **NS** are the drain, gate, and source nodes, respectively.

MNAME is the model name, **AREA** is the area factor, and **OFF** indicates an (optional) initial condition on the device for DC analysis. If the area factor is omitted, a value of 1.0 is assumed.

The (optional) initial condition specification, using **IC=VDS, VGS** is intended for use with the UIC option on the .TRAN control line, when a transient analysis is desired starting from other than the quiescent operating point. See the .IC control line for a better way to set initial conditions.

4.4.5 Zxxxx: MESFETs

General form:

```
ZXXXXXXX ND NG NS MNAME <AREA> <OFF> <IC=VDS, VGS>
```

Examples:

```
Z1 7 2 3 ZM1 OFF
```

4.4.5.1 MESFET Models (NMF/PMF)

The MESFET model is derived from the GaAs FET model of Statz et al. as described in [11]. The DC characteristics are defined by the parameters VTO, B, and BETA, which determine the variation of drain current with gate voltage, ALPHA, which determines saturation voltage, and LAMBDA, which determines the output conductance. The formula are given by:

$$I_d = \frac{\beta(V_{gs} - V_T)^2}{1 + b(V_{gs} - V_T)}\left[1 - \left[1 - \alpha\frac{V_{ds}}{3}\right]^3\right](1 + \lambda V_{ds}) \qquad \text{for } 0 < V_{ds} < \frac{3}{\alpha}$$

$$I_d = \frac{\beta(V_{gs} - V_T)^2}{1 + b(V_{gs} - V_T)}(1 + \lambda V_{ds}) \qquad \text{for } V_{ds} > \frac{3}{\alpha}$$

Two ohmic resistances, RD and RS, are included. Charge storage is modelled by total gate charge as a function of gate-drain and gate-source voltages and is defined by the parameters CGS, CGD, and PB.

name	Parameter	units	default	example	area
VTO	pinch-off voltage	V	-2.0	-2.0	
BETA	transconductance parameter	A/V^2	1.0e-4	1.0e-3	*
B	doping tail extending parameter	1/V	0.3	0.3	*
ALPHA	saturation voltage parameter	1/V	2	2	*
LAMBDA	channel-length modulation parameter	1/V	0	1.0e-4	
RD	drain ohmic resistance	Ω	0	100	*
RS	source ohmic resistance	Ω	0	100	*
CGS	zero-bias G-S junction capacitance	F	0	5pF	*
CGD	zero-bias G-D junction capacitance	F	0	1pF	*
PB	gate junction potential	V	1	0.6	
KF	flicker noise coefficient	-	0		
AF	flicker noise exponent	-	1		
FC	coefficient for forward-bias depletion capacitance formula	-	0.5		

5 ANALYSES AND OUTPUT CONTROL

The following command lines are for specifying analyses or plots within the circuit description file. Parallel commands exist in the interactive command interpreter (detailed in the following section). Specifying analyses and plots (or tables) in the input file is useful for batch runs. Batch mode is entered when either the -b option is given or when the default input source is redirected from a file. In batch mode, the analyses specified by the control lines in the input file (e.g. ".ac", ".tran", etc.) are immediately executed (unless ".control" lines exist; see the section on the interactive command interpreter).

Output plots (in "line-printer" form) and tables can be printed according to the .PRINT, .PLOT, and .FOUR control lines, described next. .PLOT, .PRINT, and .FOUR lines are meant for compatibility with SPICE2.

5.1 .OPTIONS: Simulator Variables

Various parameters of the simulations available in **WinSpice3** can be altered to control the accuracy, speed, or default values for some devices. These parameters may be changed via the "set" command (described later in the section on the interactive front-end) or via the ".OPTIONS" line:

General form:

```
.OPTIONS OPT1 OPT2 ... (or OPT=OPTVAL ...)
```

Examples:

```
.OPTIONS RELTOL=.005 TRTOL=8
```

The options line allows the user to reset program control and user options for specific simulation purposes. See the following section on the interactive command interpreter for the parameters that may be set with a .OPTIONS line and the format of the 'set' command. Any combination of the following options may be included, in any order. 'x' (below) represents some positive number.

option	effect
ABSTOL=x	Sets the absolute current error tolerance of the program. The default value is 1 picoamp.
BADMOS3	Use the older version of the MOS3 model with the "kappa" discontinuity.
BYPASS=x	This option, when set to a non-zero value, avoids recomputation of nonlinear functions that do not change with iterations. The default value is 0.
CAPBRANCH	Calculate capacitor branch currents during analyses. This is an experimental feature which can cause convergence problems but which may be useful in some cases.
CHGTOL=x	Sets the charge tolerance of the program. The default value is 1.0e-14.
DEFAD=x	Sets the value for MOS drain diffusion area; the default is 0.0.
DEFAS=x	Sets the value for MOS source diffusion area; the default is 0.0.
DEFL=x	Sets the value for MOS channel length; the default is 100.0 micrometer.
DEFW=x	Sets the value for MOS channel width; the default is 100.0 micrometer.
DELMIN=x	Sets the minimum timestep value used in transient analyses. If WinSpice3 tries to go below this value in attempting to achieve convergence, the analysis will be aborted. A value of 0 disables the minimum limit. Any negative value sets the minimum timestep to (1E-9*TMAX) as in the original Spice3. The default value is –1.
GMIN=x	Sets the value of GMIN, the minimum conductance allowed by the program. The default value is 1.0e-12.
ITL1=x	Sets the DC iteration limit. The default is 100.
ITL2=x	Sets the DC transfer curve iteration limit. The default is 50.
ITL3=x	Sets the lower transient analysis iteration limit. The default value is 4. (Note: not implemented in WinSpice3).
ITL4=x	Sets the transient analysis timepoint iteration limit. The default is 10.
ITL5=x	Sets the transient analysis total iteration limit. A value of 0 (the default) disables this limit.
KEEPOPINFO	Retain the operating point information when an AC, Distortion, or Pole-Zero analysis is run. This is particularly useful if the circuit is large and you do not want to run a (redundant) ".OP" analysis.

option	effect
METHOD=name	Sets the numerical integration method used by SPICE. Possible names are "Gear" or "trapezoidal" (or just "trap"). The default is trapezoidal.
MINTIMESTEP=x	The same as DELMIN.
PIVREL=x	Sets the relative ratio between the largest column entry and an acceptable pivot value. The default value is 1.0e-3. In the numerical pivoting algorithm the allowed minimum pivot value is determined by EPSREL=AMAX1(PIVREL*MAXVAL, PIVTOL) where MAXVAL is the maximum element in the column where a pivot is sought (partial pivoting).
PIVTOL=x	Sets the absolute minimum value for a matrix entry to be accepted as a pivot. The default value is 1.0e-13.
RELTOL=x	Resets the relative error tolerance of the program. The default value is 0.001 (0.1%).
RESBRANCH	Calculate resistor branch currents during analyses. This is an experimental feature which can cause convergence problems but which may be useful in some cases.
RSHUNT=x	Shunt resistors of value x are placed between all voltage nodes and node 0 (the ground node). This helps avoid nodes having no DC paths to ground and hence not converging. It also allow for more realistic circuits to be simulated. If x is zero, shunt resistors are not placed in the circuit. By default, x = 0.
SCALE=x	Element scaling factor used as a multiplier for device dimension parameters L, W, AD, AS, PD and PS. Currently used only used by the following device models:- MOS1 (MOSFET level 1) MOS2 (MOSFET level 2) MOS3 (MOSFET level 3) BSIM1 (MOSFET level 4) BSIM2 (MOSFET level 5) MOS6 (MOSFET level 6) BSIM3 (MOSFET level 8) EKV (MOSFET level 44) The default value is 1.0.
TEMP=x	Sets the operating temperature of the circuit. The default value is 27 deg C (300 deg K). TEMP can be overridden by a temperature specification on any temperature dependent instance.

option	effect
TNOM=x	Sets the nominal temperature at which device parameters are measured. The default value is 27 deg C (300 deg K). TNOM can be overridden by a specification on any temperature dependent device model.
TRTOL=x	Sets the transient error tolerance. The default value is 7.0. This parameter is an estimate of the factor by which SPICE overestimates the actual truncation error.
TRYTOCOMPACT	Applicable only to the LTRA model. When specified, the simulator tries to condense LTRA transmission lines' past history of input voltages and currents.
VNTOL=x	Sets the absolute voltage error tolerance of the program. The default value is 1 microvolt.

In addition, the following options have the listed effect when operating in **SPICE2** emulation mode:

option	effect
ACCT	causes accounting and run time statistics to be printed
LIST	causes the summary listing of the input data to be printed
NOMOD	suppresses the printout of the model parameters
NOPAGE	suppresses page ejects
NODE	causes the printing of the node table.
OPTS	causes the option values to be printed.

5.2 Initial Conditions

5.2.1 .NODESET: Specify Initial Node Voltage Guesses

General form:

```
.NODESET V(NODNUM)=VAL V(NODNUM)=VAL ...
```

Examples:

```
.NODESET V(12)=4.5 V(4)=2.23
```

The Nodeset line helps the program find the DC or initial transient solution by making a preliminary pass with the specified nodes held to the given voltages. The restriction is then released and the iteration continues to the true solution. The .NODESET line may be necessary for convergence on bistable or a-stable circuits. In general, this line should not be necessary.

5.2.2 .IC: Set Initial Conditions

General form:

```
.IC V(NODNUM)=VAL V(NODNUM)=VAL ...
```

Examples:

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

The IC line is for setting transient initial conditions. It has two different interpretations, depending on whether the UIC parameter is specified on the .TRAN control line. Also, one should not confuse this line with the .NODESET line. The .NODESET line is only to help DC convergence, and does not affect final bias solution (except for multi-stable circuits). The two interpretations of this line are as follows:

1. When the UIC parameter is specified on the .TRAN line, then the node voltages specified on the .IC control line are used to compute the capacitor, diode, BJT, JFET, and MOSFET initial conditions. This is equivalent to specifying the IC=... parameter on each device line, but is much more convenient. The IC=... parameter can still be specified and takes precedence over the .IC values. Since no DC bias (initial transient) solution is computed before the transient analysis, one should take care to specify all DC source voltages on the .IC control line if they are to be used to compute device initial conditions.

2. When the UIC parameter is not specified on the .TRAN control line, the DC bias (initial transient) solution is computed before the transient analysis. In this case, the node voltages specified on the .IC control line are forced to the desired initial values during the bias solution. During transient analysis, the constraint on these node voltages is removed. This is the preferred method since it allows SPICE to compute a consistent DC solution.

5.3 Analyses

5.3.1 .AC: Small-Signal AC Analysis

General form:

```
.AC DEC ND FSTART FSTOP
.AC OCT NO FSTART FSTOP
.AC LIN NP FSTART FSTOP
```

Examples:

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

DEC stands for decade variation, and **ND** is the number of points per decade. **OCT** stands for octave variation, and **NO** is the number of points per octave. **LIN** stands for linear variation, and **NP** is the number of points. **FSTART** is the starting frequency, and **FSTOP** is the final frequency. If this line is included in the input file, **WinSpice3** performs an AC analysis of the circuit over the specified frequency range. Note that in order for this analysis to be meaningful, at least one independent source must have been specified with an AC value.

5.3.2 .DC: DC Transfer Function

General form:

```
.DC SRCNAM VSTART VSTOP VINCR [SRC2 START2 STOP2 INCR2]
```

Examples:

```
.DC VIN 0.25 5.0 0.25
.DC VDS 0 10 .5 VGS 0 5 1
.DC VCE 0 10 .25 IB 0 10U 1U
```

The DC line defines the DC transfer curve source and sweep limits (again with capacitors open and inductors shorted). **SRCNAM** is the name of an independent voltage or current source. **VSTART**, **VSTOP**, and **VINCR** are the starting, final, and incrementing values respectively.

The first example causes the value of the voltage source VIN to be swept from 0.25 Volts to 5.0 Volts in increments of 0.25 Volts. A second source (SRC2) may optionally be specified with associated sweep parameters. In this case, the first source is swept over its range for each value of the second source. This option can be useful for obtaining semiconductor device output characteristics. See the second example circuit description in Appendix A.

5.3.3 .DISTO: Distortion Analysis

General form:

```
.DISTO DEC ND FSTART FSTOP <F2OVERF1>
.DISTO OCT NO FSTART FSTOP <F2OVERF1>
.DISTO LIN NP FSTART FSTOP <F2OVERF1>
```

Examples:

```
.DISTO DEC 10 1kHz 100Mhz
.DISTO DEC 10 1kHz 100Mhz 0.9
```

The DISTO line does a small-signal distortion analysis of the circuit. A multi-dimensional Volterra series analysis is done using multi-dimensional Taylor series to represent the nonlinearities at the operating point. Terms of up to third order are used in the series expansions.

If the optional parameter F2OVERF1 is not specified, .DISTO does a harmonic analysis - i.e., it analyses distortion in the circuit using only a single input frequency F1, which is swept as specified by arguments of the .DISTO command exactly as in the .AC command. Inputs at this frequency may be present at more than one input source, and their magnitudes and phases are specified by the arguments of the DISTOF1 keyword in the input file lines for the input sources (see the description for independent sources - the arguments of the DISTOF2 keyword are not relevant in this case). The analysis produces information about the AC values of all node voltages and branch currents at the harmonic frequencies 2F1 and 3F1, vs. the input frequency F1 as it is swept. A value of 1 (as a complex distortion output) signifies cos(2J(2F1)t) at 2F1 and cos(2J(3F1)t) at 3F1, using the convention that 1 at the input fundamental frequency is equivalent to cos(2JF1t). The distortion component desired (2F1 or 3F1) can be selected using commands in WinSpice3, and then printed or plotted (normally, one is interested primarily in the magnitude of the harmonic components, so the magnitude of the AC distortion value is looked at). It should be noted that these are the AC values of the actual harmonic components, and are not equal to HD2 and HD3. To obtain HD2 and HD3, one must divide by the corresponding AC values at F1, obtained from a .AC line. This division can be done using WinSpice3 commands.

If the optional F2OVERF1 parameter is specified, it should be a real number between (and not equal to) 0.0 and 1.0; in this case, .DISTO does a spectral analysis. It considers the circuit with sinusoidal inputs at two different frequencies F1 and F2. F1 is swept according to the .DISTO control line options exactly as in the .AC control line. F2 is kept fixed at a single frequency as F1 sweeps - the value at which it is kept fixed is equal to F2OVERF1 times FSTART. Each independent source in the circuit may potentially have two (superimposed) sinusoidal inputs for distortion, at the frequencies F1 and F2. The magnitude and phase of the F1 component are specified by the arguments of the DISTOF1 keyword in the source's input line (see the description of independent sources); the magnitude and phase of the F2 component are specified by the

arguments of the DISTOF2 keyword. The analysis produces plots of all node voltages/branch currents at the intermodulation product frequencies F1 + F2, F1 - F2, and (2 F1) - F2, vs. the swept frequency F1. The IM product of interest may be selected using the Setplot command, and displayed with the print and plot commands. It is to be noted as in the harmonic analysis case, the results are the actual AC voltages and currents at the intermodulation frequencies, and need to be normalised with respect to .AC values to obtain the IM parameters.

If the DISTOF1 or DISTOF2 keywords are missing from the description of an independent source, then that source is assumed to have no input at the corresponding frequency. The default values of the magnitude and phase are 1.0 and 0.0 respectively. The phase should be specified in degrees.

It should be carefully noted that the number F2OVERF1 should ideally be an irrational number. Since this is not possible in practice, efforts should be made to keep the denominator in its fractional representation as large as possible, certainly above 3, for accurate results. That is, if F2OVERF1 is represented as a fraction A/B, where A and B are integers with no common factors, B should be as large as possible. Note that A < B because F2OVERF1 is constrained to be < 1.

To illustrate why, consider the cases where F2OVERF1 is 49/100 and 1/2. In a spectral analysis, the outputs produced are at F1 + F2, F1 - F2 and 2 F1 - F2. In the latter case, F1 - F2 = F2, so the result at the F1-F2 component is erroneous because there is the strong fundamental F2 component at the same frequency. Also, F1 + F2 = 2 F1 - F2 in the latter case, and each result is erroneous individually. This problem is not there in the case where F2OVERF1 = 49/100, because F1-F2 = 51/100 F1 < > 49/100 F1 = F2. In this case, there are two very closely spaced frequency components at F2 and F1 - F2. One of the advantages of the Volterra series technique is that it computes distortions at mix frequencies expressed symbolically (i.e. n F1 + m F2). Therefore one is able to obtain the strengths of distortion components accurately even if the separation between them is very small, as opposed to transient analysis for example. The disadvantage is of course that if two of the mix frequencies coincide, the results are not merged together and presented (though this could presumably be done as a post-processing step). Currently, the interested user should keep track of the mix frequencies and add the distortions at coinciding mix frequencies together, should it be necessary.

5.3.4 .NOISE: Noise Analysis

General form:

```
.NOISE V(OUTPUT <,REF>) SRC ( DEC | LIN | OCT ) PTS FSTART FSTOP
+ <PTS_PER_SUMMARY>
```

Examples:

```
.NOISE V(5) VIN DEC 10 1kHZ 100Mhz
.NOISE V(5,3) V1 OCT 8 1.0 1.0e6 1
```

The Noise line does a noise analysis of the circuit.

OUTPUT is the node at which the total output noise is desired; if REF is specified, then the noise voltage V(OUTPUT) - V(REF) is calculated. By default, REF is assumed to be ground. SRC is the name of an independent source to which input noise is referred. PTS, FSTART and FSTOP are .AC type parameters that specify the frequency range over which plots are desired. PTS_PER_SUMMARY is an optional integer; if specified, the noise contributions of each noise generator is produced every PTS_PER_SUMMARY frequency points. These are stored in the spectral density curves (use 'Setplot' command to select the correct set of curves.

The .NOISE control line produces two plots - one for the Noise Spectral Density curves and one for the total Integrated Noise over the specified frequency range. All noise voltages/currents are in units of V^2/Hz and A^2/Hz for spectral density, Vand A for integrated noise.

NOTE: The output vector units generated by WinSpice and Spice3 are different from Berkeley Spice2. If a pure Spice2 circuit is loaded into WinSpice, .PLOT and .PRINT lines will result in output in units of $V/Hz^{0.5}$ and $A/Hz^{0.5}$ or $V^{0.5}$ or $A^{0.5}$ to be compatible with Spice2 and to prevent confusion. If the input circuit contains .control/.endc lines or commands prefixed with *# this conversion is not made.

For examples, take the simple circuit below:-

```
simple resistor circuit
*simple resistor circuit to test which ones of vspice, uf77spice and
* spice3 give the correct results

iin 1 0 1m AC
rl 1 0 1k

.noise v(1) iin dec 10 10 100k 1
.print noise onoise
.end
```

A sample session showing how WinSpice3 stores the results is shown below.

```
Spice 1 -> run
Noise analysis ...
Spice 2 -> setplot
        Type the name of the desired plot:

        new     New plot
Current noise2  simple resistor circuit (Integrated Noise - V or A)
        noise1  simple resistor circuit (Noise Spectral Density
Curves - (V or A)^2/Hz
        const   Constant values (constants)
? noise1
Spice 3 -> display
Here are the vectors currently active:

Title: simple resistor circuit
Name: noise1 (Noise Spectral Density Curves - (V or A)^2/Hz
Date: Tue Aug 20 23:35:17  1996

    frequency           : frequency, real, 41 long, grid = xlog
[default scale]
    inoise_spectrum     : voltage, real, 41 long
    onoise_rl           : voltage, real, 41 long
    onoise_spectrum     : voltage, real, 41 long
Spice 4 ->
```

The onoise_rl plot contains the noise contributions of resistor rl at each frequency point. If the last option on the .noise line had been omitted, this vector would not have been created.

NOTE: in SPICE2 the syntax for .noise lines was different and SPICE2 required an .AC line to be present. The .AC line is not required for WinSpice3. For your information, to make the circuit above work on SPICE2, the .noise line above would need to be replaced by

.ac dec 10 10 100k
.noise v(1) iin 1

5.3.5 .OP: Operating Point Analysis

General form:

```
.OP
```

The inclusion of this line in an input file directs **WinSpice3** to determine the DC operating point of the circuit with inductors shorted and capacitors opened.

NOTE: a DC analysis is automatically performed prior to a transient analysis to determine the transient initial conditions, and prior to an AC small-signal, Noise, and Pole-Zero analysis to determine the linearized, small-signal models for non-linear devices (see the KEEPOPINFO in section 5.1).

5.3.6 .PZ: Pole-Zero Analysis

General form:

```
.PZ NODE1 NODE2 NODE3 NODE4 CUR POL
.PZ NODE1 NODE2 NODE3 NODE4 CUR ZER
.PZ NODE1 NODE2 NODE3 NODE4 CUR PZ
.PZ NODE1 NODE2 NODE3 NODE4 VOL POL
.PZ NODE1 NODE2 NODE3 NODE4 VOL ZER
.PZ NODE1 NODE2 NODE3 NODE4 VOL PZ
```

Examples:

```
.PZ 1 0 3 0 CUR POL
.PZ 2 3 5 0 VOL ZER
.PZ 4 1 4 1 CUR PZ
```

CUR stands for a transfer function of the type (output voltage)/(input current) while **VOL** stands for a transfer function of the type (output voltage)/(input voltage). **POL** stands for pole analysis only, **ZER** for zero analysis only and **PZ** for both. This feature is provided mainly because if there is a non-convergence in finding poles or zeros, then, at least the other can be found. Finally, **NODE1** and **NODE2** are the two input nodes and **NODE3** and **NODE4** are the two output nodes. Thus, there is complete freedom regarding the output and input ports and the type of transfer function.

In interactive mode (see section 6.9.30), the command syntax is the same except that the first field is PZ instead of .PZ. To print the results, one should use the command 'print all'.

5.3.7 .SENS: DC or Small-Signal AC Sensitivity Analysis

General form:

```
.SENS OUTVAR
.SENS OUTVAR AC DEC ND FSTART FSTOP
.SENS OUTVAR AC OCT NO FSTART FSTOP
.SENS OUTVAR AC LIN NP FSTART FSTOP
```

Examples:

```
.SENS V(1,OUT)
.SENS V(OUT) AC DEC 10 100 100k
.SENS I(VTEST)
```

The sensitivity of **OUTVAR** to all non-zero device parameters is calculated when the **SENS** analysis is specified. **OUTVAR** is a circuit variable (node voltage or voltage-source branch current).

The first form calculates sensitivity of the DC operating-point value of **OUTVAR**.

The second, third and fourth forms calculate sensitivity of the AC values of **OUTVAR**. The parameters listed for AC sensitivity are the same as in an AC analysis (see ".AC" above). The output values are in dimensions of change in output per unit change of input (as opposed to percent change in output or per percent change of input).

5.3.8 .TF: Transfer Function Analysis

General form:

```
.TF OUTVAR INSRC
```

Examples:

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

The TF line defines the small-signal output and input for the DC small-signal analysis. **OUTVAR** is the small signal output variable and **INSRC** is the small-signal input source. If this line is included, **WinSpice3** computes the DC small-signal value of the transfer function (output/input), input resistance, and output

resistance. For the first example, **WinSpice3** would compute the ratio of V(5, 3) to VIN, the small-signal input resistance at VIN, and the small-signal output resistance measured across nodes 5 and 3.

5.3.9 .TRAN: Transient Analysis

General form:

```
.TRAN TSTEP TSTOP <TSTART <TMAX>><UIC>
```

Examples:

```
.TRAN 1NS 100NS
.TRAN 1NS 1000NS 500NS
.TRAN 10NS 1US
```

TSTEP is the printing or plotting increment for line printer output. For use with the post processor, **TSTEP** is the suggested computing increment.

TSTOP is the final time, and **TSTART** is the initial time. If **TSTART** is omitted, it is assumed to be zero. The transient analysis always begins at time zero. In the interval <zero, TSTART>, the circuit is analysed (to reach a steady state), but no outputs are stored. In the interval <TSTART, TSTOP>, the circuit is analysed and outputs are stored.

TMAX is the maximum step size that **WinSpice3** uses. By default, the program chooses either **TSTEP** or **(TSTOP-TSTART)/50.0**, whichever is smaller. **TMAX** is useful when one wishes to guarantee a computing interval that is smaller than the printer increment, **TSTEP**.

UIC (use initial conditions) is an optional keyword that indicates that the user does not want **WinSpice3** to solve for the quiescent operating point before beginning the transient analysis. If this keyword is specified, **WinSpice3** uses the values specified using IC=... on the various elements as the initial transient condition and proceeds with the analysis. If the .IC control line has been specified, then the node voltages on the .IC line are used to compute the initial conditions for the devices. Look at the description on the .IC control line for its interpretation when **UIC** is not specified.

NOTE: **WinSpice3** uses a dynamic timestep algorithm where the timestep is varied according to the slope of the output curve. This helps to speed up analysis during parts of the curve that have small rates of change and concentrate the analysis where the rate of change is high. For this reason, the value of TSTEP is only used as a guide to the initial timestep.

A minimum timestep can be enforced in **WinSpice3** using the DELMIN or MINTIMESTEP system variables (see section 5.1 in this document).

5.4 Batch Output

These lines are ignored by the interactive **WinSpice3** and are only handled by the batch mode versions (cspice and bspice). They are provided for backward compatibility with **SPICE2**.

5.4.1 .SAVE Lines

General form:

```
.SAVE vector vector vector ...
```

Examples:

```
.SAVE i(vin) input output
.SAVE @m1[id]
.SAVE ALL
```

The vectors listed on the .SAVE line are recorded in the rawfile for use later with **WinSpice3**. The standard vector names are accepted.

If no .SAVE line is given, then the default set of vectors is saved (all node voltages and voltage source branch currents). If .SAVE lines are given, only those vectors specified are saved.

For more discussion on internal device data, see Appendix B. See also the section on the interactive command interpreter for information on how to use the rawfile. The interactive version of this statement is described in section 6.9.38.

5.4.2 .PRINT Lines

General form:

```
.PRINT PRTYPE OV1 <OV2 ... OV8>
```

Examples:

```
.PRINT TRAN V(4) I(VIN)
.PRINT DC V(2) I(VSRC) V(23, 17)
.PRINT AC VM(4, 2) VR(7) VP(8, 3)
```

The Print line defines the contents of a tabular listing of one to eight output variables. PRTYPE is the type of the analysis (DC, AC, TRAN, NOISE, or DISTO) for which the specified outputs are desired. **SPICE2** restricts the output variable to the following forms (though this restriction is not enforced by **WinSpice3**):

V(N1<,N2>)

specifies the voltage difference between nodes N1 and N2. If N2 (and the preceding comma) is omitted, ground (0) is assumed. For AC analysis, V(N1<,N2>) gives the magnitude of the complex voltage. For compatibility with **SPICE2**, the following five additional values can be accessed for the AC analysis by replacing the "V" in V(N1,N2) with:

V	magnitude (same as VM below)
VR	real part
VI	imaginary part
VM	magnitude
VP	phase (in radians or degrees - see the **units** variable description)
VDB	20 log10(magnitude)

I(VXXXXXXX)

specifies the current flowing in the independent voltage source named VXXXXXXX. Positive current flows from the positive node, through the source, to the negative node. For the AC analysis, the corresponding replacements for the letter I may be made in the same way as described for voltage outputs i.e.

I	magnitude (same as IM below)
IR	real part
II	imaginary part
IM	magnitude
IP	phase (in radians or degrees - see the **units** variable description)
IDB	20 log10(magnitude)

Output variables for the noise and distortion analyses have a different general form from that of the other analyses.

There is no limit on the number of .PRINT lines for each type of analysis.

5.4.3 .PLOT Lines

General form:

```
.PLOT PLTYPE OV1 <(PLO1, PHI1)> <OV2 <(PLO2, PHI2)> ... OV8>
```

Examples:

```
.PLOT DC V(4) V(5) V(1)
.PLOT TRAN V(17, 5) (2,5) I(VIN) V(17) (1,9)
.PLOT AC VM(5) VM(31, 24) VDB(5) VP(5)
.PLOT DISTO HD2 HD3(R) SIM2
.PLOT TRAN V(5,3) V(4) (0,5) V(7) (0,10)
```

The Plot line defines the contents of one plot of from one to eight output variables. **PLTYPE** is the type of analysis (**DC**, **AC**, **TRAN**, **NOISE**, or **DISTO**) for which the specified outputs are desired. The syntax for the **OV1** is identical to that for the .PRINT line and for the plot command in the interactive mode.

The letter X indicates the overlap of two or more traces on any plot.

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 all variables is desired, then a companion .PRINT line should be included.

There is no limit on the number of .PLOT lines specified for each type of analysis.

5.4.4 .FOUR: Fourier Analysis of Transient Analysis Output

General form:

```
.FOUR FREQ OV1 <OV2 OV3 ...>
```

Examples:

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

The Four (or Fourier) line controls whether **WinSpice3** performs a Fourier analysis as a part of the transient analysis. **FREQ** is the fundamental frequency, and **OV1** the desired output vector. The Fourier analysis is performed over the interval <TSTOP-period, TSTOP>, where TSTOP is the final time specified for the transient analysis, and period is one period of the fundamental frequency. The DC component and the first nine harmonics are determined. For maximum accuracy, **TMAX** (see the .TRAN line) should be set to period/100.0 (or less for very high-Q circuits).

6 INTERACTIVE INTERPRETER

WinSpice3 consists of a simulator and a front-end for data analysis and plotting. The command line interface has most of the capabilities of the UNIX C-shell.

WinSpice3 can plot data from a simulation on a graphics terminal or a workstation display. Note that the raw output file is different from the data that **SPICE2** writes to the standard output.

6.1 Command Interpretation

If a word is typed as a command, and there is no built-in command with that name, the directories in the **sourcepath** list (see section 6.2) are searched in order for the file. If it is found, it is read in as a command file (as if it had been loaded using the source command – see section 6.9.49).

Before it is read, however, the variable **argc** is set to the number of words following the filename on the command line, and **argv** is set to a list of those words. After the file is finished, these variables are unset. Note that if a command file calls another, it must save its **argv** and **argc** since they are altered. Also, command files may not be re-entrant since there are no local variables (of course, the procedures may explicitly manipulate a stack...). This way one can write scripts analogous to UNIX shell scripts for **WinSpice3**.

Note that for the script to work with **WinSpice3**, it must begin with a blank line (or whatever else, since it is thrown away) and then a line with **.control** on it. This is an unfortunate result of the source command being used for both circuit input and command file execution. Note also that this allows the user to merely type the name of a circuit file as a command and it is automatically run. The commands are executed immediately, without running any analyses that may be specified in the circuit (to execute the analyses before the script executes, include a **run** command in the script).

C-shell type quoting with "" and ", and backquote substitution may be used. Within single quotes, no further substitution (like history substitution) is done, and within double quotes, the words are kept together but further substitution is done. Any text between backquotes is replaced by the result of executing the text as a command to the shell.

If any command takes a filename, the filename must be enclosed in double quotes if the filename contains spaces (as is permitted in Windows long filenames).

You may type multiple commands on one line, separated by semicolons.

There are various command scripts installed in **\???\lib\scripts** (where **\???** is the directory containing the .EXE file), and the default **sourcepath** variable includes this directory, so you can use these command files (almost) like built-in commands. In fact, the **setplot** command is actually implemented as a script in this way.

6.2 Variables

The operation of **WinSpice3** may be affected by setting variables with the **set** command. In addition to the variables mentioned below, the **set** command in **WinSpice3** also affect the behaviour of the simulator via the options previously described under the section on ".OPTIONS".

Variables can contain text strings, numbers or be Boolean (i.e. have the meaning TRUE or FALSE). Variables can be defined and deleted with the **set** and **unset** commands (see later). String and number variables can be defined with a command of the form:-

```
set {variable}={value}
```

Boolean variables are a little odd in that they take the value TRUE if they are defined and FALSE if they do not exist. For example, the variable **slowplot** can be set to TRUE with the command

```
set slowplot
```

Setting a variable like **slowplot** to FALSE is done with the command

```
unset slowplot
```

To enter a list (e.g. the sourcepath variable), the list must be supplied within '(' and ')' e.g.

```
set sourcepath = ( . c:\mike\spice3f5 )
```

The variables in **WinSpice3** which may be altered by the **set** command are:

Variable	Type	Description
appendwrite	Boolean	Append to the file when a write command is issued, if one already exists.
colorN		These variables determine the colours used for plots. Colour 0 is the background, colour 1 is the grid and text colour, and colours 2 onwards are used in order for vector plots. On Unix/Linux, N may be in the range 0-15. The value of the colour variables should be names of colours, which may be found in the file /usr/lib/rgb.txt. In Windows, N can be in the range 0-19 and the available colour names are:- white black lt_red lt_green lt_blue lt_yellow lt_cyan lt_magenta red green blue yellow cyan magenta grey brown orange pink
cpdebug	Boolean	Print cshpar debugging information (must be complied with the -DCPDEBUG flag). Unsupported in the current release.
debug	Boolean	If set then a lot of debugging information is printed (must be compiled with the -DFTEDEBUG flag). Unsupported in the current release.
device	String	The name (/dev/tty??) of the graphics device. If this variable isn't set then the user's terminal is used. To do plotting on another monitor you probably have to set both the device and term variables. (If device is set to the name of a file, **WinSpice3** dumps the graphics control codes into this file -- this is useful for saving plots.)

Variable	Type	Description
diff_abstol	Real	The absolute tolerance used by the diff command.
diff_reltol	Real	The relative tolerance used by the diff command.
diff_vntol	Real	The absolute voltage tolerance used by the diff command.
echo	Boolean	Print out each command before it is executed.
exec_path	String	Path to the **WinSpice** executable.
filetype	String	This can be either **ascii** or **binary**, and determines the file format used by the **write** command (see section 6.9.66) is used. The default is **ascii**.
fourgridsize	Number	How many points to use for interpolating into when doing Fourier analysis.
gridsize	Number	If this variable is set to an integer, this number is used as the number of equally spaced points to use for the Y-axis when plotting. Otherwise the current scale is used (which may not have equally spaced points). If the current scale isn't strictly monotonic, then this option has no effect.
gridstyle	String	Sets the style of grid to be used. The possible values are:- **lingrid** **loglog** **xlog** **ylog** **smith** **smithgrid** **polar** **nogrid** See the **plot** command for details.
hcopydev	String	If this is set, when the **hardcopy** command is run the resulting file is automatically printed on the printer named **hcopydev** with the command **lpr -Phcopydev -g file**.
hcopyfont	String	This variable specifies the font name for hardcopy output plots. The value is device dependent.
hcopyfontsize	String	The font size for hardcopy plots.
hcopyfontscale	String	This is a scaling factor for the font used in hardcopy plots.
hcopydevtype	String	This variable specifies the type of the printer output to use in the **hardcopy** command. If **hcopydevtype** is not set, **plot(5)** format is assumed. The standard distribution currently recognises postscript as an alternative output format. When used in conjunction with **hcopydev**, **hcopydevtype** should specify a format supported by the printer.

Variable	Type	Description
height	Number	The length of the page for **asciiplot** and **print col**.
history	Number	The number of events to save in the history list.
lprplot5	String	This is a **printf(3s)** style format string used to specify the command to use for sending **plot(5)**-style plots to a printer or plotter. The first parameter supplied is the printer name, the second parameter supplied is a file name containing the plot. Both parameters are strings. It is trivial to cause WinSpice3 to abort by supplying an unreasonable format string.
lprps	String	This is a **printf(3s)** style format string used to specify the command to use for sending PostScript plots to a printer or plotter. The first parameter supplied is the printer name, the second parameter supplied is a file name containing the plot. Both parameters are strings. It is trivial to cause WinSpice3 to abort by supplying a unreasonable format string.
maxcircuits	Number	The maximum number of circuits that WinSpice3 will store. When a circuit is opened with the **source** command, and the number of circuits exceeds this value, the earliest circuit is deleted. A list of circuits in the system can be displayed using the **setcirc** command (see section 6.9.41 for details).
maxplots	Number	The maximum number of plot windows that can be open. As new plots are created, old ones are closed. If maxplots is zero, automatic closure of plot windows is disabled. If maxplots is not defined, a maximum of 10 plots can be displayed.
nfreqs	Number	The number of frequencies to compute in the **fourier** command. (Defaults to 10.)
nobreak	Boolean	Don't have **asciiplot** and **print col** break between pages.
noasciiplotvalue	Boolean	Don't print the first vector plotted to the left when doing an asciiplot.
noclobber	Boolean	Don't overwrite existing files when doing IO redirection.
noglob	Boolean	Don't expand the global characters `*', `?', `[', and `]'. This is the default.
nogrid	??? Does nothing!	Don't plot a grid when graphing curves (but do label the axes).
nomoremode	Boolean	If **nomoremode** is not set, whenever a large amount of data is being printed to the screen (e.g., the **print** or **asciiplot** commands), the output is stopped every screenful and continues when a carriage return is typed. If **nomoremode** is set then data scrolls off the screen without check.

Variable	Type	Description
nonomatch		If **noglob** is unset and a global expression cannot be matched, use the global characters literally instead of complaining.
nosort	Boolean	Don't have **display** sort the variable names.
nopadding	Boolean	If TRUE, enables rawfile padding.
noprintscale	Boolean	Don't print the scale in the leftmost column when a **print col** command is given.
numdgt	??? Does nothing!	The number of digits to print when printing tables of data (**fourier**, **print col**). The default precision is 6 digits. Approximately 16 decimal digits are available using double precision, so **numdgt** should not be more than 16. If the number is negative, one fewer digit is printed to ensure constant widths in tables.
plotstyle	String	This should be one of **linplot**, **combplot**, or **pointplot**:chars. **linplot**, the default, causes points to be plotted as parts of connected lines. **combplot** causes a comb plot to be done (see the description of the **combplot** variable above). **pointplot** causes each point to be plotted separately - the chars are a list of characters that are used for each vector plotted. If they are omitted then a default set is used.
pointchars	String	A string of characters to be used when **plotstyle** is set to **pointplot** or the **pointplot** keyword is used in the **plot** command. If not defined, and internal set of characters is used.
polydegree	Number	The degree of the polynomial that the **plot** command should fit to the data. If polydegree is N, then WinSpice3 fits a degree N polynomial to every set of N points and draw 10 intermediate points in between each endpoint. If the points aren't monotonic, then it tries rotating the curve and reducing the degree until a fit is achieved.
polysteps	Number	The number of points to interpolate between every pair of points available when doing curve fitting. The default is 10.
printinfo	???	???
program	String	The name of the current program (argv[0]).
prompt	String	The prompt, with the character `!' replaced by the current event number.
rawfile	String	The default name for rawfiles created.
remote_shell	String	Overrides the name used for generating rspice runs (default is "rsh").
slowplot	Boolean	Stop between each graph plotted and wait for the user to type return before continuing.

Variable	Type	Description
sourcepath	List	A list of the directories to search when a source command is given. The default is the current directory and the standard SPICE library (/usr/local/lib/spice, or whatever LIBPATH is #defined to in the WinSpice3 source).
term	String	The mfb name of the current terminal.
ticmarks	Number or Boolean	If this variable is defined with a numerical value n e.g. 'set ticmarks=5', then every nth point on a plot is marked with a character. If defined as a Boolean variable i.e. with no number supplied, a ticmark is printed every 10 plot points.
ticdata	???	???
units	String	If set to **degrees**, then all the trig functions will use degrees instead of radians. This also means that the ph() operator for phase also give a phase angle in degrees.
unixcom	Boolean	If this variable is defined, the interactive command line will attempt to run a program with the same name.
verbose	Boolean	Be verbose. This is midway between **echo** and **debug**/**cpdebug**.
width	Number	The width of the page for **asciiplot** and **print col**.
x11lineararcs	Boolean	Some X11 implementations have poor arc drawing. If you set this option, WinSpice3 will plot using an approximation to the curve using straight lines.
xbrushheight		The height of the brush to use if X is being run.
xbrushwidth	Number	The width of the brush to use if X is being run.
xfont		The name of the X font to use when plotting data and entering labels. The plot may not look good if this is a variable-width font.

There are several set variables that **WinSpice3** uses. They are:

Variable	Type	Description
editor	String	The command used to start the circuit editor.. Used by the edit command.
modelcard	String	The name of the model card (normally **.model**).
modelline	String	The name of the model card (normally **.model**). Same as **modelcard**.

Variable	Type	Description
noaskquit	Boolean	Do not check to make sure that there are no circuits suspended and no plots unsaved. Normally WinSpice3 warns the user when he tries to quit if this is the case.
nobjthack	Boolean	Assume that BJTs have 4 nodes.
noparse	Boolean	Don't attempt to parse input files when they are read in (useful for debugging). Of course, they cannot be run if they are not parsed.
nosubckt	Boolean	Don't expand subcircuits.
renumber	Boolean	Renumber input lines when an input file has **.include**'s.
subend	String	The card to end subcircuits (normally **.ends**)
subinvoke	String	The prefix to invoke subcircuits (normally x).
Substart	String	The card to begin subcircuits (normally **.subckt**)

6.3 Variable Substitution

The values of variables may be used in commands by writing **$varname** where the value of the variable is to appear.

The special variables **$$** and **$<** refer to the process ID of the program and a line of input which is read from the terminal when the variable is evaluated, respectively.

If a variable has a name of the form **$&word**, then **word** is considered a vector (see above), and its value is taken to be the value of the variable.

If **$foo** is a valid variable, and is of type list, then the expression **$foo[low-high]** represents a range of elements. Either the upper index or the lower may be left out, and the reverse of a list may be obtained with **$foo[len-0]**. Also, the notation **$?foo** evaluates to 1 if the variable **foo** is defined, 0 otherwise, and **$#foo** evaluates to the number of elements in **foo** if it is a list, 1 if it is a number or string, and 0 if it is a boolean variable.

6.4 Redirection

IO redirection is available in the same was as is found in the MSDOS and UNIX command shells as follows:-

	Description
> **file**	Sends standard output to **file**. If the file already exists, it is truncated to zero length and its contents discarded. If it doesn't exist, it is created.
>> **file**	Appends standard output to **file**. If the file already exists, the output is added to the end of the file. If it doesn't exist, it is created.
>**& file**	Sends standard output and standard error streams to **file**. If the file already exists, it is truncated to zero length and its contents discarded. If it doesn't exist, it is created.
>>**&**	Appends standard output and standard error streams to **file**. If the file already exists, the output is added to the end of the file. If it doesn't exist, it is created.
< **file**	Takes standard input from the file **file**

6.5 Vectors & Scalars

WinSpice3 data is in the form of vectors: time, voltage, etc. Each vector has a type, and vectors can be operated on and combined algebraically in ways consistent with their types. Vectors are normally created when a data file is read in (see the **load** command in section 6.9.25), and when the initial datafile is loaded. They can also be created with the **let** command (see section 6.9.22).

A scalar is a vector of length 1.

A vector may be either the name of a vector already defined or a floating-point number (a scalar). A number may be written in any format acceptable to SPICE, such as **14.6Meg** or **-1.231e-4**. Note that you can either use scientific notation or one of the abbreviations like MEG or G, but not both. As with SPICE, a number may have trailing alphabetic characters after it.

The notation **expr [num]** denotes the num'th element of expr. For multi-dimensional vectors, a vector of one less dimension is returned. Also for multi-dimensional vectors, the notation **expr[m][n]** will return the nth element of the mth subvector. To get a subrange of a vector, use the form **expr[lower, upper]**.

To reference vectors in a plot that is not the current plot (see the setplot command, below), the notation **plotname.vecname** can be used.

Either a plotname or a vector name may be the wildcard **all**. If the plotname is **all**, matching vectors from all plots are specified, and if the vector name is **all**, all vectors in the specified plots are referenced.

Vector names in SPICE may have a name such as **@name[param]**, where **name** is either the name of a device instance or model. This denotes the value of the **param** parameter of the device or model. See Appendix B for details of what parameters are available. The value is a vector of length 1. This function is also available with the **show** command, and is available with variables for convenience for command scripts.

6.5.1 Expressions

An expression is an algebraic formula involving vectors and scalars and the following operations:

```
+ - * / ^ %
```

% is the modulo operator, and the comma operator has two meanings: if it is present in the argument list of a user-definable function, it serves to separate the arguments. Otherwise, the term **x , y** is synonymous with **x + j(y)**.

Also available are the logical operations **&** (and), **|** (or), **!** (not), and the relational operations <, >, >=, <=, =, and <> (not equal). If used in an algebraic expression they work like they would in C, producing values of 0 or 1. The relational operators have the following synonyms:

gt	>
lt	<
ge	>=
le	<=
ne	<>
eq	=
and	&
or	|
not	!

These are useful when < and > might be confused with IO redirection (which is almost always).

Note that you may not use binary operations on expressions involving wildcards - it is not obvious what **all + all** should denote, for instance.

Thus some (contrived) examples of expressions are:

```
cos(TIME) + db(v(3))
sin(cos(log([1 2 3 4 5 6 7 8 9 10])))
TIME * rnd(v(9)) - 15 * cos(vin#branch) ^ [7.9e5 8]
not ((ac3.FREQ[32] & tran1.TIME[10]) gt 3)
```

6.5.2 Functions

The following functions are available:

Function	Description
mag(vector) **magnitude(vector)**	The result is a REAL vector with each element containing the magnitude of each element in the COMPLEX vector **vector**.
ph(vector) **phase(vector)**	The result is a REAL vector with each element containing the phase of each element in the COMPLEX vector **vector**. If the **units** variable is not defined, the phase is in radians. If **units** has the value of **degrees**, the phase angle will be in degrees.
j(vector)	i (sqrt(-1)) times COMPLEX vector **vector**.
real(vector) **re(vector)**	The real component of vector
imag(vector) **im(vector)**	The imaginary part of vector
db(vector)	20 log10(mag(vector))
log(vector) **log10(vector)**	The logarithm (base 10) of vector
ln(vector)	The natural logarithm (base e) of vector
exp(vector)	e to the vector power
abs(vector)	The absolute value of vector i.e. the magnitude. Same as mag(vector).
sqrt(vector)	The square root of vector.
sin(vector)	The sine of vector.
cos(vector)	The cosine of vector.
tan(vector)	The tangent of vector.
atan(vector)	The inverse tangent of vector.
norm(vector)	The vector normalised to 1 (i.e., the largest magnitude of any component is 1).
rnd(vector)	A vector with each component a random integer between 0 and the absolute value of the vector's corresponding component.
pos(vector)	The result is a vector containing 1.0 if the corresponding element of **vector** was >0.0 and zero otherwise.

Function	Description
mean(vector)	The result is a scalar (a length 1 vector) that is the mean of the elements of the vector.
sum(vector)	The result is a scalar (a length 1 vector) that is the sum of the elements of the vector.
vector(number)	The result is a vector of length **number**, with elements 0, 1, ... **number - 1**. If number is a vector then just the first element is taken, and if it isn't an integer then the floor of the magnitude is used.
unitvec(vector)	The result is a vector with each component set to 1.0. The length of the resultant vector is the value of the first number in **vector**. If **vector** was complex, the length is mag(vector[0]).
length(vector)	The result is a scalar (a length 1 vector) that is the length of the vector **vector**.
gd(vector)	The result is a vector which contains the group delay of complex vector **vector**.
rad(vector)	Perform degree-radian conversion on vector **vector**.
deg(vector)	Perform radian-degree conversion on vector **vector**.
interpolate(plot.vector)	The result of interpolating the named vector onto the scale of the current plot. This function uses the variable **polydegree** to determine the degree of interpolation.
deriv(vector)	Calculates the derivative of the given vector. This uses numeric differentiation by interpolating a polynomial and may not produce satisfactory results (particularly with iterated differentiation). The implementation only calculates the derivative with respect to the real component of that vector's scale.

6.5.3 Constants

There are a number of pre-defined scalar constants in WinSpice3 which can be used in expressions. They are:

Constant	Description
true	The value 1
yes	The value 1
no	The value 0
false	The value 0
pi	π (3.14159...)
e	The base of natural logarithms (2.71828...)
c	The speed of light (299,792,500 m/sec)
i	The square root of –1 i.e. (0, 1)
kelvin	Absolute 0 in Centigrade (-273.15 °C)
echarge	The charge on an electron (1.6021918e-19 C)
boltz	Boltzman's constant (1.3806226e-23)
planck	Planck's constant (h = 6.626200e-34)

These are all in MKS units. If you have another variable with a name that conflicts with one of these then it takes precedence.

6.6 History Substitutions

A history substitution enables you to reuse a portion of a previous command as you type the current command. History substitutions save typing and also help reduce typing errors.

A history substitution normally starts with a '!'. A history substitution has three parts: an *event* that specifies a previous command, a *selector* that selects one or more word of the event, and some *modifiers* that modify the selected words. The selector and modifiers are optional. A history substitution has the form

```
![event][[:]selector[:modifier] . . .]
```

The event is required unless it is followed by a selector that does not start with a digit. The ':' can be omitted before *selector* if *selector* does not begin with a digit.

History substitutions are interpreted before anything else – even before quotations and command substitutions. The only way to quote the '!' of a history substitution is to escape it with a preceding backslash. A '!' need not be escaped , however, if it is followed by whitespace, '=', or '('.

6.6.1 Events and Their Specifications

WinSpice3 saves each command that you type on a history list provided that the command contains at least one word. The commands on the history list are called events. The events are numbered, with the first command that you issue when you start **WinSpice** being number one. For complex commands such as 'for' that consist of more than one line, only the first line makes its way to the history list. The **history** variable

specified how many events are retained on the history list. You can view the history list with the **history** command (see section 6.9.20 on Page 82).

These are the forms of an event in a history substitution:

!! The preceding event. Typing '!!' is an easy way to reissue the previous command.

!*n* Event number *n*.

!-*n* The *n*th previous event. For example, '!-1' refers to the immediately preceding event and is equivalent to '!!'.

!*str* The unique previous event whose name starts with *str*.

!?*str*? The unique previous event containing the string *str*. The closing '?' can be omitted if it is followed by a newline.

6.6.2 Selectors

You can select a subset of the words of an event by attaching a *selector* to the event. A history substitution without a selector includes all of the words of the event. These are the possible selectors for selecting words of the event:

:0 The command name.

[:]^ The first argument.

[:]$ The last argument.

:n The nth argument ($n \geq 1$)

:n_1-n_2 Words n_1 through n_2

[:]* Words 1 through $

:x* Words x through $

:x- Words x through ($ - 1)

[:]-x Words 0 through x

[:]% The word matched by the preceding '?str?' search

The colon preceding a selector can be omitted if the selector does not start with a digit.

6.6.3 Modifiers

You can modify the words of an event by attaching one or more modifiers. Each modifier must be preceded by a colon.

The following modifiers assume that the first selected word is a file name:

:r Removes the trailing '.str' extension from the first selected word.

:h Removes a trailing path name component from the first selected word.

:t Removes all leading path name components from the first selected word.

For example, if the command

```
ls –l /usr/elsa/toys.txt
```

has just been executed, then the command

```
echo !!^:r !!^:h !!^:t !!^:t:r
```

produces the output

```
/usr/else/toys /usr/elsa toys.txt toys
```

The following modifiers enable you to substitute within the selected words of an event. If the modifier includes ‘g’, the substitution applies to the entire event; otherwise it applies only to the first modifiable word.

:[g]s/*l*/*r* Substitutes the string *r* for the string *l*. The delimiter ‘/’ may be replaced by any other delimiting character. Within the substitution, the delimiter can be quoted by escaping it with ‘\’. If *l* is empty, the most recently used string takes its place – either a previous *l* or the string *str* in an event selector of the form ‘!?str?’. The closing delimiter can be omitted if it is followed by a newline.

:[g]& Repeats the previous substitution.

The following modifiers quote the selected words, possibly after earlier substitutions:

:q Quotes the selected words, preventing further substitutions.

:x Quotes the selected words but breaks the selected text into words at whitespace.

:p Shows (“prints”) the new command but doesn’t execute it.

6.6.4 Special Conventions

The following additional special conventions provide abbreviations for commonly used forms of history substitution:

- An event specification can be omitted from a history substitution if it is followed by a selector that does not start with a digit. In this case the event is taken to be the event used in the most recent history reference on the same line if there is one, or the preceding event otherwise. For example, the command

  Echo !?quetzal?^ !$

  echoes the first and last arguments of the most recent command containing the string ‘quetzal’.

- If the first nonblank character of an input line is ‘^’, the ‘^’ is taken as an abbreviation for ‘!:s^’. This form provides a convenient way to correct a simple spelling error in the previous line. For example, if by mistake you typed the command

  cat /etc/lasswd

  you could re-execute the command with ‘lasswd’ changed to ‘passwd’ by typing

  ^l^p

- You can enclose a history substitution in braces to prevent it from absorbing the following characters. In this case the entire substitution except for the starting ‘!’ must be within the braces. For example, suppose that you previously issued the command

  cp accounts ../money

Then the command '!cps' looks for a previous command starting with 'cps' while the command '!{cp}s' turns into a command

cp accounts ../moneys

6.7 Filename Expansions

The characters ~, {, and } have the same effects as they do in the C-Shell, i.e., home directory and alternative expansion. It is possible to use the wildcard characters *, ?, [, and] also, but only if you **unset noglob** first. This makes them rather useless for typing algebraic expressions, so you should **set noglob** again after you are done with wildcard expansion. Note that the pattern **[^abc]** matches all characters *except* a, b, *and* c.

6.8 Control Structures

6.8.1 While - End

General Form

```
while condition
    statement
    ...
end
```

While condition, an arbitrary algebraic expression, is true, execute the statements.

The condition is an expression involving vector and scalar variables (see sections 6.5 and 6.5.1).

For example, the following will sweep a resistor value:-

```
.control
echo ********************************************************
echo Sweep altering R1 directly
echo ********************************************************
let res = 1
while res <= 100
    alter @r1[resistance] = res
    op
    print v(1) v(2)
    let res = res + 5
end
.endc
```

6.8.2 Repeat - End

General Form

```
repeat [number]
    statement
    ...
end
```

Execute the statements number times, or forever if no argument is given.

6.8.3 Dowhile - End

General Form

```
dowhile condition
    statement
    ...
end
```

The same as while, except that the condition is tested after the statements are executed.

The condition is an expression involving vector and scalar variables (see sections 6.5 and 6.5.1).

6.8.4 Foreach - End

General Form

```
foreach var value ...
    statement
    ...
end
```

The statements are executed once for each of the values in the list, each time with the variable ‘var’ set to the current one. ‘var’ can be accessed by the $var notation (see sections 6.2 and 6.3 for details).

6.8.5 If - Then - Else

General Form

```
if condition
    statement
    ...
else
    statement
    ...
end
```

If the condition is non-zero then the first set of statements are executed, otherwise the second set. The **else** and the second set of statements may be omitted.

The condition is an expression involving vector and scalar variables (see sections 6.5 and 6.5.1).

6.8.6 Label

General Form

```
label word
```

If a statement of the form **goto word** is encountered, control is transferred to this point, otherwise this is a no-op.

6.8.7 Goto

General Form

```
goto word
```

If a statement of the form label word is present in the block or an enclosing block, control is transferred there. Note that if the label is at the top level, it must be before the goto statement (i.e., a forward goto may occur only within a block).

6.8.8 Continue

General Form

```
continue
```

If there is a **while**, **dowhile**, or **foreach** block enclosing this statement, control passes to the test, or in the case of foreach, the next value is taken. Otherwise an error results.

6.8.9 Break

General Form

```
break
```

If there is a **while**, **dowhile**, or **foreach** block enclosing this statement, control passes out of the block. Otherwise an error results.

Of course, control structures may be nested. When a block is entered and the input is the terminal, the prompt becomes a number of >'s corresponding to the number of blocks the user has entered. The current control structures may be examined with the debugging command cdump.

6.9 Commands

6.9.1 Ac: Perform an AC frequency response analysis

General Form

```
ac ( DEC | OCT | LIN ) N Fstart Fstop
```

Do an AC analysis. See section 5.3.1 of this manual for more details.

6.9.2 Alias: Create an alias for a command

General Form

```
alias [word] [text ...]
```

Causes **word** to be aliased to **text**. History substitutions may be used, as in C-shell aliases.

6.9.3 Alter: Change a device or model parameter

General Form

```
alter name = expression
alter name parameter = expression
alter @name[parameter] = expression
```

Alter changes the value for a device or a specified parameter of a device or model. The first form is used by simple devices which have one principal value (resistors, capacitors, etc.) where the second and third forms are for more complex devices (BJTs, etc.).

If ‘name’ is the name of a device instance then the command will change a parameter within an individual device instance e.g.

```
alter @m1[temp] = 273
```

If ‘name’ is the name of a model then the command will change a model parameter and this will affect all device instances in the circuit which use this model e.g.

```
alter @nmos[lambda] = 3
```

For specifying vectors as expressions, start the vector with "[", followed by the values in the vector, and end with "]". Be sure to place a space between each of the values and before and after the "[" and "]" e.g.

```
alter @vin[pulse] = [ 0 5 10n 10n 10n 50n 100n ]
```

Lists of alterable parameters for each device model is given in section 10. Only these parameters will be accepted for a given device.

6.9.4 Asciiplot: Plot values using old-style character plots

General Form

```
asciiplot plotargs
```

Produce a line printer plot of the vectors. The plot is sent to the standard output, so you can put it into a file with asciiplot args ... > file. The **set** options **width**, **height**, and **nobreak** determine the width and height of the plot, and whether there are page breaks, respectively. Note that you will have problems if you try to

asciiplot something with an X-scale that isn't monotonic (i.e., something like sin(TIME)), because asciiplot uses a simple-minded linear interpolation.

6.9.5 Bug: Mail a bug report

General Form

```
bug
```

Send a bug report. Please include a short summary of the problem, the version number and name of the operating system that you are running, the version of SPICE that you are running, and the relevant SPICE input file. If you have defined **BUGADDR**, the mail is delivered to there.

NOTE: this command does not work yet – but it seems too useful to take out! Future versions of WinSpice3 will use this command to email bug reports to the author.

6.9.6 Cd: Change directory

General Form

```
cd [directory]
```

Change the current working directory to directory. Displays the current directory if [directory] is not given.

6.9.7 Cross: Create a new vector

General Form

```
cross vecname n [vector1 vector2 ...]
```

Create a new vector **vecname**.from index **n** in each of the input vectors. n=0 selects the first item in each vector. If any input vector is complex then the output vector will be complex.

The index value **n** may be a constant or a vector. If **n** is not scalar, only the first value in the vector is used. If **n** is a complex vector, only the real part is used.

This command can be used to get the nth value in a vector e.g.

```
cross val 5 v(3)

let index = 5
cross val index v(3)
```

Both of the above are equivalent. The second example uses scalar vector 'index' to fetch the 6^{th} item in vector v(3).

6.9.8 Dc: Perform a DC-sweep analysis

General Form

```
dc Source-Name Vstart Vstop Vincr [Source2 Vstart2 Vstop2 Vincr2]
```

Do a DC transfer curve analysis. See section 5.3.2 of this manual for more details.

6.9.9 Define: Define a function

General Form

```
define function(arg1, arg2, ...) expression
```

Define the user-definable function with the name function and arguments arg1, arg2, ... to be expression, which may involve the arguments. When the function is later used, the arguments it is given are substituted for the formal arguments when it is parsed. If expression is not present, any definition for function is

printed, and if there are no arguments to define then all currently active definitions are printed. Note that you may have different functions defined with the same name but different arities.

Some useful definitions are:

```
define max(x,y) (x > y) * x + (x <= y) * y
define min(x,y) (x < y) * x + (x >= y) * y
```

6.9.10 Delete: Remove a trace or breakpoint

General Form

```
delete [ debug-number ... ]
```

Delete the specified breakpoints and traces. The **debug-numbers** are those shown by the status command (unless you do **status > file**, in which case the debug numbers are not printed).

6.9.11 Destroy: Delete a data set (plot)

General Form

```
destroy [plotnames | all]
```

Release the memory holding the data for the specified runs.

The command 'destroy all' also resets plot numbering back to 1 such that running an AC analysis, say, after 'destroy all' always generates the 'ac1' plot vector. This is useful if .cir files contain plot lines which address explicit plot names like 'tran1.v(6)' because if other circuits are run first then the plot numbering may be changed.

6.9.12 Diff: Compare vectors

General Form

```
diff plot1 plot2 [vec ...]
```

Compare all the vectors in the specified plots, or only the named vectors if any are given. There are different vectors in the two plots, or any values in the vectors differ significantly the difference is reported. The variable **diff_abstol**, **diff_reltol**, and **diff_vntol** are used to determine a significant difference.

6.9.13 Display: List known vectors and types

General Form

```
display [varname ...]
```

Prints a summary of currently defined vectors, or of the names specified. The vectors are sorted by name unless the variable **nosort** is set. The information given is the name of the vector, the length, the type of the vector, and whether it is real or complex data. Additionally, one vector is labelled **[scale]**. When a command such as plot is given without a 'vs' argument, this scale is used for the X-axis. It is always the first vector in a rawfile, or the first vector defined in a new plot. If you undefine the scale (i.e., let TIME = []), one of the remaining vectors becomes the new scale (which is undetermined).

6.9.14 Disto: Perform a distortion analysis

General Form

```
disto DEC ND FSTART FSTOP <F2OVERF1>
disto OCT NO FSTART FSTOP <F2OVERF1>
disto LIN NP FSTART FSTOP <F2OVERF1>
```

Examples:

```
disto dec 10 1kHz 100Mhz
disto dec 10 1kHz 100Mhz 0.9
```

The command line form of the .DISTO directive. See section 5.3.3 for details.

6.9.15 Echo: Print text

General Form

```
echo [text...]
```

Echoes the given text to the screen.

6.9.16 Edit: Edit the current circuit

General Form

```
edit [file]
```

Open the current **WinSpice3** input file in the editor and allow the user to modify it. While the editor is running, **WinSpice3** watches for the original file to be updated and, if so, reads the file back in. If a filename is given, then edit that file and load it, making the circuit the current one.

By default, Windows Notepad is used. This can be changed by setting the environment variable **editor** (see section 6.2) e.g.

```
WinSpice3 18 -> set editor="c:\program files\accessories\wordpad.exe"
WinSpice3 19 -> edit
```

6.9.17 Fourier: Perform a fourier transform

General Form

```
fourier fundamental_frequency [value ...]
```

Does a fourier analysis of each of the given values, using the first 10 multiples of the fundamental frequency (or the first **nfreq**s, if that variable is set - see below). The output is like that of the .four **WinSpice3** line. The values may be any valid expression. The values are interpolated onto a fixed-space grid with the number of points given by the **fourgridsize** variable, or 200 if it is not set. The interpolation is of degree **polydegree** if that variable is set, or 1. If **polydegree** is 0, then no interpolation is done. This is likely to give erroneous results if the time scale is not monotonic, though.

6.9.18 Hardcopy: Save a plot to a file for printing

General Form

```
hardcopy file plotargs
```

Just like **plot**, except creates a file called **file** containing the plot. The file is an image in plot(5) format, and can be printed by either the **plot(1)** program or **lpr** with the **-g** flag.

6.9.19 Help: Print summaries of WinSpice3 commands

General Form

```
help [all] [command ...]
```

Prints help. If the argument ‘all’ is given, a short description of everything you could possibly type is printed. If commands are given, descriptions of those commands are printed. Otherwise help for only a few major commands is printed.

6.9.20 History: Review previous commands

General Form

```
history [number]
```

Print out the history, or the last number commands typed at the keyboard.

6.9.21 Iplot: Incremental plot

General Form

```
iplot [node ...]
```

Example

```
iplot v(1) v(2)
```

Incrementally plot the values of the nodes while **WinSpice3** runs. The **iplot** command can be used with the **where** command to find trouble spots in a transient simulation.

The **iplot** command adds a form of visual trace to the circuit. See the **trace** command (section 6.9.56) for a different type of trace that is available.

Several iplot commands may be active at once. Iplotting is not applicable for all analyses. To remove an iplot trace entry, use the **delete** command (see section 6.9.10). To display a list of iplots, use the **status** command (see section 6.9.51).

6.9.22 Let: Assign a value to a vector

General Form

```
let name = expr
```

Creates a new vector called **name** with the value specified by **expr**, an expression as described above. If **expr** is **[]** (a zero-length vector) then the vector becomes undefined. Individual elements of a vector may be modified by appending a subscript to name (ex. **name[0]**).

A vector variable can be used within the scripting language like variables in other languages. For example:-

```
.control
    destroy all
    let ii = 0
    while ii < 2
        alter r1 = 10k + 10k * ii
        ac dec 10 1 10k
        let ii = ii + 1
    end
    plot db(ac1.v(2)) db(ac2.v(2))
.endc
v1 1 0 dc 0 ac 1
r1 1 2 1k
c1 2 0 1uf
.end
```

In the example shown, ‘ii’ is a single-element vector (scalar) used as a loop counter.

6.9.23 Linearize: Interpolate to a linear scale

General Form

```
linearize [vec ...]
```

Create a new plot with all of the vectors in the current plot, or only those mentioned if arguments are given. The new vectors are interpolated onto a linear time scale, which is determined by the values of tstep, tstart, and tstop in the currently active transient analysis. The currently loaded input file must include a transient analysis (a tran command may be run interactively before the last reset, alternately), and the current plot must be from this transient analysis.

This command is needed because **WinSpice3** doesn't output the results from a transient analysis in the same manner that **SPICE2** did. WinSpice3 uses a dynamic timestep which means that the timescale is non-monotonic. **SPICE2** internally does the same, but it does an automatic linearization – the non-linearized result is not normally available.

6.9.24 Listing: Print a listing of the current circuit

General Form

```
listing [logical] [physical] [deck] [expand]
```

If the logical argument is given, the listing is with all continuation lines collapsed into one line, and if the physical argument is given the lines are printed out as they were found in the file. The default is logical. A deck listing is just like the physical listing, except without the line numbers it recreates the input file verbatim (except that it does not preserve case). If the word expand is present, the circuit is printed with all subcircuits expanded.

6.9.25 Load: Load rawfile data

General Form

```
load [filename] ...
```

Loads either binary or ASCII format rawfile data from the files named. The default filename is **rawspice.raw**, or the argument to the **-r** flag if there was one.

6.9.26 Noise: Perform a noise analysis

General Form

```
noise V(OUTPUT <,REF>) SRC ( DEC | LIN | OCT ) PTS FSTART FSTOP
+ <PTS_PER_SUMMARY>
```

See section 5.3.4 for details of this command.

6.9.27 Op: Perform an operating point analysis

General Form

```
op
```

Do an operating point analysis. See section 5.3.5 of this manual for more details.

6.9.28 Plot: Plot values on the display

General Form

```
plot exprs [ylimit ylo yhi] [xlimit xlo xhi] [xindices xilo xihi]
[xcompress comp] [xdelta xdel] [ydelta ydel] [xlog] [ylog] [loglog]
[vs xname] [xlabel word] [ylabel word] [title word] [samep]
[linear] [linplot | combplot | pointplot]
```

Plot the given **exprs** on the screen (if you are on a graphics terminal). The **xlimit** and **ylimit** arguments determine the high and low x- and y-limits of the axes, respectively. The **xindices** arguments determine what range of points are to be plotted - everything between the **xilo**'th point and the **xihi**'th point is plotted. The **xcompress** argument specifies that only one out of every **comp** points should be plotted. If an **xdelta** or a **ydelta** parameter is present, it specifies the spacing between grid lines on the X- and Y-axis. These parameter names may be abbreviated to **xl**, **yl**, **xind**, **xcomp**, **xdel**, and **ydel** respectively.

The **xname** argument is an expression to use as the scale on the x-axis. If **xlog** or **ylog** are present then the X or Y scale, respectively, is logarithmic (**loglog** is the same as specifying both). The **xlabel** and **ylabel** arguments cause the specified labels to be used for the X and Y axes, respectively.

If **samep** is given, the values of the other parameters (other than **xname**) from the previous **plot**, **hardcopy**, or **asciiplot** command is used unless re-defined on the command line.

The **title** argument is used in the place of the plot name at the bottom of the graph.

The **linear** keyword is used to override a default log-scale plot (as in the output for an AC analysis).

Different styles of plot can be selected via the **linplot**, **pointplot** and **combplot** keywords. Specifying **linplot** gives a plot where each point is connected to the next by a line. If **pointplot** is used, the points are represented by a character with no joining lines. The **combplot** is drawn with a vertical line from each point to the X-axis. The plot type can also be specified via the plotstyle variable e.g. '**set plotstyle=combplot**'. The if a plot style is given in the plot command, this overrides the variable.

Finally, the keyword **polar** to generate a polar plot. To produce a smith plot, use the keyword **smith**. Note that the data is transformed, so for smith plots you will see the data transformed by the function (x-1)/(x+1). To produce a polar plot with a smith grid but without performing the smith transform, use the keyword **smithgrid**.

If **maxplots** is non zero, as each new plot window is displayed,older ones may be automatically closed. See section 6.2.

6.9.29 Print: Print values

General Form

```
print [col] [line] expr ...
```

Prints the vector described by the expression **expr**. If the **col** argument is present, print the vectors named side by side. If **line** is given, the vectors are printed horizontally. **col** is the default, unless all the vectors named have a length of one, in which case **line** is the default. The options **width**, **length**, and **nobreak** are effective for this command (see **asciiplot**). If the expression is **all**, all of the vectors available are printed. Thus **print col all > file** prints everything in the file in **SPICE2** format. The scale vector (time, frequency) is always in the first column unless the variable **noprintscale** is true.

6.9.30 Pz: Perform a Pole-Zero Analysis

General Form

```
pz NODE1 NODE2 NODE3 NODE4 CUR POL
pz NODE1 NODE2 NODE3 NODE4 CUR ZER
pz NODE1 NODE2 NODE3 NODE4 CUR PZ
pz NODE1 NODE2 NODE3 NODE4 VOL POL
pz NODE1 NODE2 NODE3 NODE4 VOL ZER
pz NODE1 NODE2 NODE3 NODE4 VOL PZ
```

Examples:

```
pz 1 0 3 0 CUR POL
pz 2 3 5 0 VOL ZER
pz 4 1 4 1 CUR PZ
```

CUR stands for a transfer function of the type (output voltage)/(input current) while **VOL** stands for a transfer function of the type (output voltage)/(input voltage). **POL** stands for pole analysis only, **ZER** for zero analysis only and **PZ** for both. This feature is provided mainly because if there is a non-convergence in finding poles or zeros, then, at least the other can be found. Finally, **NODE1** and **NODE2** are the two input nodes and **NODE3** and **NODE4** are the two output nodes. Thus, there is complete freedom regarding the output and input ports and the type of transfer function.

6.9.31 Quit: Leave WinSpice3

General Form

```
quit
```

Quit WinSpice3.

6.9.32 Rawfile: Send further results directly to a rawfile

General Form

```
rawfile [rawfile][OFF]
```

Send the output of subsequent analyses directly to a file. ‘rawfile off’ restores default operation.

6.9.33 Reset: Reset an analysis

General Form

```
reset
```

Throw out any intermediate data in the circuit (e.g., after a breakpoint or after one or more analyses have been done already), and re-parse the input file. The circuit can then be re-run from it's initial state, overriding the affect of any set or alter commands. In SPICE-3e and earlier versions this was done automatically by the run command.

6.9.34 Reshape: Alter the dimensionality or dimensions of a vector

General Form

```
reshape vector vector ...
```

or

```
reshape vector vector ...  [ dimension, dimension, ...  ]
```

or

```
reshape vector vector ... [ dimension ][ dimension ] ...
```

This command changes the dimensions of a vector or a set of vectors. The final dimension may be left off and it will be filled in automatically. If no dimensions are specified, then the dimensions of the first vector are copied to the other vectors. An error message of the form 'dimensions of x were inconsistent' can be ignored.

6.9.35 Resume: Continue a simulation after a stop

General Form

```
resume
```

Resume a simulation after a stop or interruption (control-C).

6.9.36 Run: Run analysis from the input file

General Form

```
run [rawfile]
```

Run the simulation loaded by a previous ‘source’ command. If there were any of the control lines **.ac**, **.op**, **.tran**, or **.dc**, they are executed.

The output is put in **rawfile** if it was given, in addition to being available interactively.

6.9.37 Rusage: Resource usage

General Form

```
rusage [resource ...]
```

Print resource usage statistics. If any resources are given, just print the usage of that resource. Most resources require that a circuit be loaded.

Currently valid resources are:-

Item	Description
elapsed	The amount of time elapsed since the last rusage elapsed call.
faults	Number of page faults and context switches (BSD only).
space	Data space used.
time	CPU time used so far.
temp	Operating temperature.
tnom	Temperature at which device parameters were measured.
equations	Circuit Equations
time	Total Analysis Time
totiter	Total iterations
accept	Accepted timepoints
rejected	Rejected timepoints
loadtime	Time spent loading the circuit matrix and RHS.
reordertime	Matrix reordering time
lutime	L-U decomposition time
solvetime	Matrix solve time
trantime	Transient analysis time
tranpoints	Transient timepoints
traniter	Transient iterations
trancuriters	Transient iterations for the last time point*
tranlutime	Transient L-U decomposition time
transolvetime	Transient matrix solve time
everything	All of the above.

* listed incorrectly as "Transient iterations per point".

6.9.38 Save: Save a set of output vectors

General Form

```
save [all | vector vector ...]
```

Examples:

```
save i(vin) input output
save @m1[id]
```

Save a set of output vectors, discarding the rest. If a vector has been mentioned in a save command, it appears in the working plot after a run has completed, or in the rawfile if SPICE is run in batch mode. If a vector is traced or plotted (see below) it is also saved.

For backward compatibility, if there are no save commands given, all outputs are saved.

When the keyword ‘all’ appears in the save command, all default values (node voltages and voltage source currents) are saved in addition to any other values listed.

6.9.39 Sens: Run a sensitivity analysis

General Form

```
sens output_variable
sens output_variable ac ( DEC | OCT | LIN ) N Fstart Fstop
```

Perform a Sensitivity analysis. **output_variable** is either a node voltage (ex. "v(1)" or "v(A,out)") or a current through a voltage source (ex. "i(vtest)"). The first form calculates DC sensitivities, the second form calculates AC sensitivities. The output values are in dimensions of change in output per unit change of input (as opposed to percent change in output or per percent change of input).

6.9.40 Set: Set the value of a variable

General Form

```
set [word]
set [word = value] ...
```

Set the value of word to be value, if it is present. You can set any word to be any value, numeric or string. If no value is given then the value is the boolean 'true'.

The value of word may be inserted into a command by writing $word. If a variable is set to a list of values that are enclosed in parentheses (which must be separated from their values by white space), the value of the variable is the list.

The variables used by **WinSpice3** are listed in section 6.2.

6.9.41 Setcirc: Change the current circuit

General Form

```
setcirc [circuit name]
```

The current circuit is the one that is used for the simulation commands below. When a circuit is loaded with the **source** command (see below) it becomes the current circuit.

WinSpice3 maintains a list of circuits which have been loaded into the system. The length of this list is defined by the value of the environment variable **maxcircuits** which, to conserve memory, is set to 1 by default.

6.9.42 Setplot: Switch the current set of vectors

General Form

```
setplot [plotname]
```

Set the current plot to the plot with the given name, or if no name is given, prompt the user with a menu. (Note that the plots are named as they are loaded, with names like tran1 or op2. These names are shown by the **setplot** and **display** commands and are used by **diff**, below.) If the "New plot" item is selected, the current plot becomes one with no vectors defined.

Note that here the word "plot" refers to a group of vectors that are the result of one SPICE run. When more than one file is loaded in, or more than one plot is present in one file, WinSpice3 keeps them separate and only shows you the vectors in the current plot.

6.9.43 Setscale: Set the scale for a plot

General Form

```
setscale [vector]
```

Changes the scale vector for the current plot. If ‘vector’ is not given, this comment displays the scale for the plot.

6.9.44 Settype: Set the type of a vector

General Form

```
settype type vector ...
```

Change the type of the named vectors to **type**. The available type names are as follows:-

type	Units shown on plots
notype	None
time	“S”
frequency	“Hz”
voltage	"V"
current	"A"
onoise-spectrum	"(V or A)^2/Hz"
onoise-integrated	"V or A"
inoise-spectrum	"(V or A)^2/Hz"
inoise-integrated	"V or A"
output-noise	None
input-noise	None
pole	None
Zero	None
s-param	None
impedance	"Ohms"
admittance	"Mhos"
power	"W"
phase	"Degrees" or “Radians”
decibel	"dB"

6.9.45 Shell: Call the command interpreter

General Form

```
shell [command]
```

Call the operating system's command interpreter; execute the specified command or call for interactive use.

6.9.46 Shift: Alter a list variable

General Form

```
shift [varname] [number]
```

If **varname** is the name of a list variable, it is shifted to the left by **number** elements (i.e., the **number** leftmost elements are removed). The default **varname** is **argv**, and the default **number** is 1.

6.9.47 Show: List device state

General Form

```
show
show devs : params
show devs : params ; devs : params
show dev dev dev : param param param , dev dev : param param
show t : param param param, t : param param
```

The **show** command prints out tables summarising the operating condition of selected devices (much like the **SPICE2** operation point summary).

- If **device** is missing, a default set of devices are listed.
- If device is a single letter, devices of that type are listed.
- If device is a subcircuit name (beginning and ending in ":") only devices in that subcircuit are shown (end the name in a double-":" to get devices within sub-subcircuits recursively).

The second and third forms may be combined ("**letter:subcircuit:**") or "**letter:subcircuit::**") to select a specific type of device from a subcircuit. A device's full name may be specified to list only that device. Finally, devices may be selected by model by using the form "**#modelname**" or "**:subcircuit#modelname**" or "**letter:subcircuit#modelname**".

If no parameters are specified, the values for a standard set of parameters are listed. If the list of parameters contains a "**+**", the default set of parameters is listed along with any other specified parameters.

For both devices and parameters, the word "**all**" has the obvious meaning. For example

```
show all : all
```

shows all output parameters in all devices.

Note: there must be spaces separating the ":" that divides the device list from the parameter list.

6.9.48 Showmod: List model parameter values

General Form

```
showmod models [:parameters] , ...
```

The showmod command operates like the show command (above) but prints out model parameter values. The applicable forms for models are a single letter specifying the device type letter, "**letter:subckt:**", "**modelname**", "**:subckt:modelname**", or "**letter:subcircuit:modelname**".

6.9.49 Source: Read a WinSpice3 input file

General Form

```
source file
```

For **WinSpice3**: Read the **WinSpice3** input file.

WinSpice3 commands may be included in the file, and must be enclosed between the lines **.control** and **.endc**. These commands are executed immediately after the circuit is loaded, so a control line of **ac ...** works the same as the corresponding .ac card. The first line in any input file is considered a title line and not parsed but kept as the name of the circuit. The exception to this rule is the file **.spiceinit**. Thus, a **WinSpice3** command script must begin with a blank line and then with a **.control** line.

Also, any line beginning with the characters *# is considered a control line. This makes it possible to embed commands in **WinSpice3** input files that are ignored by **SPICE2**.

Lines beginning with the character * are considered comments and ignored.

6.9.50 Spec: Generate a Fourier transform vector

General Form

```
spec startf stopf stepf vector
```

Calculates a new vector containing the Fourier transform of the input vector. This vector should be the output of a transient analysis.

This command takes note of the following shell variables which can be set using the 'set' command (see section 6.9.40):-

Variable	Type	Description
specwindow	String	Specifies the windowing function. Possible values are:- **none** **hanning** or **cosine** **rectangular** **hamming** **triangle** or **bartlet** **blackman** **gaussian** If this variable is not defined, the **hanning** window is used.
specwindoworder	Number	Specifies the window order for the **gaussian** window only.

Note that the time axis of the input vector should be linearised first by using the 'linearize' command (see section 6.9.23) because **WinSpice3** does not produce a linear time axis for transient analyses. After using the 'spec' command, the spectrum can be displayed by plotting the magnitude of the resultant vector.

For example, after a transient analysis resulting in transient vector v(1), the spectrum can be plotted with the following commands:-

```
linearize
spec 10 100000 5000 v(1)
plot mag(v(1))
```

6.9.51 Status: Display breakpoint and trace information

General Form

```
status
```

Display all of the traces, iplots and breakpoints currently in effect.

6.9.52 Step: Run a fixed number of time points

General Form

```
step [number]
```

Iterate number times, or once, and then stop.

6.9.53 Stop: Set a breakpoint

General Form

```
stop [after n] [when value cond value] ...
```

Set a breakpoint. The argument **after n** means stop after **n** iteration number n, and the argument **when value cond value** means stop when the first value is in the given relation with the second value, the possible relations being

eq	or	=	equal to
ne	or	<>	not equal to
gt	or	>	greater than
lt	or	<	less than
ge	or	>=	greater than or equal to
le	or	<=	less than or equal to

I/O redirection is disabled for the stop command, since the relational operations conflict with it (it doesn't produce any output anyway). The values above may be node names in the running circuit, or real values. If more than one condition is given, e.g. stop after 4 when v(1) > 4 when v(2) < 2, the conjunction of the conditions is implied.

6.9.54 Strcmp: Compare strings

General Form

```
strcmp res var1 var2
```

Example

```
strcmp i $resp new
if $i = 0
    set curplot = new
    goto bottom
end
```

Compare two string variables **var1** and **var2** for equality and set variable **res as follows:-**

0	if they are equal
–1	if var1 < var2
+1	if var1 > var2

6.9.55 Tf: Run a Transfer Function analysis

General Form

```
tf output_node input_source
```

The **tf** command performs a transfer function analysis, returning the transfer function (output/input), output resistance, and input resistance between the given output node and the given input source. The analysis assumes a small-signal DC (slowly varying) input.

6.9.56 Trace: Trace nodes

General Form

```
trace [node ...]
```

For every step of an analysis, the value of the node is printed. Several traces may be active at once. Tracing is not applicable for all analyses. To remove a trace, use the **delete** command.

See the **iplot** command (see section 6.9.21) for a visual form of trace.

6.9.57 Tran: Perform a transient analysis

General Form

```
tran Tstep Tstop [Tstart [Tmax]] [UIC]
```

Perform a transient analysis. See section 5.3.9 of this manual for more details.

6.9.58 Transpose: Swap the elements in a multi-dimensional data set

General Form

```
transpose vector vector ...
```

Example

```
transpose i(vdd) v(drain)
```

This command transposes a multidimensional vector. No analysis in WinSpice3 produces multidimensional vectors, although the DC transfer curve may be run with two varying sources. You must use the "reshape" command to reform the one-dimensional vectors into two dimensional vectors. In addition, the default scale is incorrect for plotting. You must plot versus the vector corresponding to the second source, but you must also refer only to the first segment of this second source vector. For example (circuit to produce the transfer characteristic of a MOS transistor):

```
spice3 > dc vgg 0 5 1 vdd 0 5 1
spice3 > plot i(vdd)
spice3 > reshape all [6,6]
spice3 > transpose i(vdd) v(drain)
spice3 > plot i(vdd) vs v(drain)[0]
```

6.9.59 Tutorial: Display hypertext help

General Form

```
tutorial [subject]
```

Display hierarchical help information from an on-line manual.

6.9.60 Unalias: Retract an alias

General Form

```
unalias [word ...]
```

Removes any aliases present for the words.

6.9.61 Undefine: Retract a definition

General Form

```
undefine function
```

Definitions for the named user-defined functions are deleted.

6.9.62 Unlet: Delete vectors

General Form

```
unlet varname ...
```

Delete one or more vectors.

6.9.63 Unset: Clear a variable

General Form

```
unset [word ...]
```

Clear the value of the specified variable(s) (word).

6.9.64 Version: Print the version of WinSpice

General Form

```
version [version id]
```

Print out the version of WinSpice that is running. If there are arguments, it checks to make sure that the arguments match the current version of WinSpice.

6.9.65 Where: Identify troublesome node or device

General Form

```
where
```

When performing a transient or operating point analysis, the name of the last node or device to cause non-convergence is saved. The **where** command prints out this information so that you can examine the circuit and either correct the problem or make a bug report. You may do this either in the middle of a run or after the simulator has given up on the analysis. For transient simulation, the **iplot** command can be used to monitor the progress of the analysis. When the analysis slows down severely or hangs, interrupt the simulator (with control-C) and issue the **where** command. Note that only one node or device is printed; there may be problems with more than one node.

6.9.66 Write: Write data to a file

General Form

```
write [file [exprs]]
```

Writes out the expressions to file.

First vectors are grouped together by plots, and written out as such (i.e. if the expression list contained three vectors from one plot and two from another, then two plots are written, one with three vectors and one with two). Additionally, if the scale for a vector isn't present, it is automatically written out as well.

The default format is ASCII, but this can be changed with the **set filetype** command. The default filename is **rawspice.raw**, or the argument to the -r flag on the command line, if there was one, and the default expression list is **all**.

If file is given and it has the file extension '.csv', the file will be written as ASCII in comma separated value format.

6.10 Miscellaneous

If there are subcircuits in the input file, **WinSpice3** expands instances of them. A subcircuit is delimited by the cards **.subckt** and **.ends**, or whatever the value of the variables **substart** and **subend** is, respectively. An instance of a subcircuit is created by specifying a device with type 'x' - the device line is written

```
xname node1 node2 ... subcktname
```

where the nodes are the node names that replace the formal parameters on the **.subckt** line. All nodes that are not formal parameters are prepended with the name given to the instance and a ':', as are the names of the devices in the subcircuit. If there are several nested subcircuits, node and device names look like **subckt1:subckt2:...:name**. If the variable **subinvoke** is set, then it is used as the prefix that specifies instances of subcircuits, instead of 'x'.

WinSpice3 occasionally checks to see if it is getting close to running out of space, and warns the user if this is the case.

6.11 Bugs

When defining aliases like

```
alias pdb plot db( '!:1' - '!:2' )
```

you must be careful to quote the argument list substitutions in this manner. If you quote the whole argument it might not work properly.

In a user-defined function, the arguments cannot be part of a name that uses the *plot.vec* syntax. For example:

```
define check(v(1)) cos(tran1.v(1))
```

does not work.

If you type **plot all all**, or otherwise use a wildcard reference for one plot twice in a command, the effect is unpredictable.

The **asciiplot** command doesn't deal with log scales or the **delta** keywords.

WinSpice3 recognises all the notations used in **SPICE2 .plot** cards, and translates **vp(1)** into **ph(v(1))**, and so forth. However, if there are spaces in these names it won't work. Hence **v(1, 2)** and **(-.5, .5)** aren't recognised.

BJTs can have either 3 or 4 nodes, which makes it difficult for the subcircuit expansion routines to decide what to rename. If the fourth parameter has been declared as a model name, then it is assumed that there are 3 nodes, otherwise it is considered a node. To disable this, you can set the variable **nobjthack** which forces BJTs to have 4 nodes (for the purposes of subcircuit expansion, at least).

The **@name[param]** notation might not work with **trace**, **iplot**, etc. yet.

The first line of a command file (except for the **.spiceinit** file) should be a comment, otherwise WinSpice3 may create an empty circuit.

Files specified on the command line are read before **.spiceinit** is read.

7 CONVERGENCE

Both DC and transient solutions are obtained by an iterative process, which is terminated when both of the following conditions hold:

- The non-linear branch currents converge to within a tolerance of 0.1% or 1 picoamp (1.0e-12 Amp), whichever is larger.
- The node voltages converge to within a tolerance of 0.1% or 1 microvolt (1.0e-6 Volt), whichever is larger.

Although the algorithm used in SPICE has been found to be very reliable, in some cases it fails to converge to a solution. When this failure occurs, the program terminates the job.

Failure to converge in DC analysis is usually due to an error in specifying circuit connections, element values, or model parameter values. Regenerative switching circuits or circuits with positive feedback probably will not converge in the DC analysis unless the OFF option is used for some of the devices in the feedback path, or the .NODESET control line is used to force the circuit to converge to the desired state.

7.1 Solving Convergence Problems

The following techniques on solving convergence problems are taken from various sources including:

1. Meares, L.G., Hymowitz C.E. "Simulating With Spice", Intusoft, 1988
2. Muller, K.H. "A SPICE Cookbook", Intusoft, 1990
3. Meares, L.G., Hymowitz C.E. "Spice Applications Handbook", Intusoft, 1990
4. Intusoft Newsletters, various dates from 1986 to present.
5. Quarles, T. L., "Analysis of Performance and Convergence Issues for Circuit Simulation", U.C.Berkeley, ERL Memo M89/42, April 1989.

7.2 What is Convergence? (or Non-Convergence!)

The answer to a non-linear problem, such as those in the SPICE DC and Transient analyses, is found via an iterative solution. For example, WinSpice3 makes an initial guess at the circuit's node voltages and then, using the circuit conductances, finds the mesh currents. The currents are then used to recalculate the node voltages and the cycle begins again. This continues until all of the node voltages settle to within certain tolerance limits, which can be altered using various .OPTIONS parameters such as RELTOL, VNTOL, and ABSTOL.

If the node voltages do not settle down within a certain number of iterations, the DC analysis will issue an error message, such as "No convergence in DC analysis", "PIVTOL Error", "Singular Matrix", or "Gmin/Source Stepping Failed". SPICE will then terminate the run because both the AC and transient analyses require an initial stable operating point in order to start. During the transient analysis, this iterative process is repeated for each individual time step. If the node voltages do not settle down, the time step is reduced and SPICE tries again to determine the node voltages. If the time step is reduced beyond a certain fraction of the total analysis time, the transient analysis will issue an error message ("Time step too small") and the analysis will be halted.

Solutions to the DC analysis may fail to converge because of incorrect initial voltage guesses, model discontinuities, unstable/bistable operation, or unrealistic circuit impedances. Transient analysis failures are usually due to model discontinuities or unrealistic circuit, source, or parasitic modelling. The various solutions to convergence problems fall under one of two types. Some are simply Band-Aids. That is, they

merely try to fix the symptom by adjusting the simulator options. While other solutions actually effect the real cause of the convergence problems.

The following techniques can be used to solve 90-95% of all convergence problems. When a convergence problem is encountered you should start at solution 1 and continue on with the subsequent fixes until convergence is achieved. The order of the solutions is set-up so those lower number fixes can be left in the simulation as additional fixes are added. The order is also set-up so that the initial fixes will be of the most benefit. The user should note that fixes involving simulation options might simply mask the underlying circuit instabilities. Invariably, the user will find that once the circuit is properly modelled, many of the "options" fixes will no longer be required!

7.3 SPICE3 - New Convergence Algorithms

In addition to automatically invoking the traditional source stepping algorithm, SPICE3 contains a new superior algorithm called "Gmin Stepping". This algorithm uses a constant minimal junction conductance to keep the sparse matrix well conditioned and a separate variable conductance to ground at each node as a DC convergence aid. These variable conductances make the solution converge faster, they are then reduced and the solution re-computed. Eventually, the solution is found with a sufficiently small conductance. Finally, the conductance is removed entirely to obtain a final solution. This technique has been found to work very well and SPICE3 uses it by default when convergence problems occur. The suggestion, made in a number of textbooks, of reducing the .OPTIONS GMIN value in order to solve convergence problems is performed automatically by this new algorithm.

7.4 Non-Convergence Error Messages/Indications

DC Analysis	(.OP, small signal bias solution before the AC analysis or Initial transient solution before the Transient analysis) - "No Convergence in DC analysis", "PIVTOL Error". WinSpice3 programs will issue "Gmin/Source Stepping Failed" or "Singular Matrix" messages.
DC SWEEP Analysis (.DC)	"No Convergence in DC analysis at Step = xxx"
Transient (.TRAN)	"Internal timestep too small"

IMPORTANT NOTE: The suggestions listed below are applicable to most SPICE programs, especially if they are Berkeley SPICE compatible.

7.5 Convergence Solutions

7.5.1 DC Convergence Solutions

1. Check the circuit topology and connectivity.

.OPTIONS NODE LIST will provide a nice summary print out of the nodal connections.

Common mistakes:

- Make sure all of the circuit connections are valid. Check for incorrect node numbering or dangling nodes
- Make sure you didn't use the letter O instead of a zero (0)
- Check for syntax mistakes. Make sure you used the correct SPICE units (MEG instead of M for 1E6)
- Check for a DC path to ground from every node

- Ensure that there are two connections at each node
- No loops of inductors or voltage sources
- No series capacitors or current sources
- Have ground (node 0) somewhere in the circuit. Be careful when using floating grounds; a large valued resistor connected from the floating point to ground may be needed
- Check to see that voltage/current generators are at their proper values
- Check to see that dependent source gains are correct
- Check for realistic model parameters; especially if you copied the model into the netlist by hand
- Check to see that all resistors have a value. In WinSpice3 resistors without values are given a default of 1KOhm

2. Increase ITL1 to 300 in the .OPTIONS statement.

Example: .OPTIONS ITL1=300

Increases the number of DC iterations WinSpice3 will go through before giving up. Further increases in ITL1, in all but the most complex circuits, will not usually yield convergence.

3. Set ITL6 =100 in the .OPTIONS statement.

Example: .OPTIONS ITL6=100

Invokes the source stepping algorithm. 100 are the number of steps used. This solution is unnecessary for WinSpice3 users as source stepping is automatically invoked after both the default method and the new Gmin stepping algorithms have been tried. Note for SPICE2 users, this is an undocumented Berkeley SPICE2 option.

4. Add .NODESETs

Example:

```
NODESET V(6)=0
```

Check the node voltage table in the output file. Add .NODESETS statements to nodes that SPICE says have unrealistic or way out voltages. Use a .NODESET of 0V if you do not have a better estimation of the proper DC voltage.

5. Add resistors and use the OFF keyword

Example:

```
D1 1 2 DMOD OFF
RD1 1 2 100MEG
```

Add resistors across diodes to simulate leakage and resistors across MOSFET drain to source connections to simulate realistic channel impedances. Add ohmic resistances (RC, RB, and RE) to transistors. Reduce Gmin an order of magnitude in the .OPTIONS statement. Add the OFF keyword to semiconductors (especially diodes) that may be causing convergence problems. The OFF keyword tells **WinSpice3** to first solve the operating point with the device off. Then, the device is turned on and the previously found operating point is used as a starting condition for the final operating point.

6. Change DC power supplies into PULSE statements.

Example:

From

```
VCC 1 0 15 DC
```

To

```
VCC 1 0 PULSE 0 15
```

This allows the user to selectively turn on certain power supplies just like in real life. This is sometimes known as the "Pseudo-Transient" method. Use a reasonable rise time in the PULSE statement to simulate realistic turn on, for example,

V1 1 0 PULSE 0 5 0 1U would provide a 5 volt supply with a turn on of 1 microsecond. The first value after the 5 voltage (in this case 0) is the turn-on delay that can be used to let the circuit settle down before turning on the power supply.

7. UIC

Example:

```
.TRAN .1N 100N UIC
```

Insert the UIC keyword in the .TRAN statement. UIC means Use Initial Conditions. UIC will cause WinSpice3 to completely by-pass the DC analysis. You should add any applicable .IC and IC= initial conditions statements to assist in the initial stages of the transient analysis. Note: this solution is not viable when you want to perform an AC analysis because the AC analysis must be proceeded by an operating point.

AC Analysis Note: Solutions 5 and 6 should be used as a last resort because they will not produce a valid DC operating point for the circuit (All supplies turned ON). However, if your aim is to get to the transient analysis, then solutions 5 and 6 may help you get there and possibly uncover the hidden problems plaguing the DC analysis along the way.

7.5.2 DC Sweep Convergence Solutions

1. Check circuit topology and connectivity

This is the same as 0 in DC analysis.

Set ITL2=100 in the .OPTIONS statement

Example:

```
.OPTIONS ITL2=100
```

Increases the number of DC iterations WinSpice3 will go through before giving up.

2. Make the steps in the .DC sweep larger or smaller

Example: From

```
.DC VCC 0 1 .1
```

To

```
.DC VCC 0 1 .01
```

Discontinuities in the SPICE models can cause convergence problems. Larger steps can help to bypass the discontinuities while smaller steps can help **WinSpice3** find the intermediate answers that will be used to find non-converging point.

3. Do not use the DC sweep analysis

Example:

From

```
.DC VCC 0 5 .1
VCC 1 0
```

To

```
.TRAN .01 1
VCC 1 0 PULSE 0 5 0 1
```

In many cases it is more effective and efficient to use the transient analysis, by ramping the appropriate voltage and/or current sources, than to use the .DC analysis.

7.5.3 Transient Convergence Solutions

1. Check circuit topology and connectivity

This is similar to 0 in DC analysis.

2. Set RELTOL=.01 in the .OPTIONS statement

Example:

```
.OPTIONS RELTOL=.01
```

This option is actually encouraged for most simulations as reducing the RELTOL will speed the simulation up greatly (10-50%) with only a very minor loss in accuracy. A useful recommendation is to set RELTOL to 0.01 for initial simulations and then reset it when you have the simulation going the way you like it and a more accurate answer is required.

3. Reduce the accuracy of ABSTOL/VNTOL if current/voltage levels allow

Example:

```
.OPTIONS ABSTOL=1N VNTOL=1M
```

ABSTOL/VNTOL can be set to about 8 orders of magnitude below the average voltage/current. Defaults are ABSTOL=1PA and VNTOL=1UV.

4. Set ITL4=100 in the .OPTIONS statement

Example:

```
.OPTIONS ITL4=100
```

Increases the number of transient iterations at each time point that WinSpice3 will go through before giving up.

5. Realistically Model Your Circuit; add parasitics, especially stray/junction capacitance

The idea here is to smooth any strong nonlinearities or discontinuities. Adding capacitance to various nodes and making sure that all semiconductor junctions have capacitance can do this.

Other tips include:

- Use RC snubbers around diodes
- Capacitance for all semiconductor junctions (3PF for diodes, 5PF for BJTs if no specific value is known)
- Add realistic circuit and element parasitics

- Find a subcircuit representation if the model doesn't fit the device behaviour, especially for RF and power devices like RF BJTs and power MOSFETs.

Many vendors cheat and try to "force fit" the SPICE .MODEL statement to represent a device's behaviour. This is a sure sign that the vendor has skimped on quality in favour of quantity. Primitive .MODEL statements CAN NOT be used to model most devices above 200MHz because of the effect of package parasitics. And .MODEL statements CAN NOT be used to model most power devices because of extreme non-linear behaviour. In particular, if your vendor uses a .MODEL statement to model a power MOSFET, throw away the model. It's almost certainly useless for transient analysis.

6. Reduce the rise/fall times of the PULSE sources.

Example:

From

```
VCC 1 0 PULSE 0 1 0 0 0
```

To

```
VCC 1 0 PULSE 0 1 0 1U 1U
```

Again, we are trying to smooth strong nonlinearities. The pulse times should be realistic, not ideal. If no rise or fall times are given, or if 0 is specified, the rise and fall time will be set equal to the TSTEP value in the .TRAN statement.

7. Change the integration to Gear

Example:

```
.OPTIONS METHOD=GEAR
```

Gear should be coupled with a reduction in the RELTOL value. Gear integration, with a reduction in RELTOL, tends to produce answers in the direction of a more stable numerical solution, while trapezoidal integration tends to produce a less stable solution. Gear integration often produces superior results for power circuitry simulations due to the fact that high frequency ringing and long simulation periods are often encountered. **WinSpice3** includes both Trapezoidal and Gear integration.

Gear Integration is a very valuable, especially for Power supply designers.

7.5.4 Special Cases

MOSFETs - Check the connectivity. Connecting two gates together, but to nothing else will give a PIVTOL/Singular matrix error. Check Model Level. SPICE2 does not behave properly when MOSFETs of different Levels are used in the same simulation.

Long Transient Runs, ITL5=0. Don't forget to change the ITL5 .OPTIONS parameter (# of transient iterations) to 0, which means run until completion no matter how many iterations it takes.

7.5.5 WinSpice3 Convergence Helpers

WinSpice3 has several other options available to help convergence.

1. Gminsteps (DC Convergence)

Example: .OPTIONS GMINSTEPS=200

The Gminsteps option adjusts the number of increments that Gmin will be stepped during the DC analysis. Gmin stepping is invoked automatically when there is a convergence problem. Gmin stepping is a new algorithm in WinSpice3 that greatly improves DC convergence.

2. The 'where' function (DC/Transient Convergence)

Example:

```
.control
    where
.endc
```

The new ICL (Interactive Command Language) in WinSpice3 allows the user to ask for specific information about where a convergence problem is taking place. In some cases WinSpice3 does not report the node or device that is failing to converge. The "where" function, is normally added to the control block after the simulation fails. When the simulation is run again, the problem area will be reported.

3. ALTINIT function (Transient Convergence)

Example:

```
.OPTIONS ALTINIT=1
```

Setting ALTINIT to one causes the default algorithm used when the UIC (use initial condition) keyword is issued in the .TRAN to be bypassed in favour of a second more lenient algorithm. Normally the second algorithm is automatically invoked when the default method fails.

4. RSHUNT option

Example:

```
.OPTION RSHUNT=1e9
```

If a circuit fails to converge, or simulates very slowly, try using ‘.option rshunt=1e9’ to the circuit file. This guarantees that all voltage nodes have a path to ground and avoids one cause of non-convergence.

8 BIBLIOGRAPHY

[1] A. Vladimirescu and S. Liu, *The Simulation of MOS Integrated Circuits Using SPICE2*
ERL Memo No. ERL M80/7, Electronics Research Laboratory
University of California, Berkeley, October 1980

[2] T. Sakurai and A. R. Newton, *A Simple MOSFET Model for Circuit Analysis and its application to CMOS gate delay analysis and series-connected MOSFET Structure*
ERL Memo No. ERL M90/19, Electronics Research Laboratory,
University of California, Berkeley, March 1990

[3] B. J. Sheu, D. L. Scharfetter, and P. K. Ko, *SPICE2 Implementation of BSIM*
ERL Memo No. ERL M85/42, Electronics Research Laboratory
University of California, Berkeley, May 1985

[4] J. R. Pierret, *A MOS Parameter Extraction Program for the BSIM Model*
ERL Memo Nos. ERL M84/99 and M84/100, Electronics Research Laboratory
University of California, Berkeley, November 1984

[5] Min-Chie Jeng, *Design and Modeling of Deep-Submicrometer MOSFETSs*
ERL Memo Nos. ERL M90/90, Electronics Research Laboratory
University of California, Berkeley, October 1990

[6] Soyeon Park, *Analysis and SPICE implementation of High Temperature Effects on MOSFET*,
Master's thesis, University of California, Berkeley, December 1986.

[7] Clement Szeto, *Simulator of Temperature Effects in MOSFETs (STEIM)*,
Master's thesis, University of California, Berkeley, May 1988.

[8] J.S. Roychowdhury and D.O. Pederson, *Efficient Transient Simulation of Lossy Interconnect*,
Proc. of the 28th ACM/IEEE Design Automation Conference, June 17-21 1991, San Francisco

[9] A. E. Parker and D. J. Skellern, *An Improved FET Model for Computer Simulators*,
IEEE Trans CAD, vol. 9, no. 5, pp. 551-553, May 1990.

[10] R. Saleh and A. Yang, Editors, *Simulation and Modeling*,
IEEE Circuits and Devices, vol. 8, no. 3, pp. 7-8 and 49, May 1992

[11] H.Statz et al., *GaAs FET Device and Circuit Simulation in SPICE*,
IEEE Transactions on Electron Devices, V34, Number 2, February, 1987 pp160-169.

9 APPENDIX A: EXAMPLE CIRCUITS

9.1 Circuit 1: Differential Pair

The following deck determines the DC operating point of a simple differential pair. In addition, the ac small-signal response is computed over the frequency range 1Hz to 100MEGHz.

```
SIMPLE DIFFERENTIAL PAIR
VCC  7  0    12
VEE  8  0    -12
VIN  1  0    AC 1
RS1  1  2    1K
RS2  6  0    1K
Q1   3  2  4 MOD1
Q2   5  6  4 MOD1
RC1  7  3    10K
RC2  7  5    10K
RE   4  8    10K
.MODEL MOD1 NPN BF=50 VAF=50 IS=1.E-12 RB=100 CJC=.5PF TF=.6NS
.TF V(5) VIN
.AC DEC 10 1 100MEG
.END
```

9.2 Circuit 2: MOSFET Characterisation

The following deck computes the output characteristics of a MOSFET device over the range 0-10V for VDS and 0-5V for VGS.

```
MOS OUTPUT CHARACTERISTICS
.OPTIONS NODE NOPAGE
VDS  3  0
VGS  2  0
M1   1  2  0  0 MOD1 L=4U W=6U AD=10P AS=10P
* VIDS MEASURES ID, WE COULD HAVE USED VDS, BUT ID WOULD BE NEGATIVE
VIDS 3  1
.MODEL MOD1 NMOS VTO=-2 NSUB=1.0E15 UO=550
.DC VDS 0 10 .5 VGS 0 5 1
.END
```

9.3 Circuit 3: RTL Inverter

The following deck determines the DC transfer curve and the transient pulse response of a simple RTL inverter. The input is a pulse from 0 to 5 Volts with delay, rise, and fall times of 2ns and a pulse width of 30ns. The transient interval is 0 to 100ns, with printing to be done every nanosecond.

```
SIMPLE RTL INVERTER
VCC  4  0    5
VIN  1  0    PULSE 0 5 2NS 2NS 2NS 30NS
RB   1  2    10K
Q1   3  2  0 Q1
RC   3  4    1K
.MODEL Q1 NPN BF 20 RB 100 TF .1NS CJC 2PF
.DC VIN 0 5 0.1
.TRAN 1NS 100NS
.END
```

9.4 Circuit 4: Four-Bit Binary Adder

The following deck simulates a four-bit binary adder, using several subcircuits to describe various pieces of the overall circuit.

```
ADDER - 4 BIT ALL-NAND-GATE BINARY ADDER

*** SUBCIRCUIT DEFINITIONS
.SUBCKT NAND 1 2 3 4
*   NODES:  INPUT(2), OUTPUT, VCC
Q1        9  5  1 QMOD
D1CLAMP   0  1    DMOD
Q2        9  5  2 QMOD
D2CLAMP   0  2    DMOD
RB        4  5    4K
R1        4  6    1.6K
Q3        6  9  8 QMOD
R2        8  0    1K
RC        4  7    130
Q4        7  6 10 QMOD
DVBEDROP 10  3    DMOD
Q5        3  8  0 QMOD
.ENDS NAND

.SUBCKT ONEBIT 1 2 3 4 5 6
*   NODES:  INPUT(2), CARRY-IN, OUTPUT, CARRY-OUT, VCC
X1   1  2  7  6   NAND
X2   1  7  8  6   NAND
X3   2  7  9  6   NAND
X4   8  9 10  6   NAND
X5   3 10 11  6   NAND
X6   3 11 12  6   NAND
X7  10 11 13  6   NAND
X8  12 13  4  6   NAND
X9  11  7  5  6   NAND
.ENDS ONEBIT

.SUBCKT TWOBIT 1 2 3 4 5 6 7 8 9
*   NODES:  INPUT - BIT0(2) / BIT1(2), OUTPUT - BIT0 / BIT1,
*           CARRY-IN, CARRY-OUT, VCC
X1   1  2  7  5 10  9   ONEBIT
X2   3  4 10  6  8  9   ONEBIT
.ENDS TWOBIT

.SUBCKT FOURBIT 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
*   NODES:  INPUT - BIT0(2) / BIT1(2) / BIT2(2) / BIT3(2),
*           OUTPUT - BIT0 / BIT1 / BIT2 / BIT3, CARRY-IN, CARRY-OUT,
VCC
X1   1  2  3  4  9 10 13 16 15   TWOBIT
X2   5  6  7  8 11 12 16 14 15   TWOBIT
.ENDS FOURBIT

*** DEFINE NOMINAL CIRCUIT
.MODEL DMOD D
.MODEL QMOD NPN(BF=75 RB=100 CJE=1PF CJC=3PF)
VCC   99  0   DC 5V
VIN1A  1  0   PULSE(0 3 0 10NS 10NS   10NS   50NS)
VIN1B  2  0   PULSE(0 3 0 10NS 10NS   20NS  100NS)
VIN2A  3  0   PULSE(0 3 0 10NS 10NS   40NS  200NS)
VIN2B  4  0   PULSE(0 3 0 10NS 10NS   80NS  400NS)
VIN3A  5  0   PULSE(0 3 0 10NS 10NS  160NS  800NS)
VIN3B  6  0   PULSE(0 3 0 10NS 10NS  320NS 1600NS)
VIN4A  7  0   PULSE(0 3 0 10NS 10NS  640NS 3200NS)
VIN4B  8  0   PULSE(0 3 0 10NS 10NS 1280NS 6400NS)
X1     1  2  3  4  5  6  7  8  9 10 11 12  0 13 99 FOURBIT
```

```
RBIT0  9  0   1K
RBIT1 10  0   1K
RBIT2 11  0   1K
RBIT3 12  0   1K
RCOUT 13  0   1K

*** (FOR THOSE WITH MONEY (AND MEMORY) TO BURN)
.TRAN 1NS 6400NS
.END
```

9.5 Circuit 5: Transmission-Line Inverter

The following deck simulates a transmission-line inverter. Two transmission-line elements are required since two propagation modes are excited. In the case of a coaxial line, the first line (T1) models the inner conductor with respect to the shield, and the second line (T2) models the shield with respect to the outside world.

```
TRANSMISSION-LINE INVERTER
V1   1  0         PULSE(0 1 0 0.1N)
R1   1  2         50
X1   2  0  0  4   TLINE
R2   4  0         50

.SUBCKT TLINE 1 2 3 4
T1   1  2  3  4   Z0=50 TD=1.5NS
T2   2  0  4  0   Z0=100 TD=1NS
.ENDS TLINE

.TRAN 0.1NS 20NS
.END
```

10 APPENDIX B: MODEL AND DEVICE PARAMETERS

The following tables summarise the parameters available for each of the devices and models. There are several tables for each type of device supported by WinSpice.

Input parameters to instances and models are parameters that can occur on an instance or model definition line in the form "keyword=value" where "keyword" is the parameter name as given in the tables. Default input parameters (such as the resistance of a resistor or the capacitance of a capacitor) obviously do not need the keyword specified.

Output parameters are parameters which are available for the output of operating point and debugging information. There are two types of parameters:-

- instance parameters
  These are parameters which are calculated for each instance of a device

- model parameters
  These are parameters which are calculated for a specific model and are shared by all instances of the model.

Instance parameters are specified as "@device[keyword]" and are available for the most recent point computed or, if specified in a ".save" statement, for an entire simulation as a normal output vector. Thus, to monitor the gate-to-source capacitance of a MOSFET, a command

```
save @m1[cgs]
```

given before a transient simulation causes the specified capacitance value to be saved at each timepoint, and a subsequent command such as

```
plot @m1[cgs]
```

produces the desired plot (note that the show command does not use this format).

Model parameters are specified as “@model[keyword]” and are available for the most recent point computed or, if specified in a ".save" statement, for an entire simulation as a normal output vector as for instance parameters.

Another example showing the use of instance and model parameters for a BJT model is below:-

```
.model BC107 NPN(Is=1.527f Xti=3 Eg=1.11 Vaf=106.8 Bf=334.5 Ne=1.642
+ Ise=222f Ikf=.1596 Xtb=1.5 Br=.788 Nc=2 Isc=0 Ikr=0 Re=.6 Rc=0.25
+ Cjc=6.072p Mjc=.3333 Vjc=.75 Fc=.5 Cje=10.67p Mje=.3333 Vje=.75
+ Tr=10n Tf=471.8p Itf=0 Vtf=0 Xtf=0)
. . .
* Declare two BC107 instances in a circuit
Q1 22 24 25 BC107
Q2 42 44 45 BC107
. . .
* Save internal base resistance (model parameter)
* and the base-emitter voltage for q1 and q2.
save @bc107[rb], @q1[vbe], @q2[vbe]
. . .
* Perform an analysis
op
. . .
print @bc107[rb], @q1[vbe], @q2[vbe]
```

Some variables are listed as both input and output, and their output simply returns the previously input value, or the default value after the simulation has been run. Some parameters are input only because the output system can not handle variables of the given type yet, or the need for them as output variables has not been apparent. Many such input variables are available as output variables in a different format, such as the

initial condition vectors that can be retrieved as individual initial condition values. Finally, internally derived values are output only and are provided for debugging and operating point output purposes.

Please note that these tables do not provide the detailed information available about the parameters provided in the section on each device and model, but are provided as a quick reference guide.

10.1 URC: Uniform R.C. line

URC - instance parameters (input-output)	
l	Length of transmission line
n	Number of lumps

URC - instance parameters (output-only)	
pos_node	Positive node of URC
neg_node	Negative node of URC
gnd	Ground node of URC

URC - model parameters (input-only)	
urc	Uniform R.C. line model

URC - model parameters (input-output)	
k	Propagation constant of interest
fmax	Maximum frequency
rperl	Resistance per unit length
cperl	Capacitance per unit length
isperl	Saturation current per length
rsperl	Diode resistance per length

10.2 ASRC: Arbitrary Source

ASRC - instance parameters (input-only)	
i	Current source
v	Voltage source

ASRC - instance parameters (output-only)	
i	Current through source
v	Voltage across source
pos_node	Positive Node
neg_node	Negative Node

10.3 BJT: Bipolar Junction Transistor

BJT - instance parameters (input-only)	
ic	Initial condition vector (vbe, vce)

BJT - instance parameters (input-output)	
off	Device initially off
icvbe	Initial B-E voltage
icvce	Initial C-E voltage
area	Area factor
temp	Instance temperature

BJT - instance parameters (output-only)	
colnode	Number of collector node
basenode	Number of base node
emitnode	Number of emitter node
substnode	Number of substrate node
colprimenode	Internal collector node
baseprimenode	Internal base node
emitprimenode	Internal emitter node
ic	Current at collector node
ib	Current at base node
ie	Emitter current
is	Substrate current
vbe	B-E voltage
vbc	B-C voltage
gm	Small signal transconductance
gpi	Small signal input conductance - pi
gmu	Small signal conductance - mu
gx	Conductance from base to internal base
go	Small signal output conductance
geqcb	d(Ibe)/d(Vbc)
gccs	Internal C-S cap. equiv. cond.
geqbx	Internal C-B-base cap. equiv. cond.
cpi	Internal base to emitter capacitance
cmu	Internal base to collector capacitance
cbx	Base to collector capacitance
ccs	Collector to substrate capacitance
cqbe	Cap. due to charge storage in B-E jct.
cqbc	Cap. due to charge storage in B-C jct.
cqcs	Cap. due to charge storage in C-S jct.
cqbx	Cap. due to charge storage in B-X jct.
cexbc	Total Capacitance in B-X junction
qbe	Charge storage B-E junction
qbc	Charge storage B-C junction
qcs	Charge storage C-S junction
qbx	Charge storage B-X junction
p	Power dissipation

BJT - model parameters (input-output)	
npn	NPN type device
pnp	PNP type device
is	Saturation Current
bf	Ideal forward beta
nf	Forward emission coefficient
vaf	Forward Early voltage
va	Forward Early voltage (same as vaf)
ikf	Forward beta roll-off corner current
ik	Forward beta roll-off corner current (same as ikf)
ise	B-E leakage saturation current
ne	B-E leakage emission coefficient
br	Ideal reverse beta
nr	Reverse emission coefficient
var	Reverse Early voltage
vb	Reverse Early voltage (same as var)
ikr	reverse beta roll-off corner current
isc	B-C leakage saturation current
nc	B-C leakage emission coefficient
rb	Zero bias base resistance
irb	Current for base resistance=(rb+rbm)/2
rbm	Minimum base resistance
re	Emitter resistance
rc	Collector resistance
cje	Zero bias B-E depletion capacitance
vje	B-E built in potential
pe	B-E built in potential (same as vje)
mje	B-E junction grading coefficient
me	B-E junction grading coefficient (same as mje)
tf	Ideal forward transit time
xtf	Coefficient for bias dependence of TF
vtf	Voltage giving VBC dependence of TF
itf	High current dependence of TF
ptf	Excess phase
cjc	Zero bias B-C depletion capacitance
vjc	B-C built in potential
pc	B-C built in potential (same as vjc)
mjc	B-C junction grading coefficient
mc	B-C junction grading coefficient (same as mjc)
xcjc	Fraction of B-C cap to internal base
tr	Ideal reverse transit time
cjs	Zero bias C-S capacitance
ccs	Zero bias C-S capacitance
vjs	Substrate junction built in potential
ps	Substrate junction built in potential (same as vjs)
mjs	Substrate junction grading coefficient
ms	Substrate junction grading coefficient (same as mjs)
xtb	Forward and reverse beta temp. exp.
eg	Energy gap for IS temp. dependency

xti	Temp. exponent for IS
fc	Forward bias junction fit parameter
tnom	Parameter measurement temperature
kf	Flicker Noise Coefficient
af	Flicker Noise Exponent

BJT - model parameters (output-only)	
type	NPN or PNP
invearlyvoltf	Inverse early voltage:forward
invearlyvoltr	Inverse early voltage:reverse
invrollofff	Inverse roll off - forward
invrolloffr	Inverse roll off - reverse
collectorconduct	Collector conductance
emitterconduct	Emitter conductance
transtimevbcfact	Transit time VBC factor
excessphasefactor	Excess phase fact.

10.4 BSIM1: Berkeley Short Channel IGFET Model

BSIM1 - instance parameters (input-only)	
ic	Vector of DS,GS,BS initial voltages

BSIM1 - instance parameters (input-output)	
l	Length
w	Width
ad	Drain area
as	Source area
pd	Drain perimeter
ps	Source perimeter
nrd	Number of squares in drain
nrs	Number of squares in source
off	Device is initially off
icvds	Initial Drain-Source voltage
icvgs	Initial Gate-Source voltage
icvbs	Initial Bulk-Source voltage

BSIM1 - instance parameters (output-only)	
vds	Drain-Source voltage
vgs	Gate-Source voltage
vbs	Bulk-Source voltage
vbd	Bulk-Drain voltage
gm	Transconductance

BSIM1 - model parameters (input-only)	
nmos	Flag to indicate NMOS
pmos	Flag to indicate PMOS

BSIM1 - model parameters (input-output)	
vfb	Flat band voltage
lvfb	Length dependence of vfb
wvfb	Width dependence of vfb
phi	Strong inversion surface potential
lphi	Length dependence of phi
wphi	Width dependence of phi
k1	Bulk effect coefficient 1
lk1	Length dependence of k1
wk1	Width dependence of k1
k2	Bulk effect coefficient 2
lk2	Length dependence of k2
wk2	Width dependence of k2
eta	VDS dependence of threshold voltage
leta	Length dependence of eta
weta	Width dependence of eta
x2e	VBS dependence of eta
lx2e	Length dependence of x2e
wx2e	Width dependence of x2e
x3e	VDS dependence of eta
lx3e	Length dependence of x3e
wx3e	Width dependence of x3e
dl	Channel length reduction in um
dw	Channel width reduction in um
muz	Zero field mobility at VDS=0 VGS=VTH
x2mz	VBS dependence of muz
lx2mz	Length dependence of x2mz
wx2mz	Width dependence of x2mz
mus	Mobility at VDS=VDD VGS=VTH, channel length modulation
lmus	Length dependence of mus
wmus	Width dependence of mus
x2ms	VBS dependence of mus
lx2ms	Length dependence of x2ms
wx2ms	Width dependence of x2ms
x3ms	VDS dependence of mus
lx3ms	Length dependence of x3ms
wx3ms	Width dependence of x3ms
u0	VGS dependence of mobility
lu0	Length dependence of u0
wu0	Width dependence of u0
x2u0	VBS dependence of u0
lx2u0	Length dependence of x2u0
wx2u0	Width dependence of x2u0
u1	VDS dependence of mobility, velocity saturation
lu1	Length dependence of u1
wu1	Width dependence of u1
x2u1	VBS dependence of u1
lx2u1	Length dependence of x2u1
wx2u1	Width dependence of x2u1

x3u1	VDS dependence of u1
lx3u1	Length dependence of x3u1
wx3u1	Width dependence of x3u1
n0	Sub threshold slope
ln0	Length dependence of n0
wn0	Width dependence of n0
nb	VBS dependence of sub threshold slope
lnb	Length dependence of nb
wnb	Width dependence of nb
nd	VDS dependence of sub threshold slope
lnd	Length dependence of nd
wnd	Width dependence of nd
tox	Gate oxide thickness in um
temp	Temperature in degree Celsius
vdd	Supply voltage to specify mus
cgso	Gate source overlap capacitance per unit channel width(m)
cgdo	Gate drain overlap capacitance per unit channel width(m)
cgbo	Gate bulk overlap capacitance per unit channel length(m)
xpart	Flag for channel charge partitioning
rsh	Source drain diffusion sheet resistance in ohm per square
js	Source drain junction saturation current per unit area
pb	Source drain junction built in potential
mj	Source drain bottom junction capacitance grading coefficient
pbsw	Source drain side junction capacitance built in potential
mjsw	Source drain side junction capacitance grading coefficient
cj	Source drain bottom junction capacitance per unit area
cjsw	Source drain side junction capacitance per unit area
wdf	Default width of source drain diffusion in um
dell	Length reduction of source drain diffusion

10.5 BSIM2: Berkeley Short Channel IGFET Model

BSIM2 - instance parameters (input-only)	
ic	Vector of DS,GS,BS initial voltages

BSIM2 - instance parameters (input-output)	
l	Length
w	Width
ad	Drain area
as	Source area
pd	Drain perimeter
ps	Source perimeter
nrd	Number of squares in drain
nrs	Number of squares in source
off	Device is initially off
icvds	Initial D-S voltage
icvgs	Initial G-S voltage
icvbs	Initial B-S voltage

BSIM2 - instance parameters (output-only)	
vds	Drain-Source voltage
vgs	Gate-Source voltage
vbs	Bulk-Source voltage
vbd	Bulk-Drain voltage
gm	Transconductance

BSIM2 - model parameters (input-only)	
nmos	Flag to indicate NMOS
pmos	Flag to indicate PMOS

BSIM2 - model parameters (input-output)	
vfb	Flat band voltage
lvfb	Length dependence of vfb
wvfb	Width dependence of vfb
phi	Strong inversion surface potential
lphi	Length dependence of phi
wphi	Width dependence of phi
k1	Bulk effect coefficient 1
lk1	Length dependence of k1
wk1	Width dependence of k1
k2	Bulk effect coefficient 2
lk2	Length dependence of k2
wk2	Width dependence of k2
eta0	VDS dependence of threshold voltage at VDD=0
leta0	Length dependence of eta0
weta0	Width dependence of eta0
etab	VBS dependence of eta
letab	Length dependence of etab
wetab	Width dependence of etab
dl	Channel length reduction in um
dw	Channel width reduction in um
mu0	Low-field mobility, at VDS=0 VGS=VTH
mu0b	VBS dependence of low-field mobility
lmu0b	Length dependence of mu0b
wmu0b	Width dependence of mu0b
mus0	Mobility at VDS=VDD VGS=VTH
lmus0	Length dependence of mus0
wmus0	Width dependence of mus
musb	VBS dependence of mus
lmusb	Length dependence of musb
wmusb	Width dependence of musb
mu20	VDS dependence of mu in tanh term
lmu20	Length dependence of mu20
wmu20	Width dependence of mu20
mu2b	VBS dependence of mu2
lmu2b	Length dependence of mu2b
wmu2b	Width dependence of mu2b
mu2g	VGS dependence of mu2
lmu2g	Length dependence of mu2g
wmu2g	Width dependence of mu2g
mu30	VDS dependence of mu in linear term
lmu30	Length dependence of mu30
wmu30	Width dependence of mu30
mu3b	VBS dependence of mu3
lmu3b	Length dependence of mu3b
wmu3b	Width dependence of mu3b
mu3g	VGS dependence of mu3
lmu3g	Length dependence of mu3g
wmu3g	Width dependence of mu3g

mu40	VDS dependence of mu in linear term
lmu40	Length dependence of mu40
wmu40	Width dependence of mu40
mu4b	VBS dependence of mu4
lmu4b	Length dependence of mu4b
wmu4b	Width dependence of mu4b
mu4g	VGS dependence of mu4
lmu4g	Length dependence of mu4g
wmu4g	Width dependence of mu4g
ua0	Linear VGS dependence of mobility
lua0	Length dependence of ua0
wua0	Width dependence of ua0
uab	VBS dependence of ua
luab	Length dependence of uab
wuab	Width dependence of uab
ub0	Quadratic VGS dependence of mobility
lub0	Length dependence of ub0
wub0	Width dependence of ub0
ubb	VBS dependence of ub
lubb	Length dependence of ubb
wubb	Width dependence of ubb
u10	VDS dependence of mobility
lu10	Length dependence of u10
wu10	Width dependence of u10
u1b	VBS dependence of u1
lu1b	Length dependence of u1b
wu1b	Width dependence of u1b
u1d	VDS dependence of u1
lu1d	Length dependence of u1d
wu1d	Width dependence of u1d
n0	Sub threshold slope at VDS=0 VBS=0
ln0	Length dependence of n0
wn0	Width dependence of n0
nb	VBS dependence of n
lnb	Length dependence of nb
wnb	Width dependence of nb
nd	VDS dependence of n
lnd	Length dependence of nd
wnd	Width dependence of nd
vof0	Threshold voltage offset AT VDS=0 VBS=0
lvof0	Length dependence of vof0
wvof0	Width dependence of vof0
vofb	VBS dependence of vof
lvofb	Length dependence of vofb
wvofb	Width dependence of vofb
vofd	VDS dependence of vof
lvofd	Length dependence of vofd
wvofd	Width dependence of vofd
ai0	Pre-factor of hot-electron effect

lai0	Length dependence of ai0
wai0	Width dependence of ai0
aib	VBS dependence of ai
laib	Length dependence of aib
waib	Width dependence of aib
bi0	Exponential factor of hot-electron effect
lbi0	Length dependence of bi0
wbi0	Width dependence of bi0
bib	VBS dependence of bi
lbib	Length dependence of bib
wbib	Width dependence of bib
vghigh	Upper bound of the cubic spline function
lvghigh	Length dependence of vghigh
wvghigh	Width dependence of vghigh
vglow	Lower bound of the cubic spline function
lvglow	Length dependence of vglow
wvglow	Width dependence of vglow
tox	Gate oxide thickness in um
temp	Temperature in degree Celsius
vdd	Maximum Vds
vgg	Maximum Vgs
vbb	Maximum Vbs
cgso	Gate source overlap capacitance per unit channel width(m)
cgdo	Gate drain overlap capacitance per unit channel width(m)
cgbo	Gate bulk overlap capacitance per unit channel length(m)
xpart	Flag for channel charge partitioning
rsh	Source drain diffusion sheet resistance in ohm per square
js	Source drain junction saturation current per unit area
pb	Source drain junction built in potential
mj	Source drain bottom junction capacitance grading coefficient
pbsw	Source drain side junction capacitance built in potential
mjsw	Source drain side junction capacitance grading coefficient
cj	Source drain bottom junction capacitance per unit area
cjsw	Source drain side junction capacitance per unit area
wdf	Default width of source drain diffusion in um
dell	Length reduction of source drain diffusion

10.6 Capacitor: Fixed capacitor

Capacitor - instance parameters (input-output)	
capacitance	Device capacitance
ic	Initial capacitor voltage
w	Device width
l	Device length

Capacitor - instance parameters (output-only)	
i	Device current
p	Instantaneous device power

Capacitor - model parameters (input-only)	
c	Capacitor model

Capacitor - model parameters (input-output)	
cj	Bottom Capacitance per area
cjsw	Sidewall capacitance per meter
defw	Default width
tc1	First order temp. coefficient
tc2	Second order temp. coefficient
vc1	First order voltage coefficient
vc2	Second order voltage coefficient
narrow	Width correction factor

10.7 CCCS: Current controlled current source

CCCS - instance parameters (input-output)	
gain	Gain of source
control	Name of controlling source

CCCS - instance parameters (output-only)	
neg_node	Negative node of source
pos_node	Positive node of source
i	CCCS output current
v	CCCS voltage at output
p	CCCS power

10.8 CCVS: Linear current controlled current source

CCVS - instance parameters (input-output)	
gain	Transresistance (gain)
control	Controlling voltage source

CCVS - instance parameters (output-only)	
pos_node	Positive node of source
neg_node	Negative node of source
i	CCVS output current
v	CCVS output voltage
p	CCVS power

10.9 CSwitch: Current controlled ideal switch

CSwitch - instance parameters (input-only)	
on	Initially closed
off	Initially open

CSwitch - instance parameters (input-output)	
control	Name of controlling source

CSwitch - instance parameters (output-only)	
pos_node	Positive node of switch
neg_node	Negative node of switch
i	Switch current
p	Instantaneous power

Cswitch - model parameters (input-output)	
csw	Current controlled switch model
it	Threshold current
ih	Hysteresis current
ron	Closed resistance
roff	Open resistance
ion	Control current to switch on
ioff	Control current to switch off

Cswitch - model parameters (output-only)	
gon	Closed conductance
goff	Open conductance

10.10 Diode: Junction Diode model

Diode - instance parameters (input-output)	
off	Initially off
temp	Instance temperature
ic	Initial device voltage
area	Area factor

Diode – instance parameters (output-only)	
vd	Diode voltage
id	Diode current
c	Diode current
gd	Diode conductance
cd	Diode capacitance
charge	Diode capacitor charge
capcur	Diode capacitor current
p	Diode power

Diode - model parameters (input-only)	
d	Diode model

Diode - model parameters (input-output)	
is	Saturation current
tnom	Parameter measurement temperature
rs	Ohmic resistance
n	Emission Coefficient
tt	Transit Time
cjo	Junction capacitance
cj0	Junction capacitance
vj	Junction potential
m	Grading coefficient
eg	Activation energy
xti	Saturation current temperature exp.
kf	flicker noise coefficient
af	flicker noise exponent
fc	Forward bias junction fit parameter
bv	Reverse breakdown voltage
ibv	Current at reverse breakdown voltage

Diode - model parameters (output-only)	
cond	Ohmic conductance

10.11 Inductor: Inductors

Inductor - instance parameters (input-output)	
inductance	Inductance of inductor
ic	Initial current through inductor

Inductor - instance parameters (output-only)	
flux	Flux through inductor
v	Terminal voltage of inductor
Volt	Terminal voltage of inductor. Same a ‘v’.
i	Current through the inductor
current	Current through the inductor. Same as ‘i’.
p	Instantaneous power dissipated by the inductor

10.12 mutual: Mutual inductors

mutual - instance parameters (input-output)	
k	Mutual inductance
coefficient	(null)
inductor1	First coupled inductor
inductor2	Second coupled inductor

10.13 Isource: Independent current source

Isource - instance parameters (input-only)	
pulse	Pulse description
sine	Sinusoidal source description
sin	Sinusoidal source description
exp	Exponential source description
pwl	Piecewise linear description
sffm	single freq. FM description
ac	AC magnitude,phase vector
c	Current through current source
distof1	f1 input for distortion
distof2	f2 input for distortion

Isource - instance parameters (input-output)	
dc	DC value of source
acmag	AC magnitude
acphase	AC phase

Isource - instance parameters (output-only)	
neg_node	Negative node of source
pos_node	Positive node of source
acreal	AC real part
acimag	AC imaginary part
function	Function of the source
order	Order of the source function
coeffs	Coefficients of the source
v	Voltage across the supply
p	Power supplied by the source

10.14 JFET: Junction Field effect transistor

JFET - instance parameters (input-output)	
off	Device initially off
ic	Initial VDS,VGS vector
area	Area factor
ic-vds	Initial D-S voltage
ic-vgs	Initial G-S voltage
temp	Instance temperature

JFET - instance parameters (output-only)	
drain-node	Number of drain node
gate-node	Number of gate node
source-node	Number of source node
drain-prime-node	Internal drain node
source-prime-node	Internal source node
vgs	Voltage G-S
vgd	Voltage G-D
ig	Current at gate node
id	Current at drain node
is	Source current
igd	Current G-D
gm	Transconductance
gds	Conductance D-S
ggs	Conductance G-S
ggd	Conductance G-D
qgs	Charge storage G-S junction
qgd	Charge storage G-D junction
cqgs	Capacitance due to charge storage G-S junction
cqgd	Capacitance due to charge storage G-D junction
p	Power dissipated by the JFET

JFET - model parameters (input-output)	
njf	N type JFET model
pjf	P type JFET model
vt0	Threshold voltage
vto	Threshold voltage
beta	Transconductance parameter
lambda	Channel length modulation param.
rd	Drain ohmic resistance
rs	Source ohmic resistance
cgs	G-S junction capacitance
cgd	G-D junction cap
pb	Gate junction potential
is	Gate junction saturation current
fc	Forward bias junction fit parm.
b	Doping tail parameter
tnom	Parameter measurement temperature
kf	Flicker Noise Coefficient
af	Flicker Noise Exponent

JFET - model parameters (output-only)	
type	N-type or P-type JFET model
gd	Drain conductance
gs	Source conductance

10.15 LTRA: Lossy transmission line

LTRA - instance parameters (input-only)	
ic	Initial condition vector:v1,i1,v2,i2

LTRA - instance parameters (input-output)	
v1	Initial voltage at end 1
v2	Initial voltage at end 2
i1	Initial current at end 1
i2	Initial current at end 2

LTRA - instance parameters (output-only)	
pos_node1	Positive node of end 1 of t-line
neg_node1	Negative node of end 1 of t.line
pos_node2	Positive node of end 2 of t-line
neg_node2	Negative node of end 2 of t-line

LTRA - model parameters (input-output)	
ltra	LTRA model
r	Resistance per metre
l	Inductance per metre
g	(null)
c	Capacitance per metre
len	Length of line
nocontrol	No timestep control
steplimit	Always limit timestep to 0.8*(delay of line)
nosteplimit	Don't always limit timestep to 0.8*(delay of line)
lininterp	Use linear interpolation
quadinterp	Use quadratic interpolation
mixedinterp	Use linear interpolation if quadratic results look unacceptable
truncnr	Use N-R iterations for step calculation in LTRAtrunc
truncdontcut	Don't limit timestep to keep impulse response calculation errors low
compactrel	Special reltol for straight line checking
compactabs	Special abstol for straight line checking

LTRA - model parameters (output-only)	
rel	Rel. rate of change of deriv. for bkpt
abs	Abs. rate of change of deriv. for bkpt

10.16 MES: GaAs MESFET model

MES - instance parameters (input-output)	
area	Area factor
icvds	Initial D-S voltage
icvgs	Initial G-S voltage

MES - instance parameters (output-only)	
off	Device initially off
dnode	Number of drain node
gnode	Number of gate node
snode	Number of source node
dprimenode	Number of internal drain node
sprimenode	Number of internal source node
vgs	Gate-Source voltage
vgd	Gate-Drain voltage
cg	Gate capacitance
cd	Drain capacitance
cgd	Gate-Drain capacitance
gm	Transconductance
gds	Drain-Source conductance
ggs	Gate-Source conductance
ggd	Gate-Drain conductance
cqgs	Capacitance due to gate-source charge storage
cqgd	Capacitance due to gate-drain charge storage
qgs	Gate-Source charge storage
qgd	Gate-Drain charge storage
is	Source current
p	Power dissipated by the mesfet

MES - model parameters (input-only)	
nmf	N type MESfet model
pmf	P type MESfet model

MES - model parameters (input-output)	
vt0	Pinch-off voltage
vto	Pinch-off voltage
alpha	Saturation voltage parameter
beta	Transconductance parameter
lambda	Channel length modulation parm.
b	Doping tail extending parameter
rd	Drain ohmic resistance
rs	Source ohmic resistance
cgs	G-S junction capacitance
cgd	G-D junction capacitance
pb	Gate junction potential
is	Junction saturation current
fc	Forward bias junction fit parm.
kf	Flicker noise coefficient
af	Flicker noise exponent

MES - model parameters (output-only)	
type	N-type or P-type MESfet model
gd	Drain conductance
gs	Source conductance
depl_cap	Depletion capacitance
vcrit	Critical voltage

10.17 Mos1: Level 1 MOSFET model with Meyer capacitance model

Mos1 - instance parameters (input-only)	
off	Device initially off
ic	Vector of D-S, G-S, B-S voltages

Mos1 - instance parameters (input-output)	
l	Length
w	Width
ad	Drain area
as	Source area
pd	Drain perimeter
ps	Source perimeter
nrd	Drain squares
nrs	Source squares
icvds	Initial Drain-Source voltage
icvgs	Initial Gate-Source voltage
icvbs	Initial Bulk-Source voltage
temp	Instance temperature

Mos1 - instance parameters (output-only)	
id	Drain current
is	Source current
ig	Gate current
ib	Bulk current
ibd	B-D junction current
ibs	B-S junction current
vgs	Gate-Source voltage
vds	Drain-Source voltage
vbs	Bulk-Source voltage
vbd	Bulk-Drain voltage
dnode	Number of the drain node
gnode	Number of the gate node
snode	Number of the source node
bnode	Number of the node
dnodeprime	Number of int. drain node
snodeprime	Number of int. source node
von	Turn-on voltage
vdsat	Saturation drain voltage
sourcevcrit	Critical source voltage
drainvcrit	Critical drain voltage
rs	Source resistance
sourceconductance	Conductance of source
rd	Drain conductance
drainconductance	Conductance of drain
gm	Transconductance
gds	Drain-Source conductance
gmb	Bulk-Source transconductance
gmbs	Bulk-Source transconductance
gbd	Bulk-Drain conductance
gbs	Bulk-Source conductance
cbd	Bulk-Drain capacitance
cbs	Bulk-Source capacitance
cgs	Gate-Source capacitance
cgd	Gate-Drain capacitance
cgb	Gate-Bulk capacitance
cqgs	Capacitance due to gate-source charge storage
cqgd	Capacitance due to gate-drain charge storage
cqgb	Capacitance due to gate-bulk charge storage
cqbd	Capacitance due to bulk-drain charge storage
cqbs	Capacitance due to bulk-source charge storage
cbd0	Zero-Bias B-D junction capacitance
cbdsw0	
cbs0	Zero-Bias B-S junction capacitance
cbssw0	
qgs	Gate-Source charge storage
qgd	Gate-Drain charge storage
qgb	Gate-Bulk charge storage
qbd	Bulk-Drain charge storage

qbs	Bulk-Source charge storage
p	Instantaneous power

Mos1 - model parameters (input-only)	
nmos	N type MOSFET model
pmos	P type MOSFET model

Mos1 - model parameters (input-output)	
vto	Threshold voltage
vt0	Threshold voltage
kp	Transconductance parameter
gamma	Bulk threshold parameter
phi	Surface potential
lambda	Channel length modulation
rd	Drain ohmic resistance
rs	Source ohmic resistance
cbd	B-D junction capacitance
cbs	B-S junction capacitance
is	Bulk junction sat. current
pb	Bulk junction potential
cgso	Gate-source overlap cap.
cgdo	Gate-drain overlap cap.
cgbo	Gate-bulk overlap cap.
rsh	Sheet resistance
cj	Bottom junction cap per area
mj	Bottom grading coefficient
cjsw	Side junction cap per area
mjsw	Side grading coefficient
js	Bulk jct. sat. current density
tox	Oxide thickness
ld	Lateral diffusion
u0	Surface mobility
uo	Surface mobility
fc	Forward bias jct. fit parm.
nsub	Substrate doping
tpg	Gate type
nss	Surface state density
tnom	Parameter measurement temperature
kf	Flicker noise coefficient
af	Flicker noise exponent

Mos1 - model parameters (output-only)	
type	N-channel or P-channel MOS

10.18 Mos2: Level 2 MOSFET model with Meyer capacitance model

Mos2 - instance parameters (input-only)	
off	Device initially off
ic	Vector of D-S, G-S, B-S voltages

Mos2 - instance parameters (input-output)	
l	Length
w	Width
ad	Drain area
as	Source area
pd	Drain perimeter
ps	Source perimeter
nrd	Drain squares
nrs	Source squares
icvds	Initial D-S voltage
icvgs	Initial G-S voltage
icvbs	Initial B-S voltage
temp	Instance operating temperature

Mos2 - instance parameters (output-only)	
id	Drain current
cd	Drain current
ibd	B-D junction current
ibs	B-S junction current
is	Source current
ig	Gate current
ib	Bulk current
vgs	Gate-Source voltage
vds	Drain-Source voltage
vbs	Bulk-Source voltage
vbd	Bulk-Drain voltage
dnode	Number of drain node
gnode	Number of gate node
snode	Number of source node
bnode	Number of bulk node
dnodeprime	Number of internal drain node
snodeprime	Number of internal source node
von	Turn-on voltage
vdsat	Saturation drain voltage
sourcevcrit	Critical source voltage
drainvcrit	Critical drain voltage
rs	Source resistance
sourceconductance	Source conductance
rd	Drain resistance
drainconductance	Drain conductance
gm	Transconductance
gds	Drain-Source conductance
gmb	Bulk-Source transconductance
gmbs	Bulk-Source transconductance
gbd	Bulk-Drain conductance
gbs	Bulk-Source conductance
cbd	Bulk-Drain capacitance
cbs	Bulk-Source capacitance
cgs	Gate-Source capacitance
cgd	Gate-Drain capacitance
cgb	Gate-Bulk capacitance
cbd0	Zero-Bias B-D junction capacitance
cbdsw0	
cbs0	Zero-Bias B-S junction capacitance
cbssw0	
cqgs	Capacitance due to gate-source charge storage
cqgd	Capacitance due to gate-drain charge storage
cqgb	Capacitance due to gate-bulk charge storage
cqbd	Capacitance due to bulk-drain charge storage
cqbs	Capacitance due to bulk-source charge storage
qgs	Gate-Source charge storage
qgd	Gate-Drain charge storage
qgb	Gate-Bulk charge storage

qbd	Bulk-Drain charge storage
qbs	Bulk-Source charge storage
p	Instantaneous power

Mos2 - model parameters (input-only)	
nmos	N type MOSFET model
pmos	P type MOSFET model

Mos2 - model parameters (input-output)	
vto	Threshold voltage
vt0	Threshold voltage
kp	Transconductance parameter
gamma	Bulk threshold parameter
phi	Surface potential
lambda	Channel length modulation
rd	Drain ohmic resistance
rs	Source ohmic resistance
cbd	B-D junction capacitance
cbs	B-S junction capacitance
is	Bulk junction sat. current
pb	Bulk junction potential
cgso	Gate-source overlap cap.
cgdo	Gate-drain overlap cap.
cgbo	Gate-bulk overlap cap.
rsh	Sheet resistance
cj	Bottom junction cap per area
mj	Bottom grading coefficient
cjsw	Side junction cap per area
mjsw	Side grading coefficient
js	Bulk jct. sat. current density
tox	Oxide thickness
ld	Lateral diffusion
u0	Surface mobility
uo	Surface mobility
fc	Forward bias jct. fit parm.
nsub	Substrate doping
tpg	Gate type
nss	Surface state density
delta	Width effect on threshold
uexp	Crit. field exp for mob. deg.
ucrit	Crit. field for mob. degradation
vmax	Maximum carrier drift velocity
xj	Junction depth
neff	Total channel charge coeff.
nfs	Fast surface state density
tnom	Parameter measurement temperature
kf	Flicker noise coefficient
af	Flicker noise exponent

Mos2 - model parameters (output-only)	
type	N-channel or P-channel MOS

10.19 Mos3: Level 3 MOSFET model with Meyer capacitance model

Mos3 - instance parameters (input-only)	
off	Device initially off

Mos3 - instance parameters (input-output)	
l	Length
w	Width
ad	Drain area
as	Source area
pd	Drain perimeter
ps	Source perimeter
nrd	Drain squares
nrs	Source squares
icvds	Initial D-S voltage
icvgs	Initial G-S voltage
icvbs	Initial B-S voltage
ic	Vector of D-S, G-S, B-S voltages
temp	Instance operating temperature

Mos3 - instance parameters (output-only)	
id	Drain current
cd	Drain current
ibd	B-D junction current
ibs	B-S junction current
is	Source current
ig	Gate current
ib	Bulk current
vgs	Gate-Source voltage
vds	Drain-Source voltage
vbs	Bulk-Source voltage
vbd	Bulk-Drain voltage
dnode	Number of drain node
gnode	Number of gate node
snode	Number of source node
bnode	Number of bulk node
dnodeprime	Number of internal drain node
snodeprime	Number of internal source node
von	Turn-on voltage
vdsat	Saturation drain voltage
sourcevcrit	Critical source voltage
drainvcrit	Critical drain voltage
rs	Source resistance
sourceconductance	Source conductance
rd	Drain resistance
drainconductance	Drain conductance
gm	Transconductance
gds	Drain-Source conductance
gmb	Bulk-Source transconductance
gmbs	Bulk-Source transconductance
gbd	Bulk-Drain conductance
gbs	Bulk-Source conductance
cbd	Bulk-Drain capacitance
cbs	Bulk-Source capacitance
cgs	Gate-Source capacitance
cgd	Gate-Drain capacitance
cgb	Gate-Bulk capacitance
cqgs	Capacitance due to gate-source charge storage
cqgd	Capacitance due to gate-drain charge storage
cqgb	Capacitance due to gate-bulk charge storage
cqbd	Capacitance due to bulk-drain charge storage
cqbs	Capacitance due to bulk-source charge storage
cbd0	Zero-Bias B-D junction capacitance
cbdsw0	Zero-Bias B-D sidewall capacitance
cbs0	Zero-Bias B-S junction capacitance
cbssw0	Zero-Bias B-S sidewall capacitance
qbs	Bulk-Source charge storage
qgs	Gate-Source charge storage
qgd	Gate-Drain charge storage

qgb	Gate-Bulk charge storage
qbd	Bulk-Drain charge storage
p	Instantaneous power

Mos3 - model parameters (input-only)	
nmos	N type MOSFET model
pmos	P type MOSFET model

Mos3 - model parameters (input-output)	
vto	Threshold voltage
vt0	Threshold voltage
kp	Transconductance parameter
gamma	Bulk threshold parameter
phi	Surface potential
rd	Drain ohmic resistance
rs	Source ohmic resistance
cbd	B-D junction capacitance
cbs	B-S junction capacitance
is	Bulk junction sat. current
pb	Bulk junction potential
cgso	Gate-source overlap cap.
cgdo	Gate-drain overlap cap.
cgbo	Gate-bulk overlap cap.
rsh	Sheet resistance
cj	Bottom junction cap per area
mj	Bottom grading coefficient
cjsw	Side junction cap per area
mjsw	Side grading coefficient
js	Bulk jct. sat. current density
tox	Oxide thickness
ld	Lateral diffusion
u0	Surface mobility
uo	Surface mobility
fc	Forward bias jct. fit parm.
nsub	Substrate doping
tpg	Gate type
nss	Surface state density
vmax	Maximum carrier drift velocity
xj	Junction depth
nfs	Fast surface state density
xd	Depletion layer width
alpha	Alpha
eta	Vds dependence of threshold voltage
delta	Width effect on threshold
input_delta	(null)
theta	Vgs dependence on mobility
kappa	Kappa
tnom	Parameter measurement temperature
kf	Flicker noise coefficient
af	Flicker noise exponent

Mos3 - model parameters (output-only)	
type	N-channel or P-channel MOS

10.20 Mos6: Level 6 MOSFET model with Meyer capacitance model

Mos6 - instance parameters (input-only)	
off	Device initially off
ic	Vector of D-S, G-S, B-S voltages

Mos6 - instance parameters (input-output)	
l	Length
w	Width
ad	Drain area
as	Source area
pd	Drain perimeter
ps	Source perimeter
nrd	Drain squares
nrs	Source squares
icvds	Initial D-S voltage
icvgs	Initial G-S voltage
icvbs	Initial B-S voltage
temp	Instance temperature

Mos6 - instance parameters (output-only)	
id	Drain current
cd	Drain current
is	Source current
ig	Gate current
ib	Bulk current
ibs	B-S junction capacitance
ibd	B-D junction capacitance
vgs	Gate-Source voltage
vds	Drain-Source voltage
vbs	Bulk-Source voltage
vbd	Bulk-Drain voltage
dnode	Number of the drain node
gnode	Number of the gate node
snode	Number of the source node
bnode	Number of the node
dnodeprime	Number of int. drain node
snodeprime	Number of int. source node
rs	Source resistance
sourceconductance	Source conductance
rd	Drain resistance
drainconductance	Drain conductance
von	Turn-on voltage
vdsat	Saturation drain voltage
sourcevcrit	Critical source voltage
drainvcrit	Critical drain voltage
gmbs	Bulk-Source transconductance
gm	Transconductance
gds	Drain-Source conductance
gbd	Bulk-Drain conductance
gbs	Bulk-Source conductance
cgs	Gate-Source capacitance
cgd	Gate-Drain capacitance
cgb	Gate-Bulk capacitance
cbd	Bulk-Drain capacitance
cbs	Bulk-Source capacitance
cbd0	Zero-Bias B-D junction capacitance
cbdsw0	
cbs0	Zero-Bias B-S junction capacitance
cbssw0	
cqgs	Capacitance due to gate-source charge storage
cqgd	Capacitance due to gate-drain charge storage
cqgb	Capacitance due to gate-bulk charge storage
cqbd	Capacitance due to bulk-drain charge storage
cqbs	Capacitance due to bulk-source charge storage
qgs	Gate-Source charge storage
qgd	Gate-Drain charge storage
qgb	Gate-Bulk charge storage
qbd	Bulk-Drain charge storage

qbs	Bulk-Source charge storage
p	Instantaneous power

Mos6 - model parameters (input-only)	
nmos	N type MOSFET model
pmos	P type MOSFET model

Mos6 - model parameters (input-output)	
vto	Threshold voltage
vt0	Threshold voltage
kv	Saturation voltage factor
nv	Saturation voltage coeff.
kc	Saturation current factor
nc	Saturation current coeff.
nvth	Threshold voltage coeff.
ps	Sat. current modification par.
gamma	Bulk threshold parameter
gamma1	Bulk threshold parameter 1
sigma	Static feedback effect par.
phi	Surface potential
lambda	Channel length modulation param.
lambda0	Channel length modulation param. 0
lambda1	Channel length modulation param. 1
rd	Drain ohmic resistance
rs	Source ohmic resistance
cbd	B-D junction capacitance
cbs	B-S junction capacitance
is	Bulk junction sat. current
pb	Bulk junction potential
cgso	Gate-source overlap cap.
cgdo	Gate-drain overlap cap.
cgbo	Gate-bulk overlap cap.
rsh	Sheet resistance
cj	Bottom junction cap per area
mj	Bottom grading coefficient
cjsw	Side junction cap per area
mjsw	Side grading coefficient
js	Bulk jct. sat. current density
ld	Lateral diffusion
tox	Oxide thickness
u0	Surface mobility
uo	Surface mobility
fc	Forward bias jct. fit parm.
tpg	Gate type
nsub	Substrate doping
nss	Surface state density
tnom	Parameter measurement temperature

Mos6 - model parameters (output-only)	
type	N-channel or P-channel MOS

10.21 Resistor: Simple resistor

Resistor - instance parameters (input-output)	
resistance	Resistance
temp	Instance operating temperature
l	Length
w	Width

Resistor - instance parameters (output-only)	
i	Current
p	Power

Resistor - model parameters (input-only)	
r	Device is a resistor model

Resistor - model parameters (input-output)	
rsh	Sheet resistance
narrow	Narrowing of resistor
tc1	First order temp. coefficient
tc2	Second order temp. coefficient
defw	Default device width
tnom	Parameter measurement temperature

10.22 Switch: Ideal voltage controlled switch

Switch - instance parameters (input-only)	
on	Switch initially closed
off	Switch initially open

Switch - instance parameters (input-output)	
pos_node	Positive node of switch
neg_node	Negative node of switch

Switch - instance parameters (output-only)	
cont_p_node	Positive contr. node of switch
cont_n_node	Positive contr. node of switch
i	Switch current
p	Switch power

Switch - model parameters (input-output)	
sw	Switch model
vt	Threshold voltage
vh	Hysteresis voltage
ron	Resistance when closed
roff	Resistance when open
von	Control voltage to switch on
voff	Control voltage to switch off

Switch - model parameters (output-only)	
gon	Conductance when closed
goff	Conductance when open

10.23 Tranline: Lossless transmission line

Tranline - instance parameters (input-only)	
ic	Initial condition vector:v1,i1,v2,i2

Tranline - instance parameters (input-output)	
z0	Characteristic impedance
zo	Characteristic impedance
f	Frequency
td	Transmission delay
nl	Normalized length at frequency given
v1	Initial voltage at end 1
v2	Initial voltage at end 2
i1	Initial current at end 1
i2	Initial current at end 2

Tranline - instance parameters (output-only)	
rel	Rel. rate of change of deriv. for bkpt
abs	Abs. rate of change of deriv. for bkpt
pos_node1	Positive node of end 1 of t. line
neg_node1	Negative node of end 1 of t. line
pos_node2	Positive node of end 2 of t. line
neg_node2	Negative node of end 2 of t. line
delays	Delayed values of excitation

10.24 VCCS: Voltage controlled current source

VCCS - instance parameters (input-output)	
gain	Transconductance of source (gain)

VCCS - instance parameters (output-only)	
pos_node	Positive node of source
neg_node	Negative node of source
cont_p_node	Positive node of contr. source
cont_n_node	Negative node of contr. source
i	Output current
v	Voltage across output
p	Power

10.25 VCVS: Voltage controlled voltage source

VCVS - instance parameters (input-only)	
ic	Initial condition of controlling source

VCVS - instance parameters (input-output)	
gain	Voltage gain

VCVS - instance parameters (output-only)	
pos_node	Positive node of source
neg_node	Negative node of source
cont_p_node	Positive node of contr. source
cont_n_node	Negative node of contr. source
i	Output current
v	Output voltage
p	Power

10.26 Vsource: Independent voltage source

Vsource - instance parameters (input-only)	
pulse	Pulse description (vector)
sine	Sinusoidal source description (vector)
sin	Sinusoidal source description (vector)
exp	Exponential source description (vector)
pwl	Piecewise linear description (vector)
sffm	Single freq. FM description
ac	AC magnitude, phase vector
distof1	f1 input for distortion
distof2	f2 input for distortion

Vsource - instance parameters (input-output)	
dc	D.C. source value
acmag	A.C. Magnitude
acphase	A.C. Phase

Vsource - instance parameters (output-only)	
pos_node	Positive node of source
neg_node	Negative node of source
function	Function of the source
order	Order of the source function
coeffs	Coefficients for the function
acreal	AC real part
acimag	AC imaginary part
i	Voltage source current
p	Instantaneous power