REFERENCE MANUAL

# Multisim™ SPICE

This manual documents SPICE-based circuit syntax that is supported by Multisim's Netlist Parser. The sections describe general purpose syntax used for such operations as device declaration, and device-specific syntax used to parameterize primitive devices such as MOSFETs.

These sections are intended to serve as a reference guide. For more information about SPICE, you may wish to consult The SPICE Book, Andrei Vladimirescu, John Wiley & Sons Inc., 1994, ISBN-13: 978-0-471-60926-1, or Semiconductor Device Modeling with SPICE, Second Edition, Giuseppe Massobrio and Paolo Antognetti, McGraw-Hill, 1993, ISBN-13: 978-0070024694.

**Note** Since the simulation engine is case insensitive, the use of both upper and lower case characters in the following sections is done strictly for clarity.

**Related Information**

*Documentation Conventions*
*General Purpose Syntax*
*Mathematical Expressions*
*Analog Devices Library*
*XSPICE Syntax Reference*
*Unusual Forms of Device Syntax*
*Compatibility Modes*

## Contents



NATIONAL INSTRUMENTS™

# Documentation Conventions

This specification uses the following conventions:

- Text enclosed in <> is optional.
- `Red text` is a variable, a function, or an instance name.
- `Orange text` is a mathematical expression or numeric constant.
- `Blue text` is a node identifier.
- `Green text` is a model identifier, device name, or subckt name.
- `Brown text` is an XSPICE terminal type specifier.

# General Purpose Syntax

The following sections describe the overall syntax and building blocks of SPICE models.

**Related Information**

*Primitive Device Declarations*
*SPICE Subcircuits*
*Netlist Parameters*
*Number Format*
*Comments and Line Continuation*

## Primitive Device Declarations

A primitive device is the lowest level model that can be used in a circuit and is a building block for macromodels and entire circuits. Multisim supports many such devices.

This section looks at how a primitive device is declared and used in a circuit.

Primitive devices are comprised of either just an instance declaration or an instance declaration with an associated model definition.

**Instance Declaration**

The instance declaration places a primitive device between circuit nodes, specifies device parameters, and links the instance to a model definition (where needed). The instance declaration has the following general format:

PREFIX*_anyname* node1 <node2 <…>> my_ModelNAME *Instance_line_parameters*

| Identifier | Description |
|---|---|
| PREFIX | Device specific character. |
| _anyname | Arbitrary instance name suffix. |
| `nodeN`<br>where `N=1,2,3…` | The name of the Nth node that the device is connected to. Node names may contain any characters except for white space and the following: `\ {} () [] : # " ' ; , % < > ` & = *` |

| Identifier | Description |
|---|---|
| my_ModelNAME | Arbitrary name that links the instance declaration to a model definition. It is optional for some devices is and mandatory for others. |
| Instance_line_ parameters | A list of device specific instance parameters, some of which are mandatory and some optional. |

### Model Definition

Required for some primitive devices and optional for others, the model line allows users to specify additional parameters for a device. The model line has the following format:

.model my_ModelNAME <AKO:akomodelname> devicename (<*Model_Parameters*>)

| Identifier | Description |
|---|---|
| my_ModelNAME | Arbitrary model identifier that links the model definition to one or more instance declarations. |
| akomodelname | Model identifier from which specified model parameters will be inherited. |
| devicename | Device-specific identifier. |
| Model_Parameters | List of device-specific model parameter assignments. Note that every model parameter of every device has a default value. Assignments made in the *Model_Parameters* block overwrite these defaults. |

### Examples

*Resistor device without its optional model line
R88 1 0 10k tc1=0.1 tc2=5

*Resistor device with its optional model line
R15 in8 out myResmodel 10k
.model myResmodel res(rmult=5)

*A BJT with area factor of 8
*and BJT with an area factor 4
Q54 99 b 0 BJTer area=8
Q55 e b 0 BJTer area=4
.model BJTer npn(is=1e-12 bf=140 rc=3 tf=3n)

*MOSFET M1 will inherit
*kp=1e-3 phi=0.76 mj=0.44 ld=0.1u
M1 1 4 9 9 myMOS1
.model myMOS1 AKO:myMOS2 nmos(vto=1.4)
.model myMOS2 nmos(kp=1e-3 phi=0.76 mj=0.44 ld=0.1u)

### Related Information

*Analog Devices Library*

# SPICE Subcircuits

A SPICE subcircuit wraps around a block of circuit text and allows external connections to this circuitry only through the subcircuit's port. The benefit of this is that the internal circuitry is isolated from external circuitry, thus internal devices and node names with the same names as those external to the subcircuit are neither conflicting nor shorted. In addition, subcircuits can accept circuit parameters which can be used to assign values to internal devices or nested subcircuits.

A subcircuit is an extremely useful concept, forming the basis of any modular or hierarchical design.

### Subcircuit Definition

`.SUBCKT` mysubcktname node1 <node2 <...>> <`OPTIONAL:` optionalnode1=defaultnode1 <optionalnode2=defaultnode2 <...>>> <`PARAMS:` `param1`=default1 <`param2`=default2 <...>>> <*Subcircuit Contents*> `.ENDS`

| Identifier | Description |
|---|---|
| mysubcktname | Arbitrary subcircuit identifier that links the subcircuit to an instance declaration. |
| node*N*<br>where *N=1,2,3...* | The name of the Nth node for the subcircuit port. These are the nodes that Multisim's component symbol pins would be mapped to. |
| optionalnode*N*<br>where *N=1,2,3...* | The name of the Nth optional node for the subcircuit port. These nodes are optionally included on the instance line and if they are not provided then they will be connected to optionalnode*N*. Optional nodes are not available on Multisim components and thus they are not useful on the top level models of a component. |
| defaultnode*N*<br>where *N=1,2,3...* | The default node to connect to for optionalnodeN if that node is not provided on the instance line. This node should be a valid node from outside this subcircuit declaration. |
| param*N*<br>where *N=1,2,3...* | The name of the Nth input circuit parameter for the subcircuit. |
| param*N*<br>where *N=1,2,3...* | A default value for the Nth input parameter if `PARAMN` is not specified on the subcircuit instance declaration. |

A subcircuit definition is only useful if it is referenced by one or more instance declarations inside a circuit. This can be done as a top level model of a component on a schematic or by declaring an instance of the subcircuit within a model.

### Instance Declaration

Xanyname node1 <node2 <…>> mysubcktName <PARAMS: PARAM1=expression1 <PARAM2=expression2 <…>>>

| Identifier | Description |
|---|---|
| node*N*<br>where *N=1,2,3*... | The Nth node for the subckt. The number of nodes must match the number of nodes in the subcircuit definition. |
| mysubcktName | An arbitrary subcircuit identifier linking the instance declaration to a subcircuit. |
| param*N*<br>where *N=1,2,3*... | The name of the input parameter for the subckt. The nodes do not have to be in the same order as the subcircuit definition and do not all need to be present. If any particular PARAM*N* is omitted then the subcircuit will use default*N*. |
| expression*N*<br>where *N=1,2,3*... | The expression for the Nth input parameter. See the expression section for details regarding expressions. |

### Additional Notes

- This SPICE-based subcircuit should not be confused with a Multisim schematic capture subcircuit which is used to created hierarchy with schematic symbols.
- If the “PARAMS:” keyword is omitted within the circuit parameters portion of the declaration, the entire circuit parameters portion must be enclosed by round “( )” or curly “{ }” parenthesis.
- Node “0” is a global node—regardless of circuit or subcircuit hierarchy, all nodes with the name “0” are connected together.

### Examples

*A resistive voltage divider circuit that uses a resistor subcircuit model. The upper resistor is 10k and the lower is 47k
V1 in 0 10
X1 in mid res_block params: res_val=10k
X2 mid 0 res_block
.subckt res_block 1 2 params: res_val=47k
R1 1 2 {res_val}
.ends

*expression usage
.param gain=100
V1 5 0 3.3
X1 5 8 AMP PARAMS: ampgain={limit(gain, 200, 80)}
.subckt AMP in out PARAMS: ampgain=90
E1 out 0 in 0 {ampgain}
.ends

# Netlist Parameters

Netlist parameters allow flexibility in assigning device/model parameters. The general format for defining a netlist parameter is as follows:

```
.param my_parameter_name = expression
```

| Identifier | Description |
|---|---|
| my_parameter_name | Arbitrary parameter name. It may contain numbers, letters, and underscores but no other symbols. In addition, it must not start with a number. |
| expression | Arbitrary expression operating on numerical constants or netlist parameters. Circuit variables (node voltage and device current) are not permitted. |

### Additional Notes

- Parameters are constants, and thus may not contain any circuit variables such as node voltages or branch currents.
- Parameters take precedence over pre-defined constants. If you define a variable called pi it will be used in place of the normal built-in constant.
- The `.circuitparams` command allows you to expose parameters to certain analyses, such as parameter sweep, which can control and override the parameter value. Simply specify the `.circuitparams` keyword and follow it with a list of names of parameters. Alternatively, you can work with circuit parameters, which are defined and managed at schematic capture level, and not the SPICE netlist directly.

### Examples

```
.param a = 6
.param n = {0.5}
.param maxV = a+10

V1 in 0 {maxV}
D1 in 3 mydiode
.model mydiode d(n={n+0.01})
```

### Related Information

*Mathematical Expressions*

### Parameter Scope

You can use SPICE subckts and parameter namespaces to control the scope of parameters.

### SPICE subckts

Parameters exist at the level of the circuit in which they are defined and in all child subckts. So if a parameter is defined at the top-level (outside of all SPICE subckts) it can also be used in all SPICE subckts. However, if it is defined in a SPICE subckt, then that parameter will be meaningless in the circuit levels above. In case of a naming conflict, the local parameters take

precedence. In the example below, the resistor R1's value is 3.14, R2's value is 2, R3's value is 3, and R4's value is 4.

```
R1 51 0 {pi}
X1 1 2 mysub

.subckt mysub node1 node2
.param pi = 2
R2 node1 node2 { pi }
X2 node3 node2 mysub2
.subckt mysub2 node1 node2
.param pi = 3
R3 node1 node2 { pi }
R4 node1 node2 {varb}
.ends
.param varb = 4
.ends
```

#### Parameter namespaces

Parameters can be enclosed in `.param_namespace_begin` and `.param_namespace_end` commands.

The general hierarchical behavior is similar to that used by the SPICE `.subckt` approach—parameters exist at the namespace level in which they are defined and in all the child namespaces. However, unlike SPICE subckts, in case of a naming conflict, the parameter from the top-most namespace takes precedence.

The `.param_namespace_begin` keyword must be followed by the namespace name or a list of namespace names connected together with a dot, creating a hierarchy. Appending names using a dot creates a child namespace. For example:

```
.param_namespace_begin child1
.param vdc=10
.param_namespace_end

.param_namespace_begin child1.child2
.param vdc=5
.param pwr=1+vdc
Vcc power 0 {pwr}

.param_namespace_end
```

The parameter `pwr` will have a value of 11 because parameter `vdc` from the parent namespace `child1` overrides parameter `vdc` from the child namespace `child2`.

Do not directly embed namespaces into one another.

# Number Format

Numbers are standard floating point or integer numbers with optional suffixes. Any characters that follow directly, unless they are mathematical operators, are ignored.

Number suffixes:

| Suffix | Meaning | Multiplier |
|---|---|---|
| f, F | femto | 1e-15 |
| p, P | pico | 1e-12 |
| n, N | nano | 1e-9 |
| u, U | micro | 1e-6 |
| m, M | milli | 1e-3 |
| k, K | Kilo | 1e3 |
| meg, MEG | Mega | 1e6 |
| g, G | Giga | 1e9 |
| t, T | Tera | 1e12 |

**Caution** P in a SPICE netlist means pico, but peta in circuit parameters. Similarly, M means milli in a SPICE netlist, and mega in circuit parameters.

**Examples**

R1 1 2 10mohm

*the following voltage source has a DC value of 0.00000521
V1 4 0 {0.21e-6+5uV}

# Comments and Line Continuation

The asterisk (*) and the semicolon (;) characters can be used to comment out individual line of circuit text.

The semicolon (;) is used within a line of text to comment out everything to the right.

A plus (+) is used to continue a SPICE statement from the previous line.

The comment works on a per line basis—a single asterisk does not comment out an entire SPICE statement spread over multiple lines.

**Examples**

*This example shows how comments work
V1 1 0 10 ;V1 is in the circuit, but this is just a comment
R1 1 2 10k

*Below, Resistor R2 is taken out of the circuit
*R2 1 2 10k

*In the following, we only eliminate the ‘n’ parameter from the multiline SPICE statement
D1 2 0 myDiode
.model myDiode d(Is=1e-12
*+n=1.2
+rs=3.7)

# Mathematical Expressions

You can create arbitrary mathematical expressions consisting of various functions and operators and apply the results to device parameters. Expressions are very useful modeling tools when used within the Arbitrary Source devices. Refer to the *Arbitrary Sources* section for more information.

The functions and operators in the expression can operate on numerical constants, on circuit parameters, and, when used within the Arbitrary Source device, on live circuit variables. Refer to the *Supported Mathematical Functions, Operators and Constants* section for a list of supported mathematical functions, operators and pre-defined constants.

Within the Arbitrary Source device, special functions `V(nodeabs)`, `V(node+,node-)`, and `I(deviceX)` can be used to reference the circuit voltages and currents.

- `V(nodeabs)` references the voltage at node nodeabs relative to ground.
- `V(node+,node-)` references the difference between node+ and node-.
- `I(deviceX)` references the current through the device with the instance name `deviceX`. Currently only the Independent Voltage source, dependent voltage sources, and inductor devices are supported.

### Additional Notes

- We recommend that expressions be enclosed in {} to avoid ambiguous syntax. However, this is not required for simulation.
- Portions of expressions that are not enclosed in {} or () should not contain any spaces.

### Examples

*High-level filter specification
.param pole=1k
.param res_val=1k
R1 in out {res_val}
C1 out 0 {1/(2*pi*res_val*pole)}

*A very simple diode modeled using an expression in an Arbitrary source
G1 A C value={1e-12*(e^(V(A,C)/0.025)-1 ) }

# Supported Mathematical Functions, Operators and Constants

Mathematical functions:

| Function Name | Alternate Name | Parameters | Description | Notes |
|---|---|---|---|---|
| if | — | (test,a,b) | If-else function. If the test returns true, the result is a, else it returns b. | Example:<br>B1 out 0 V={if(v(1)>5, v(1)**2, 0)} |
| sin | sine | ( x ) | Sine function. | — |
| asin | arcsin | ( x ) | Arc-sine function. | — |
| sinh | — | ( x ) | Hyperbolic sine function. | — |
| asinh | arcsinh | ( x ) | Arc-hyperbolic sine function. | — |
| cos | — | ( x ) | Cosine function. | — |
| acos | arccos | ( x ) | Arc-cosine function. | — |
| cosh | — | ( x ) | Hyperbolic cosine function. | — |
| acosh | arccosh | ( x ) | Arc-hyperbolic cosine function. | — |
| tan | — | ( x ) | Tangent function. | — |
| tanh | — | ( x ) | Hyperbolic tangent function. | — |
| atan | arctan | ( x ) | Arc-tangent function. | — |
| atanh | arctanh | ( x ) | Arc-hyperbolic tangent function. | — |
| atan2 | — | ( x, y ) | Atan 2 function. | Same as:<br>atan (x/y) |
| exp | — | ( x ) | Calculates the exponential $e^x$. | — |
| expl | — | ( x, y ) | Calculates the exponential with a maximum value. | Same as:<br>min (exp (x), y) |
| ln | — | ( x ) | The natural logarithm function. | — |

| Function Name | Alternate Name | Parameters | Description | Notes |
|---|---|---|---|---|
| log10 | | ( x ) | The base-10 logarithm function. | |
| log | | ( x ) | The generic logarithm function.<br>This function is included for legacy compatibility and its use is not recommended. | Normally this is the same as ln(x) except in 'B' arbitrary sources where it is log10(x). This behavior is also changed by the .SYNTAX statements. |
| sqrt | — | ( x ) | Square root function. | — |
| abs | — | ( x ) | Absolute value function. | — |
| sgn | sign | ( x ) | Sign or signum function. | if (x < 0)<br>sgn(x) == -1<br>if (x > 0)<br>sgn(x) == 1<br>if (x == 0)<br>sgn(x) == 0 |
| max | — | ( x, y ) | Returns the maximum of x and y. | — |
| min | — | ( x, y ) | Returns the minimum of x and y. | — |
| uramp | — | ( x ) | Ramp function, clips the value against a minimum of 0. | if (x < 0)<br>uramp(x) == 0<br>if (x >= 0)<br>uramp = x |
| u | stp, step | ( x ) | Step function. | if (x < 0)<br>u(x) == 0<br>if (x > 0)<br>u(x) == 1<br>if (x == 0)<br>u(x) == 0.5 |

| Function Name | Alternate Name | Parameters | Description | Notes |
|---|---|---|---|---|
| table | — | (expr, x1,y1, <x2,y2 <…>>) | Piecewise linear function. Specified x-values must be increasing in value from left to right.<br>Inputs below x1 result in an output equal to y1. Inputs above xn (the largest specified x-value), result in an output equal to yn. In other words, table function acts as a limiter in those regions.<br>Using this function in a Value-type Arbitrary source is functionally equivalent to using the Table-type source. | The following line:<br>E1 out 0 value={Table(v(in), -60m,-4,0,0,140m,43.3 )}<br>is functionally equivalent to the following line:<br>E1 out 0 TABLE {v(in)} (-60m,-4,0,0,140m,43. 3) |
| limit | — | ( x, a, b ) | Clips the input value x to the range (A,B). | — |
| pwr | — | ( x, y ) | The pwr function. | Same as:<br>abs(x)^y |
| pwrs | — | ( x, y ) | The pwrs function. | if (x < 0)<br>pwrs(x) == -(x**y)<br>if (x >= 0)<br>pwrs(x) == (x**y) |
| ddt | | (x) | Time Derivative | ddt(v(1)*v(CapVoltage node)) |
| sdt | | (x) | Time Integral | 1meg*sdt(I(Vsense)+8 ) |
| V | — | (node) | Voltage of a node. | Can only be used in non-linear controlled source expressions. |

| Function Name | Alternate Name | Parameters | Description | Notes |
|---|---|---|---|---|
| V | — | (node1, node2) | Voltage difference of two nodes. | Can only be used in non-linear controlled source expressions.<br>Same as:<br>V(node1)-V(node2)<br>Example:<br>E1 out 0<br>value={V(1,2)*15} |
| I | — | (voltage source) or (inductor) | Current through a voltage source or an inductor. | Can only be used in non-linear controlled source expressions.<br>Example:<br>E1 out 0<br>value={I(Vsense)**2 + I(E99)} |
| positive | — | ( x ) | Ensure positive function. | if (x < d)<br>positive(x) = d<br>else<br>positive(x) = x<br>where d is 1.0p |
| negative | — | ( x ) | Ensure negative function. | if (x > -d)<br>negative(x) = -d<br>else<br>negative(x) = x<br>where d is 1.0p |
| nonpos | — | ( x ) | Ensure not-positive function. | if (x > 0)<br>nonpos(x) = 0<br>else<br>nonpos(x) = x |
| nonneg | — | ( x ) | Ensure not-negative function. | if (x < 0)<br>nonneg(x) = 0<br>else<br>nonneg(x) = x |
| nonzero | — | ( x ) | Ensure non-zero function. | if (x < d) & (x >= 0)<br>nonzero(x) = d<br>if (x > -d) & (x <= 0)<br>nonzero(x) = -d<br>else<br>nonzero(x) = x<br>where d is 1.0p |

| Function Name | Alternate Name | Parameters | Description | Notes |
|---|---|---|---|---|
| zero | — | ( x ) | Evaluates x but always returns a value of 0.0. | — |
| one | — | ( x ) | Evaluates x but always returns a value of 1.0. | — |
| schedule | — | (x1,y1, <x2,y2 <…>>) | Schedule function. Gives a value of yN for when time between xN and xN+1. | Time is simulation time. |

Mathematical operators:

| Symbol | Alternate Symbol | Description | Usage |
|---|---|---|---|
| + | | Addition. | A + B |
| - | | Subtraction. | A - B |
| / | | Division. | A / B |
| * | | Multiplication. | A * B |
| ** | ^ | Exponentiation (power). | A ** B or A ^ B<br>This can be interpreted differently in other simulators. |
| < | | Less than. | A < B |
| <= | | Less than or equal to. | A <= B |
| > | | Greater than. | A > B |
| >= | | Greater than or equal to. | A >= B |
| == | | Equal to. | A == B |
| != | | Not equal to. | A != B |
| & | | Logical AND. | A & B |
| \| | | Logical OR. | A \| B |
| xor | | Logical XOR. | A XOR B |
| ? : | | Ternary if. This operator has two symbols and three operands. | A ? B : C<br>This is the same as IF(A,B,C). |

Built-in constants:

| Symbol | Description | Value |
|---|---|---|
| true | Boolean true value. | 1.0 |
| false | Boolean false value. | 0.0 |
| yes | Alternate form of boolean true value. | 1.0 |
| no | Alternate form of boolean false value. | 0.0 |
| pi | The constant pi. | 3.14159265358979323846 |
| e | The constant e. | 2.71828182844590452353 |
| c | Speed of light. | 2.99792458e8 |
| kelvin | Constant to convert between degrees Kelvin and Celsius, and vice versa. | -273.0 |
| echarge | Electron charge. | 1.602176487e-19 |
| boltz | Boltzmann's constant. | 1.3806503e-23 |
| planck | Planck's constant. | 6.62606896e-34 |
| temp | Current temperature of the simulation in degrees Celsius. | The default is 27, but it can be changed from the simulation options. |
| time | Current time of the simulation in seconds. | Current time of the simulation in seconds (it is constant with respect to circuit variables). |

### Additional Notes

- IMPORTANT! Unlike most mathematical languages, Multisim considers the unary minus to have higher precedence than exponentiation (power) operators ** and ^. This means that {-5**2} is +25 while {0-5**2} is -25. Although this is unintuitive, it is standard among SPICE simulators. Use brackets to ensure logical, readable expressions in this case.

### Related Information

*VALUE Type Source*
*Compatibility Modes*

## User-Defined Functions

Similar to the way subcircuits provide modularity to connecting together blocks of circuits, user-defined functions provide modularity for using mathematical expressions.

### Declaration

`.FUNC my_function_name ( Arg1 <, Arg2 <...>> ) = valueexpression`

| Identifier | Description |
|---|---|
| my_function_name | Arbitrary function name. |
| ArgN<br>where N=1,2,3 | List of arguments used by the function. |
| valueexpression | Mathematical Expression operating on the arguments |

### Additional Notes

- User-defined functions can be called from any field where a mathematical expression is used. However, only the Arbitrary Source device can call a function with circuit variables (node voltages and device currents) in the arguments.
- Function names may contain numbers, letters and underscores but must not start with a number and may not contain any other symbols.
- The value expression may be enclosed in {} for clarity, but this is not mandatory.
- User-defined functions take precedence over pre-defined functions; if you define a function called `sin` it will take the place of the standard sin function within that context.

### Examples

```
.FUNC sinplusn ( angle, n ) = {sin(angle) + n}
.FUNC myfunc(a,b,c)=a+sinplusn(b**c,0.1)
.param foo=myfunc(1,2,3)

E1 60 59 value={5+myfunc(v(2),3*v(3),9)}
G1 0 88 value={myfunc(I(Vref), 1, 2}
```

# Analog Devices Library

Analog devices are devices (lowest level modeling elements) that are simulated by the analog simulation engine. The following sections describe all such devices supported in Multisim.

### Related Information

*Resistor*
*Capacitor*
*Inductor*
*Coupled (Mutual) Inductor*
*Diode*
*Lossless Transmission Line*
*Lossy Transmission Line*

## Resistor

Resistor instance declaration syntax:

Rxxxx node1 node2 resistance <TC=tc1 <,tc2> > <TEMP=temp>

Rxxxx node1 node2 resistance <TC1=tc1> <TC2=tc2> <TEMP=temp>

Rxxxx node1 node2 <resistance> <Model> <L=l> <W=w>

Resistor instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| resistance | Device resistance. |
| L | Device length. |
| W | Device width. |
| TC | Instance temperature coefficients. This is a two element vector for specifying TC1 and TC2. |
| TC1 | Alternate way of specifying 1st-order temperature coefficient. |
| TC2 | Alternate way of specifying 2nd-order temperature coefficient. |
| SENS_RESIST | [FLAG] Flag to request sensitivity with respect to resistance. |
| TEMP | Instance operating temperature. |

The following only applies if a model has been specified in the instance declaration as it is not mandatory for resistors.

Resistor model definition syntax:

`.MODEL mymodelname R ( <TC1=tc1 <TC2=tc2>> <Other_Model_Parameters...> )`

`.MODEL mymodelname RES ( <TC1=tc1 < TC2=tc2>> <Other_Model_Parameters...> )`

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| DEFW | Default device width. | m | $1x10^{-6}$ |
| NARROW | Narrowing due to side etching. | m | 0.0 |
| R | Resistance multiplier (same as RMULT). | — | 1.0 |
| RMULT | Resistance multiplier. | — | 1.0 |
| RSH | Sheet resistance. | Ω/sq. | 0.0 |
| TC1 | Instance 1st-order temperature coefficient. | 1/ºC | 0.0 |
| TC2 | Instance 2nd-order temperature coefficient. | $1/(ºC^2)$ | 0.0 |
| TCE | Exponential temperature coefficient. | $\%/(ºC^2)$ | 0.0 |
| TNOM | Temperature at which model parameters were measured. | ºC | — |
| T_ABS | Absolute temperature. | ºC | — |
| T_MEASURED | Temperature at which model parameters were measured. | ºC | — |
| T_REL_GLOBAL | Temperature delta relative to global temperature. | ºC | — |
| T_REL_LOCAL | Temperature delta relative to AKO model temperature. | ºC | — |

**Examples**
R1 1 0 4.7k
R2 1 0 10k myRes
.model myRes r(tc1=1e-4)

## Resistor Equations

The resistor model is ideal. It is described by the following the equation:

$$I = \frac{V}{RMULT \cdot R}$$

If the capacitance is specified using geometric parameters RSH, L, W, and NARROW, the resistance R is calculated using the following equation:

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

### Temperature dependent parameters

The capacitance value is temperature-dependent and is adjusted using the following formula:

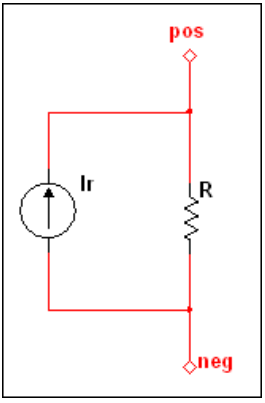
$$R(T) = R \cdot (1 + TC1 \cdot (T - TNOM) + TC2 \cdot (T - TNOM)^2)$$

T is the operating temperate and TNOM is the nominal (or measured temperature). T and TNOM can be adjusted in a number of ways. Refer to the *Device Temperature Parameters* section for more information.

### Noise model

The resistor has a thermal noise generator, $\overline{Ir}^2$ , as a result of its inherent resistance.

$$\overline{Ir}^2 = \frac{4kT\Delta f}{R}$$




# Capacitor

Capacitor instance declaration syntax:

Cxxxx node1 node2 capacitance <IC=vc0> <TC=tc1 <,tc2> >

Cxxxx node1 node2 capacitance <Model> <L=L> <W=W> <IC=vc0>

Capacitance instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| capacitance | Device capacitance. |
| IC | Initial capacitor voltage. |
| TC | Instance temperature coefficients. This is a two element vector for specifying TC1 and TC2. |
| L | Device length. |
| W | Device width. |
| SENS_CAP | [FLAG] Flag to request sensitivity with respect to capacitance. |
| TEMP | Instance temperature. |

The following only applies if a model has been specified in the instance declaration as it is not mandatory for capacitors.

Capacitor model definition syntax:

`.MODEL` mymodelname `C ( <TC1=tc1 <TC2=tc2>> <VC1=vc1 <VC2=vc2>> +` *<Other_Model_Parameters...>* `)`

Capacitor model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| CJ | Junction bottom capacitance per area. | $F/(m^2)$ | 0.0 |
| CJSW | Junction sidewall capacitance per meter. | F/m | 0.0 |
| CMULT | Capacitance multiplier. | — | 1.0 |
| DEFW | Default width. | m | $1 \times 10^{-6}$ |
| NARROW | Narrowing due to side etching. | m | 0.0 |
| TNOM | Temperature at which model parameters were measured. | ºC | — |
| T_ABS | Absolute temperature. | ºC | — |
| T_MEASURED | Temperature at which model parameters were measured. | ºC | — |
| T_REL_GLOBAL | Temperature delta relative to global temperature. | ºC | — |
| T_REL_LOCAL | Temperature delta relative to AKO model temperature. | ºC | — |

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| TC1 | 1st-order temperature coefficient. | 1/ºC | 0.0 |
| TC2 | 2nd-order temperature coefficient. | 1/(ºC$^2$) | 0.0 |
| VC1 | 1st-order voltage coefficient. | 1/V | 0.0 |
| VC2 | 2nd-order voltage coefficient. | 1/(V$^2$) | 0.0 |

**Examples**

c1 1 0 1u
C2 1 0 1e-12 myCap
.model myCap C(tc1=1e-4)

### Capacitor Equations

The capacitor model is ideal. It is described by the following the equation:

$$I = CMULT \cdot C \cdot \frac{dV}{dT}$$

If the capacitance is specified using geometric parameters CJ, CJSW, L, W, and NARROW, the capacitance C is calculated using the following equation:

$$C = CJ \cdot (L - NARROW) \cdot (W - NARROW) + 2 \cdot CJSW(L + W - 2 \cdot NARROW)$$

**Temperature dependent parameters**

The capacitance value is temperature-dependent and is adjusted using the following formula:

$$C(T) = C \cdot (1 + TC1 \cdot (T - TNOM) + TC2 \cdot (T - TNOM)^2)$$

T is the operating temperate and TNOM is the nominal (or measured temperature). T and TNOM can be adjusted in a number of ways. Refer to the *Device Temperature Parameters* section for more information.

## Inductor

Inductor instance declaration syntax:

Lxxxx node1 node2 <Model> inductance <IC=iL0>

Inductor instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| inductance | Device inductance. |
| IC | Initial current through inductor. |

| Parameter Name | Parameter Description |
|---|---|
| SENS_IND | [FLAG] Flag to request sensitivity with respect to inductance. |
| TEMP | Instance temperature. |

The following only applies if a model has been specified in the instance declaration as it is not mandatory for inductor.

Inductor model definition syntax:

.MODEL mymodelname L ( <*Other_Model_Parameters…*> )

.MODEL mymodelname IND ( <*Other_Model_Parameters…*> )

Inductor model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| IL1 | 1st-order current coefficient. | 1/A | 0.0 |
| IL2 | 2nd-order current coefficient. | $1/(A^2)$ | 0.0 |
| LMULT | Inductance multiplier. | — | 1.0 |
| TNOM | Temperature at which model parameters were measured. | ºC | — |
| T_ABS | Absolute temperature. | ºC | — |
| T_MEASURED | Temperature at which model parameters were measured. | ºC | — |
| T_REL_GLOBAL | Temperature delta relative to global temperature. | ºC | — |
| T_REL_LOCAL | Temperature delta relative to AKO model temperature. | ºC | — |
| TC1 | 1st-order temperature coefficient. | 1/ºC | 0.0 |
| TC2 | 2nd-order temperature coefficient. | $1/(ºC^2)$ | 0.0 |

**Examples**

L1 1 0 1m
l2 1 0 1e-12 myInd
.model myInd IND(tc1=1e-4)

### Inductor Equations

The inductor model is ideal. It is described by the following the equation:

$$V = LMULT \cdot L \cdot \frac{dI}{dT}$$

#### Temperature dependent parameters

The inductance value is temperature-dependent and is adjusted using the following equation:

$$L(T) = L \cdot (1 + TC1 \cdot (T - TNOM) + TC2 \cdot (T - TNOM)^2)$$

T is the operating temperate and TNOM is the nominal (or measured temperature). T and TNOM can be adjusted in a number of ways.

#### Related Information

*Device Temperature Parameters*

## Coupled (Mutual) Inductor

Coupled inductor instance declaration syntax:

Kxxxx Lname1 Lname2 <Lname3 <LnameN>>

Coupled inductor instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| k | Mutual inductance coupling coefficient (0.0 to 1.0). |
| LnameN | Name of Nth coupled inductor. |

#### Description

This device is used to introduce mutual inductance between inductors. Each inductor listed on the instance statement of this device is coupled to each of the other inductors specified on the instance statement. The mutual inductance between any two inductors is given by:

$$M_{12} = k * \sqrt{L_1 L_2}$$

Thus, in a set of inductors L1,L2,L3…LN, the voltage across inductor L1 is given by:

$$V_{L1} = L_1 \frac{dI_{L1}}{dt} + M_{12} \frac{dI_{L2}}{dt} + M_{13} \frac{dI_{L3}}{dt} + \ldots + M_{1N} \frac{dI_{LN}}{dt}$$

Positive current is referenced as current flowing into the positive node of the inductor (the first node in the inductor's instance line).

**Examples**
L1 1 0 1m
L2 77 0 2m
L3 88 0 5m
K1 L1 L2 L3 0.99

# Diode

Diode instance declaration syntax:

Dxxx AnodeNode CathodeNode Model <area> <OFF> <IC=Vd0> <TEMP=temp>

Diode instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| area | Area factor. |
| OFF | Initially off. |
| IC | Initial device voltage. |
| TEMP | Instance temperature. |

Diode model definition syntax:

.MODEL mymodelname D ( <*Other_Model_Parameters*...> )

Diode model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| AF | Flicker noise exponent. | — | 1.0 |
| BV | Reverse breakdown voltage. | V | infinite |
| CJO | Junction capacitance. | F | 0.0 |
| EG | Bandgap voltage. | eV | 1.11 |
| FC | Forward bias junction fit parameter. | — | 0.5 |
| IBV | Current at reverse breakdown voltage. | A | $1 \times 10^{-10}$ |
| IBVL | Low-level reverse breakdown knee current. | A | 0.0 |
| IKF | High-injection knee current. | A | infinite |
| IS | Saturation current. | A | $1 \times 10^{-14}$ |
| ISR | Recombination currrent parameter. | A | 0.0 |
| KF | Flicker noise coefficient. | — | 0.0 |
| M | Grading coefficient. | — | 0.5 |

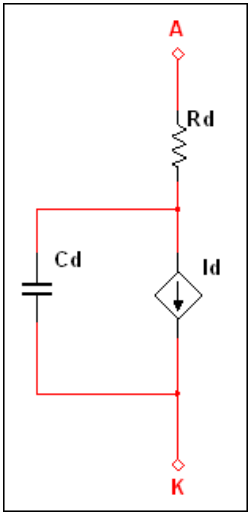
| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| N | Emission coefficient. | — | 1.0 |
| NBV | Reverse breakdown ideality factor. | — | 1.0 |
| NBVL | Low-level reverse breakdown ideality factor. | — | 1.0 |
| NR | Emission coefficient for ISR. | — | 2.0 |
| RS | Ohmic resistance. | Ω | 0.0 |
| TBV1 | Linear BV temperature coefficient. | 1/ºC | 0.0 |
| TBV2 | Quadratic BV temperature coefficient. | 1/(ºC$^2$) | 0.0 |
| TIKF | Linear IKF temperature coefficient. | 1/ºC | 0.0 |
| TNOM | Parameter measurement temperature. | ºC | — |
| TRS1 | Linear RS temperature coefficient. | 1/ºC | 0.0 |
| TRS2 | Quadratic RS temperature coefficient. | 1/(ºC$^2$) | 0.0 |
| TT | Transit time. | s | 0.0 |
| VJ | Junction potential. | V | 1.0 |
| XTI | Saturation current temperature exponent. | — | 3.0 |

### Examples

*diode with area scale of 2
d1 1 0 myDiode 2
.model myDiode d(is=1.1p)

## Diode Equations

### Large signal model




In the following equations, $V_d$ is the voltage across source Id. It does not include the drop across Rd.

### Static equations

The current through the diode is the sum of the forward current, $I_{fwd}$, and reverse current, $I_{rev}$:

$$I_d = area * (I_{fwd} - I_{rev})$$

### Forward current

The forward current is the sum of the normal and recombination currents:

$$I_{fwd} = I_{nrm} \cdot K_{inj} + I_{rec} \cdot K_{gen}$$

$$I_{nrm} = IS \cdot \left( \exp\left( \frac{V_d}{N \cdot V_t} \right) - 1 \right)$$

$$K_{inj} = \begin{cases} 1, & IKF \leq 0 \\ \left( \frac{IKF}{IKF + I_{nrm}} \right)^{0.5}, & IKF > 0 \end{cases}$$

$$I_{rec} = ISR \cdot \left( \exp\left( \frac{V_d}{NR \cdot V_t} \right) - 1 \right)$$

$$K_{gen} = \left( \left( 1 - \frac{V_d}{VJ} \right)^2 + 0.005 \right)^{M/2}$$

Notice that with the default parameter values, most of the terms drop out or the factors are set to 1. Therefore the forward current simply degenerates into the normal current, $I_{nrm}$.

### Reverse current

The reverse current is the sum of the high and low reverse currents:

$$I_{rev} = IBV \cdot \exp\left( \frac{V_d + BV}{NBV \cdot V_t} \right) + IBVL \cdot \exp\left( \frac{V_d + BV}{NBVL \cdot V_t} \right)$$

### Capacitance equations

The diode's non-linear capacitance, $C_d$, is the sum of the diffusion capacitance ( $TT \frac{dI_d}{dV_d}\ term$ ) and the junction capacitance.

$$C_d = \begin{cases} TT \frac{dI_d}{dV_d} + area \cdot CJO \cdot \left( 1 - \frac{V_d}{VJ} \right)^{-M}, & V_d < FC \cdot VJ \\ TT \frac{dI_d}{dV_d} + area \cdot \frac{CJO}{(1 - FC)^{1+M}} \cdot \left( 1 - FC \cdot (1 + M) + \frac{M \cdot V_d}{VJ} \right), & V_d \geq FC \cdot VJ \end{cases}$$

### Temperature dependent parameters

The following parameters are functions of temperature. T is the operating temperate and TNOM is the nominal (or measured temperature). T and TNOM can be adjusted in a number of ways.

$$IS(T) = IS \cdot \left(\frac{T}{TNOM}\right)^{\frac{XTI}{N}} \cdot EXP\left(\left(\frac{T}{TNOM} - 1\right) \cdot \frac{EG}{N \cdot V_t}\right)$$

$$ISR(T) = ISR \cdot \left(\frac{T}{TNOM}\right)^{\frac{XTI}{NR}} \cdot EXP\left(\left(\frac{T}{TNOM} - 1\right) \cdot \frac{EG}{N \cdot V_t}\right)$$

$$IKF(T) = IKF \cdot \left(1 + TIKF \cdot (T - TNOM)\right)$$

$$BV(T) = BV \cdot (1 + TBV1 \cdot (T - TNOM) + TBV2 \cdot (T - TNOM)^2)$$

$$RS(T) = RS \cdot (1 + TRS1 \cdot (T - TNOM) + TRS2 \cdot (T - TNOM)^2)$$

$$VJ(T) = VJ \cdot \frac{T}{TNOM} - 3 \cdot V_t \cdot \ln\left(\frac{T}{TNOM}\right) - Eg(TNOM) \cdot \frac{T}{TNOM} + Eg(T)$$

$$CJ0(T) = CJ0 \cdot \left(1 + M \cdot \left(0.004 \cdot (T - TNOM) + \left(1 - \frac{VJ(T)}{VJ}\right)\right)\right)$$

*where*

$V_t = \frac{K \cdot T}{q}$, *where K is Boltzmann's constant and q is electron charge*

$$Eg(T) = 1.16 - \frac{0.000702 \cdot T^2}{T + 1108}$$

### Noise model

The diode has a thermal noise generator, $\overline{Ird^2}$ , as a result of the series ohmic resistance, and the shot and flicker noise generators, collectively $\overline{Ipn^2}$ , as a result of the PN junction.

Ohmic resistance noise:

$$\overline{Ird^2} = \frac{4kT\Delta f}{RS}$$

Shot and flicker noise:

$$\overline{Ipn^2} = 2qI_d\Delta f + \frac{KF \cdot {I_d}^{AF}}{f}\Delta f$$

### References


1. G. Massobrio and P. Antognetti, Semiconductor Device Modeling with SPICE, 2nd edition, McGraw-Hill, 1993.
2. A. Vladimirescu, The SPICE Book, Wiley, 1994.


### Related Information

*Device Temperature Parameters*

## Lossless Transmission Line

Lossless transmission line instance declaration syntax:

Txxx nodeP1+ nodeP1- nodeP2+ nodeP2- Z0=z0 <TD=td> <F=freq <NL=nl> <IC=v1 <,i1 <,v2 <,i2>>>> <REL=rel> <ABS=abs>

Txxx nodeP1+ nodeP1- nodeP2+ nodeP2- Z0=z0 <TD=td> <F=freq <NL=nl> <V1=v1> <I1=i1> <V2=v2> <I2=i2>
<REL=rel> <ABS=abs>

Lossless Transmission Line instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| ZO | Characteristic impedance. |
| TD | Transmission delay. |
| F | Frequency. |
| NL | Normalized length a frequency given. |
| v1 | Initial voltage v1. |
| i1 | Initial current i1. |
| v2 | Initial voltage v2. |
| i2 | Initial current i2. |
| REL | Relative rate of change of derivative for breakpoint. |
| ABS | Absolute rate of change of derivative for breakpoint. |

This device does not have an associated model definition.

### Example

T1 1 0 2 0 Z0=75

# Lossy Transmission Line

Lossy Transmission Line instance declaration syntax:

Oxxx node1 node2 node3 node4 Model <IC=v1 <,i1 <,v2 <,i2>>>>

Oxxx node1 node2 node3 node4 Model <V1=v1> <I1=i1> <V2=v2> <I2=i2>

Lossy Transmission Line instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| v1 | Initial voltage at terminal 1. |
| v2 | Initial voltage at terminal 2. |
| i1 | Initial current at terminal 1. |
| i2 | Initial current at terminal 2. |

Lossy Transmission Line model definition syntax:

.MODEL mymodelname LTRA ( <NOCONTROL> <STEPLIMIT/NOSTEPLIMIT>
+ <LININTERP/QUADINTERP/MIXEDINTERP> <*Other_Model_Parameters…*> )

Lossy Transmission Line model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| C | Capacitance per meter. | F/m | 0.0 |
| G | Conductance per meter. | S/m | 0.0 |
| L | Inductance per meter. | H/m | 0.0 |
| R | Resistance per meter. | Ω/m | 0.0 |
| LEN | Length of line. | — | required |
| REL | Relative rate of change of derivative for breakpoint. | — | 1.0 |
| ABS | Absolute rate of change of derivative for breakpoint. | — | 1.0 |
| NOCONTROL | [FLAG] No timestep control. | — | — |
| STEPLIMIT | [FLAG] Always limit timestep to 0.8*(delay of line). | — | — |
| NOSTEPLIMIT | [FLAG] Don’t always limit timestep to 0.8*(delay of line). | — | — |
| LININTERP | [FLAG] Use linear interpolation. | — | — |

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| QUADINTERP | [FLAG] Use quadratic interpolation. | — | — |
| MIXEDINTERP | [FLAG] 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. | — | — |

**Example**

O1 1 0 2 0 myLossyLine
.model myLossyLine LTRA(r=3.5 L=3m g=1e-6 c=3.2e-6)

# Uniform R.C. Line (Lumped-approximation R.C. line)

Uniform R.C. line instance declaration syntax:

Uxxx node1 node2 nodeRef Model L=len <N=lumps>

Uniform R.C. line instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| L | Length of transmission line. |
| N | Number of lumps. |

Uniform R.C. line model definition syntax:

.MODEL mymodelname URC ( <*Other_Model_Parameters…*> )

Uniform R.C. line model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| CPERL | Capacitance per unit length. | F/m | $1x10^{-12}$ |
| FMAX | Maximum frequency of interest. | Hz | $1x10^{9}$ |
| ISPERL | Saturation current per length. | A/m | — |
| K | Propagation constant. | — | 1.5 |
| RPERL | Resistance per unit length. | Ω/m | 1000 |
| RSPERL | Diode resistance per length. | Ω/m | — |

**Example**
U1 1 2 0 myURC
.model myURC URC(isperl=1e-9)

# JFET

JFET instance declaration syntax:

Jxxx nodeDrain nodeGate nodeSource Model <area> <OFF> <IC=vds0, vgs0>

JFET instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| area | Area factor. |
| OFF | [FLAG] Device is initially off. |
| IC-VDS | Initial drain-to-source voltage. |
| IC-VGS | Initial gate-to-source voltage. |
| IC | Initial voltages. This is a two element vector alternate way of specifying IC-VDS, IC-VGS. |
| TEMP | Instance temperature. |

N-channel JFET model definition syntax:

.MODEL mymodelname NJF ( <*Other_Model_Parameters*...> )

P-channel JFET model definition syntax:

.MODEL mymodelname PJF ( <*Other_Model_Parameters*...> )

JFET model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| AF | Flicker noise exponent. | — | 1.0 |
| ALPHA | Ionizing coefficient. | 1/V | 0.0 |
| B | Doping tail parameter. | — | 1.0 |
| BETA | Transconductance parameter. | $A/(V^2)$ | $1x10^{-4}$ |
| BETATCE | BETA exponential temperature coefficient. | %/ºC | 0.0 |
| CGD | Zero-bias gate-to-drain junction capacitance. | F | 0.0 |
| CGS | Zero-bias gate-to-source junction capacitance. | F | 0.0 |
| FC | Forward bias junction fit parameter. | — | 0.0 |

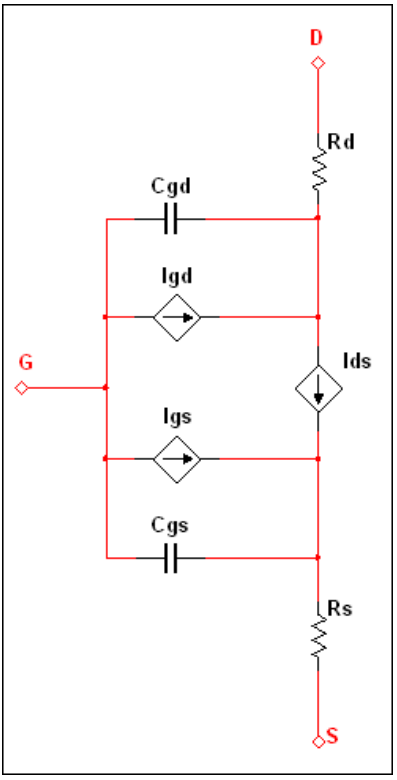
| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| IS | Gate junction saturation current. | A | $1x10^{-14}$ |
| ISR | Gate junction recombination current parameter. | A | 0.0 |
| KF | Flicker noise coefficient. | — | 0.0 |
| LAMBDA | Channel length modulation parameter. | 1/V | 0.0 |
| M | Gate junction grading coefficient. | — | 0.5 |
| N | Gate junction emission coefficient. | — | 1.0 |
| NR | Emission coefficient for ISR. | — | 2.0 |
| PB | Gate junction potential. | V | 1.0 |
| RD | Drain ohmic resistance. | Ω | 0.0 |
| RS | Source ohmic resistance. | Ω | 0.0 |
| TNOM | Temperature at which model parameters were measured. | ºC | — |
| T_ABS | Absolute temperature. | ºC | — |
| T_MEASURED | Temperature at which model parameters were measured. | ºC | — |
| T_REL_GLOBAL | Temperature delta relative to global temperature. | ºC | — |
| T_REL_LOCAL | Temperature delta relative to AKO model temperature. | ºC | — |
| VK | Ionization knee voltage. | V | 0.0 |
| VTO or VT0 | Threshold voltage. | V | -2.0 |
| VTOTC | VTO temperature coefficient. | V/ºC | 0.0 |
| XTI | IS temperature coefficient. | — | 3.0 |

**Example**

J1 d g s myJFET
.model myJFET NJF(vto=1.3)

# JFET Equations

### Large signal model




All transistor node voltage references are with respect to the internal nodes in the following equations (that is, the ohmic resistance pin that is connected the inside of the structure.)

### Static equations

Drain-source current:

$$\textbf{\textit{Linear Region}}: 0 < V_{DS} < V_{GS} - VT0$$

$$I_{DS} = BETA \cdot (2 \cdot (V_{GS} - VT0) - V_{DS}) \cdot (1 + LAMBDA \cdot V_{DS})$$

$$\textbf{\textit{Saturation Region}}: 0 < VGS - VT0 < VDS$$

$$I_{DS} = BETA \cdot (V_{GS} - VT0)^2 \cdot V_{DS} \cdot (1 + LAMBDA \cdot V_{DS})$$

$$\textbf{\textit{Cut} - \textit{off region}}: V_{GS} < VT0$$

$$I_{DS} = 0$$

### Gate currents

$$I_{gate} = area * (I_{GS} + I_{GD})$$

$$I_{GS} = IS \cdot \left(\exp\left(\frac{V_{GS}}{N \cdot V_T}\right) - 1\right) + \text{ISR} \cdot \left(\exp\left(\frac{V_{GS}}{NR \cdot V_T}\right) - 1\right) \cdot kg$$

$$where\ kg = generation\ factor = \left(\left(1 - \frac{V_{GS}}{PB}\right)^2 + 0.005\right)^{\frac{M}{2}}$$

$$I_{GD} = IS \cdot \left(\exp\left(\frac{V_{GD}}{N \cdot V_T}\right) - 1\right) + \text{ISR} \cdot \left(\exp\left(\frac{V_{GD}}{NR \cdot V_T}\right) - 1\right) \cdot kg + \text{Iion}$$

$$where\ kg = \left(\left(1 - \frac{V_{GD}}{PB}\right)^2 + 0.005\right)^{\frac{M}{2}} = generation\ factor$$

$$where\ Iion = Impact\ ionization\ current$$

$$If\ 0 < Vgs - VTO < Vds\ (Saturation\ region)$$

$$Iion = area \cdot (I_{DS} - I_{GD}) \cdot ALPHA \cdot (V_{DS} + VT0 - V_{GS}) \cdot exp\left(\frac{-VK}{V_{DS} + VT0 - V_{GS}}\right)$$

$$else$$

$$Iion = 0$$

$$V_t = \frac{K \cdot T}{q}, where\ K\ is\ Boltzmann's\ constant\ and\ q\ is\ electron\ charge$$

### Capacitances

$$For\ V_{GS}, V_{GD} < FC \cdot PB$$

$$C_{GS} = area \cdot CGS \cdot \left(1 - \frac{V_{GS}}{PB}\right)^{-M}$$

$$C_{GD} = area \cdot CGD \cdot \left(1 - \frac{V_{GD}}{PB}\right)^{-M}$$

$$For\ V_{GS}, V_{GD} > FC \cdot PB$$

$$C_{GS} = area \cdot CGS \cdot (1 - FC)^{-M-1} \cdot \left(1 - FC \cdot (1 + M) + M \cdot \frac{V_{GS}}{PB}\right)$$

$$C_{GD} = area \cdot CGD \cdot (1 - FC)^{-M-1} \cdot (1 - FC \cdot (1 + M) + M \cdot \frac{V_{GD}}{PB})$$

### Temperature dependent parameters

The following parameters are functions of temperature. T is the operating temperate and TNOM is the nominal (or measured) temperature. T and TNOM can be adjusted in a number of ways.

$$VTO(T) = VTO + VTOTC \cdot (T - TNOM)$$

$$BETA(T) = BETA \cdot 1.01 \cdot BETATCE \cdot (T - TNOM)$$

$$IS(T) = IS \cdot EXP \cdot \left(\left(\frac{T}{TNOM} - 1\right) \cdot \frac{1.11}{N \cdot Vt}\right) \cdot \left(\frac{T}{TNOM}\right)^{\frac{XTI}{N}}$$

$$ISR(T) = ISR \cdot EXP \cdot \left((-1) \cdot \frac{1.11}{NR \cdot Vt}\right) \cdot \left(\frac{T}{TNOM}\right)^{\frac{XTI}{NR}}$$

$$PB(T) = PB \cdot \frac{T}{TNOM} - 3 \cdot Vt \cdot ln\left(\frac{T}{TNOM}\right) - Eg(TNOM) \cdot \frac{T}{TNOM} + Eg(T)$$

$$CGS(T) = CGS \cdot \left(1 + M \cdot \left(0.0004 \cdot (T - TNOM) + \left(1 - \frac{PB(T)}{PB}\right)\right)\right)$$

$$CGD(T) = CGD \cdot \left(1 + M \cdot \left(0.0004 \cdot (T - TNOM) + \left(1 - \frac{PB(T)}{PB}\right)\right)\right)$$

*where*

$$Eg(T) = 1.16 - \frac{0.000702 \cdot T^2}{T + 1108}$$

### Noise equations

The device has thermal noise generators, $\overline{Irs^2}$ and $\overline{Ird^2}$, as a result of the series ohmic resistances, and the shot and flicker noise generators, collectively $\overline{Ipn^2}$, as a result of the PN junction.

Ohmic resistance noise:

$$\overline{Irs^2} = \frac{4kT\Delta f}{\frac{RS}{area}}$$

$$\overline{Ird^2} = \frac{4kT\Delta f}{\frac{RD}{area}}$$

Shot and flicker noise:

$$\overline{Id^2} = \frac{8}{3}kTgm\Delta f + \frac{KF \cdot {I_d}^{AF}}{f}\Delta f$$

$$gm = small\,signal\,transconductance, \frac{dI_d}{dV_{GS}}$$

### References


1. G. Massobrio and P. Antognetti, Semiconductor Device Modeling with SPICE, 2nd edition, McGraw-Hill, 1993.
2. A. Vladimirescu, The SPICE Book, Wiley, 1994.


### Related Information

*Device Temperature Parameters*

# MESFET

MESFET instance declaration syntax:

Zxxx nodeDrain nodeGate nodeSource Model <area> <OFF> <ICVDS=icvds> <ICVGS=icvgs>

MESFET instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| area | Area factor. |
| OFF | [FLAG] Device is initially off. |
| ICVDS | Initial drain-to-source voltage. |
| ICVGS | Initial gate-to-source voltage. |
| TEMP | Instance temperature. |

N-channel MESFET model definition syntax:

`.MODEL` mymodelname `NMF ( <`*Other_Model_Parameters...*`> )`

P-channel MESFET model definition syntax:

`.MODEL` mymodelname `PMF ( <`*Other_Model_Parameters...*`> )`

MESFET model definition parameters:

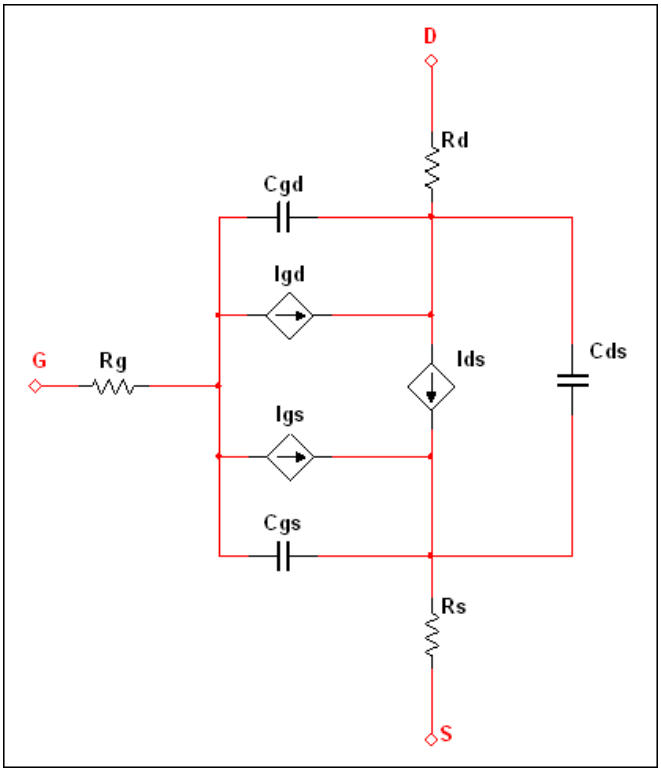
| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| AF | Flicker noise exponent. | — | 1.0 |
| ALPHA | Saturation voltage parameter. | 1/V | 2.0 |
| B | Doping tail parameter. | — | 0.3 |
| BETA | Transconductance parameter. | $A/(V^2)$ | $2.5x10^{-3}$ |
| BETATCE | BETA exponential temperature coefficient. | %/ºC | 0.0 |
| CDS | Drain-to-source junction capacitance. | F | 0.0 |
| CGD | Gate-to-drain junction capacitance. | F | 0.0 |
| CGS | Gate-to-source junction capacitance. | F | 0.0 |
| EG | Bandgap voltage. | eV | 1.11 |
| FC | Forward bias depletion capacitance coefficient. | — | 0.5 |
| IS | Junction saturation current. | A | $1x10^{-14}$ |
| KF | Flicker noise coefficient. | — | 0.0 |
| LAMBDA | Channel length modulation parameter. | 1/V | 0.0 |
| M | Gate junction grading coefficient. | — | 0.5 |
| N | Gate junction emission coefficient. | — | 1.0 |
| PB or VBI | Gate junction potential. | V | 1.0 |
| RD | Drain ohmic resistance. | Ω | 0.0 |
| RG | Gate ohmic resistance. | Ω | 0.0 |
| RS | Source ohmic resistance. | Ω | 0.0 |
| TAU | Conduction current delay time. | sec | 0.0 |
| TNOM | Temperature at which model parameters were measured. | ºC | — |
| TRD1 | RD linear temperature coefficient. | 1/ºC | 0.0 |

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| TRG1 | RG linear temperature coefficient. | 1/ºC | 0.0 |
| TRS1 | RS linear temperature coefficient. | 1/ºC | 0.0 |
| T_ABS | Absolute temperature. | ºC | — |
| T_MEASURED | Temperature at which model parameters were measured. | ºC | — |
| T_REL_GLOBAL | Temperature delta relative to global temperature. | ºC | — |
| T_REL_LOCAL | Temperature delta relative to AKO model temperature. | ºC | — |
| VDELTA | Capacitance transition voltage. | V | 0.2 |
| VMAX | Capacitance limiting voltage. | V | 0.5 |
| VTO or VT0 | Pinch-off voltage. | V | -2.0 |
| VTOTC | VTO temperature coefficient. | V/ºC | 0.0 |
| XTI | IS temperature coefficient. | — | 0.0 |

**Example**

Z1 d g s myGaSFET
.model myGaSFET NMF(vto=1.3)

**Large signal model**




All transistor node voltage references are with respect to the internal nodes in the following equations (that is, the ohmic resistance pin that is connect the inside of the structure.)

**Static equations**

Drain-source current:

$$\textbf{\textit{Linear Region}}: 0 < V_{DS} < \frac{3}{ALPHA} \; and \; V_{GS} > VT0$$

$$I_{DS} = \frac{BETA \cdot (V_{GS} - VT0)^2}{1 + B \cdot (V_{GS} - VT0)} \cdot \left(1 - \left(\frac{ALPHA \cdot V_{DS}}{3}\right)^3\right) \cdot (1 + LAMBDA \cdot V_{DS})$$

$$\textbf{\textit{Saturation Region}}: V_{DS} > \frac{3}{ALPHA} \; and \; V_{GS} > VT0$$

$$I_{DS} = \frac{BETA \cdot (V_{GS} - VT0)^2}{1 + B \cdot (V_{GS} - VT0)} \cdot (1 + LAMBDA \cdot V_{DS})$$

$$\textbf{\textit{Cut} - \textit{off region}}: V_{GS} < VT0$$

$$I_{DS} = 0$$

Gate currents:

$$I_{GS} = IS \cdot \left(\exp\left(\frac{V_{GS}}{N \cdot V_T}\right) - 1\right)$$

$$I_{GD} = IS \cdot \left(\exp\left(\frac{V_{GD}}{N \cdot V_T}\right) - 1\right)$$

*where*

$$V_{t=} \frac{K \cdot T}{q}, \textit{where } K \textit{ is Boltzmann's constant and } q \textit{ is electron charge}$$

### Capacitances

$$C_{DS} = area \cdot CDS$$

$$C_{GS,GD} = area \cdot \frac{CGS}{4 \cdot \sqrt{1 - \frac{Vn}{PB}}} \cdot \left[1 + \frac{Ve - VT0}{\sqrt{(Ve - VT0)^2 + VDELTA^2}}\right] \cdot \left[1 \pm \frac{V_{GS} - V_{GD}}{\sqrt{(V_{GS} - V_{GD})^2 + \left(\frac{1}{ALPHA}\right)^2}}\right]$$

$$+ \frac{CGD}{2} \cdot \left[1 \mp \frac{V_{GS} - V_{GD}}{\sqrt{(V_{GS} - V_{GD})^2 + \left(\frac{1}{ALPHA}\right)^2}}\right]$$

*where*

$$Ve = \frac{1}{2} \cdot \left[V_{GS} + V_{GD} + \sqrt{(V_{GS} - V_{GD})^2 + \left(\frac{1}{ALPHA}\right)^2}\right]$$

*if* $\frac{1}{2} \cdot \left[Ve + VT0 + \sqrt{(Ve - VT0)^2 + VDELTA^2}\right] < VMAX$

$$Vn = \frac{1}{2} \cdot \left[Ve + VT0 + \sqrt{(Ve - VT0)^2 + VDELTA^2}\right]$$

*else*

$$Vn = VMAX$$

In the above equations, the upper sign (of $\mp$ *and* $\pm$) holds for $C_{GS}$ and the lower sign holds for $C_{DS}$.

### Temperature dependent parameters

The following parameters are functions of temperature. T is the operating temperate and TNOM is the nominal (or measured temperature). T and TNOM can be adjusted in a number of ways.

$$VTO(T) = VTO + VTOTC \cdot (T - TNOM)$$

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

$$IS(T) = IS \cdot EXP\left(\left(\frac{T}{TNOM} - 1\right) \cdot \frac{EG}{N \cdot Vt}\right) \cdot \frac{T}{TNOM}^{\frac{XTI}{N}}$$

$$RG(T) = RG \cdot (1 + TRG1 \cdot (T - TNOM))$$

$$RD(T) = RD \cdot (1 + TRD1 \cdot (T - TNOM))$$

$$RS(T) = RS \cdot (1 + TRS1 \cdot (T - TNOM))$$

$$PB(T) = PB \cdot \frac{T}{TNOM} - 3 \cdot Vt \cdot ln(\frac{T}{TNOM}) - Eg(TNOM) \cdot \frac{T}{TNOM} + Eg(T)$$

$$CGS(T) = CGS \cdot \left(1 + M \cdot \left(0.0004 \cdot (T - TNOM) + \left(1 - \frac{PB(T)}{PB}\right)\right)\right)$$

$$CGD(T) = CGD \cdot \left(1 + M \cdot \left(0.0004 \cdot (T - TNOM) + \left(1 - \frac{PB(T)}{PB}\right)\right)\right)$$

*where*

$V_{t=}\frac{K \cdot T}{q}$, *where K is Boltzmann'sconstant and q is electron charge*

$$Eg(T) = 1.16 - \frac{0.000702 \cdot T^2}{T + 1108}$$

### Noise equations

The device has thermal noise generators, $\overline{Irs^2}$ , $\overline{Ird^2}$ and $\overline{Irg^2}$ , as a result of the series ohmic resistances, and the shot and flicker noise generators, collectively $\overline{Id^2}$ , as a result of the channel.

Ohmic resistance noise:

$$\overline{Irs^2} = \frac{4kT\Delta f}{\frac{RS}{area}}$$

$$\overline{Ird^2} = \frac{4kT\Delta f}{\frac{RD}{area}}$$

$$\overline{Irg^2} = \frac{4kT\Delta f}{RG}$$

Shot and flicker noise:

$$\overline{Id^2} = \frac{8}{3}kTgm\Delta f + \frac{KF \cdot {I_d}^{AF}}{f}\Delta f$$

The first term is shot noise and the second term is flicker noise.

$gm = small\ signal\ transconductance, \frac{dI_d}{dV_{GS}}$

### References


1. G. Massobrio and P. Antognetti, Semiconductor Device Modeling with SPICE, 2nd edition, McGraw-Hill, 1993.
2. A. Vladimirescu, The SPICE Book, Wiley, 1994.


### Related Information

*Device Temperature Parameters*

# Controlled Switches



## Voltage-Controlled Switch with Hysteresis

Voltage-Controlled switch instance declaration syntax:

Sxxx node_n+ node_n- nodeNC+ nodeNC- Model <ON/OFF>

Voltage-Controlled Switch instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| ON | [FLAG] Switch initially closed. |
| OFF | [FLAG] Switch initially open. |

Voltage-Controlled Switch model definition syntax:

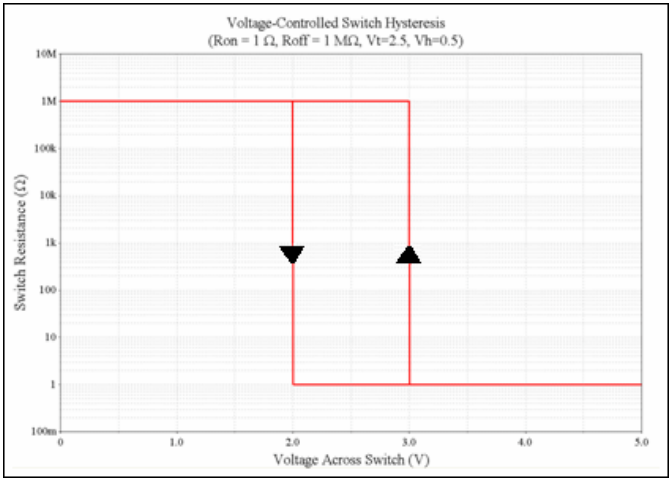
`.MODEL` mymodelname `SW ( <`*Other_Model_Parameters*`...> )`

Voltage-Controlled Switch model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| ROFF | Resistance when open. | Ω | 1/GMIN |
| RON | Resistance when closed. | Ω | 1.0 |
| VH | Hysteresis voltage. | V | 0.0 |
| VT | Threshold voltage. | V | 0.0 |

### Description

As illustrated below the voltage controlled switch changes abruptly between the ON and OFF states. Due to the hysteresis of the switch model the ON->OFF and OFF->ON transitions occur at VT+VH and VT-VH, respectively.




Note that, unless you are specifically interested in the hysteresis characteristic, use of the Smooth Transition switch is recommended because of its superior convergence property.

**Example**
S1 1 0 2 0 mySwitch
.model mySwitch SW(Ron=1 Roff=1meg Vt=2.5 vh=0.5)

## Voltage-Controlled Switch with Smooth Transition

Smooth transition Voltage-Controlled switch instance declaration syntax:

Sxxx node_n+ node_n- nodeNC+ nodeNC- Model

Sxxx node_n+ node_n- (nodeNC+, nodeNC-) Model

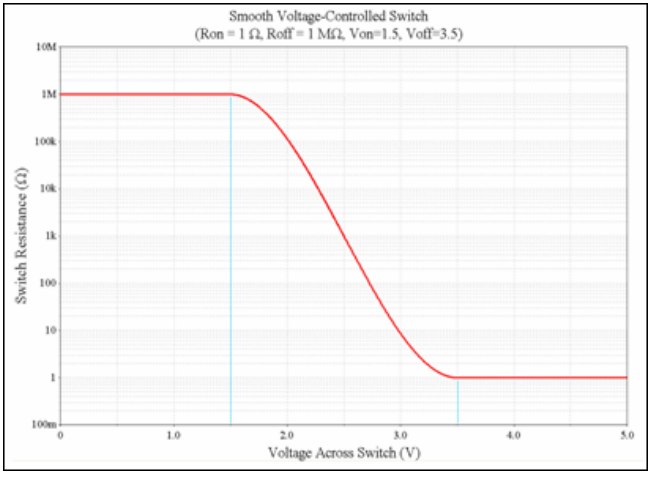
Smooth transition Voltage-Controlled Switch model definition syntax:

.MODEL mymodelname VSWITCH ( <*Other_Model_Parameters*...> )

Smooth transition Voltage-Controlled Switch model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| ROFF | Resistance when open. | Ω | $1X10^{12}$ |
| RON | Resistance when closed. | Ω | 1.0 |
| VOFF | Control OFF value. | V | 0.0 |
| VON | Control ON value. | V | 1.0 |

**Description**
The switching characteristic of the smooth transition voltage-controlled switch are illustrated below. Since this device does not exhibit hysteresis the transition between the ON and OFF states follows the same characteristic curve.



The smooth transition band, with its continuous derivative at all points, aides in finding the solution to a circuit in which this device is used.

### Example

S1 1 0 2 0 mySwitch
.model mySwitch vswitch(Ron=1 Roff=1meg von=1.5 voff=3.5)

## Current-Controlled Switch with Hysteresis

Current-Controlled Switch instance declaration syntax:

Wxxx node_n+ node_n- Vname Model <ON/OFF>

Current-Controlled Switch instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| Vname | Name of controlling voltage source. |
| ON | Switch initially closed. |
| OFF | Switch initially open. |

Current-Controlled switch model definition syntax:

.MODEL mymodelname CSW ( <*Other_Model_Parameters...*> )

Current-Controlled Switch model (CSW) line parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| IH | Hysteresis current. | A | 0.0 |
| IT | Threshold current. | A | 0.0 |
| ROFF | Resistance when open. | Ω | 1/GMIN |
| RON | Resistance when closed. | Ω | 1.0 |

### Description

The Current-Controlled switch follows a similar hysteresis curve as the Voltage-Controlled switch with Hysteresis.

Note that, unless you are specifically interested in the hysteresis characteristic, use of the Smooth Transition switch is recommended because of its superior convergence property.

### Example

S1 1 0 Vcontrol mySwitch
.model mySwitch CSW(Ron=0.1 Roff=1meg it=2 ih=0.1)

### Related Information

*Voltage-Controlled Switch with Hysteresis*

## Current-Controlled Switch with Smooth Transition

Current-Controlled Switch with Smooth Transition instance declaration syntax:

```
Wxxx node_n+ node_n- Vname Model <ON/OFF>
```

Current-Controlled Switch with Smooth Transition instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| Vname | Name of controlling Voltage Source. |

Current-Controlled switch model definition syntax:

```
.MODEL mymodelname ISWITCH ( <Other_Model_Parameters…> )
```

Current-Controlled Switch model with Smooth Transition (CSW) line parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| ROFF | Resistance when open. | Ω | $1 \times 10^{12}$ |
| RON | Resistance when closed. | Ω | 1.0 |
| IOFF | Control OFF value. | V | 0.0 |
| ION | Control ON value. | V | 1.0 |

### Description

The Current-Controlled switch follows a similar hysteresis curve as the Voltage-Controlled switch.

### Example

```
S1 1 0 Vcontrol mySwitch
.model mySwitch ISWITCH(Ron=0.1 Roff=1meg ION=200m IOFF=0)
```

### Related Information

*Voltage-Controlled Switch with Smooth Transition*

# BJT

BJT device instance declaration syntax:

Qxxx nodeCollector nodeBase nodeEmmiter <nodeBody> Model <area> <OFF>
<IC=VBE0, VCE0>

BJT instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| area | Area factor. |
| OFF | [FLAG] Device is initially off. |
| ICVBE | Initial base-emitter voltage. |
| ICVCE | Initial collector-emitter voltage. |
| IC | Initial voltages. This is a two element vector alternate way of specifying ICVBE, ICVCE. |
| SENS_AREA | [FLAG] Flag to request sensitivity with respect to area. |
| TEMP | Instance temperature. |

BJT device NPN model definition syntax:

.MODEL mymodelname NPN <LEVEL=level> (<*Other_Model_Parameters*...> )

BJT device PNP model definition syntax:

.MODEL mymodelname PNP <LEVEL=level> (<*Other_Model_Parameters*...> )

BJT device LPNP model definition syntax:

.MODEL mymodelname LPNP (<*Other_Model_Parameters*...> )

The LEVEL parameter is used to select the appropriate BJT simulation model. Multisim provides two different BJT models, which are described below:

| Level Value | Description |
|---|---|
| 1 or BJT | Gummel-Poon model (DEFAULT model). |
| 4 or VBIC | VBIC model without self heating (Version 1.2). |

Depending on the Level value, different parameters for both instance declarations and model definitions are available.

The Gummel-Poon BJT model parameters are:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| AF | Flicker noise exponent. | — | 1.0 |
| BF | Ideal maximum forward beta. | — | 100.0 |
| BR | Ideal maximum reverse beta. | — | 1.0 |
| CJC | Base-collector zero bias depletion capacitance. | F | 0.0 |
| CJE | Base-emitter zero bias depletion capacitance. | F | 0.0 |
| CJS or CCS | Collector-substrate zero bias depletion capacitance. | F | 0.0 |
| CN | Quasi-saturation temperature coefficient for hole mobility. | — | 2.42 NPN<br>2.20 PNP |
| D | Quasi-saturation temperature coefficient for scattering-limited hole carrier velocity. | — | 0.87 NPN<br>0.52 PNP |
| EG | Bandgap voltage. | eV | 1.11 |
| FC | Forward bias depletion capacitance coefficient. | — | 0.5 |
| GAMMA | Epitaxial region doping factor. | — | $1 \times 10^{-11}$ |
| IKF or IK | Corner for forward beta high current roll-off. | A | infinite |
| IKR | Corner for reverse beta high current roll-off. | A | infinite |
| IRB | Current at which base resistance is (RB+RBM)/2. | A | infinite |
| IS | Transport saturation current. | A | $1 \times 10^{-16}$ |
| ISC | Base-collector leakage saturation current. | A | 0.0 |
| ISE | Base-emitter leakage saturation current. | A | 0.0 |
| ISS | Substrate junction saturation current. | A | 0.0 |
| ITF | High current dependence of TF. | A | 0.0 |
| KF | Flicker noise coefficient. | — | 0.0 |
| MJC or MC | Base-collector junction grading coefficient. | — | 0.33 |
| MJE or ME | Base-emitter junction grading coefficient. | — | 0.33 |
| MJS or MS | Substrate junction grading coefficient. | — | 0.0 |
| NC | Base-collector leakage emission coefficient. | — | 2.0 |
| NE | Base-emitter leakage emission coefficient. | — | 1.5 |

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| NF | Forward current emission coefficient. | — | 1.0 |
| NK | High-current roll-off coefficient. | — | 0.5 |
| NR | Reverse current emission coefficient. | — | 1.0 |
| NS | Substrate junction emission coefficient. | — | 1.0 |
| PTF | Excess phase at 1/(2πTF) Hz. | º | 0.0 |
| QCO | Epitaxial region charge factor. | C | 0.0 |
| QUASIMOD | Quasi-saturation model flag for GAMMA, RCO, and VO temperature dependence:<br>= 1 perform temperature adjustment<br>= 0 don't perform temperature adjustment | — | 0 |
| RB | Zero bias base resistance. | Ω | 0.0 |
| RBM | Minimum base resistance. | Ω | RB |
| RC | Collector resistance. | Ω | 0.0 |
| RCO | Epitaxial region resistance. | Ω | 0.0 |
| RE | Emitter resistance. | Ω | 0.0 |
| TF | Ideal forward transient time. | s | 0.0 |
| TNOM | Temperature at which model parameters were measured. | ºC | — |
| TR | Ideal reverse transit time. | s | 0.0 |
| TRB1 | RB linear temperature coefficient. | 1/ºC | 0.0 |
| TRB2 | RB quadratic temperature coefficient. | $1/(ºC)^2$ | 0.0 |
| TRC1 | RC linear temperature coefficient. | 1/ºC | 0.0 |
| TRC2 | RC quadratic temperature coefficient. | $1/(ºC)^2$ | 0.0 |
| TRE1 | RE linear temperature coefficient. | 1/ºC | 0.0 |
| TRE2 | RE quadratic temperature coefficient. | $1/(ºC)^2$ | 0.0 |
| TRM1 | RBM linear temperature coefficient. | 1/ºC | 0.0 |
| TRM2 | RBM quadratic temperature coefficient. | $1/(ºC)^2$ | 0.0 |
| T_ABS | Absolute temperature. | ºC | — |
| T_MEASURED | Temperature at which model parameters were measured. | ºC | — |

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| T_REL_GLOBAL | Temperature delta relative to global temperature. | ºC | — |
| T_REL_LOCAL | Temperature delta relative to AKO model temperature. | ºC | — |
| VAF or VA | Forward Early voltage. | V | infinite |
| VAR or VB | Reverse Early voltage. | V | infinite |
| VG | Quasi-saturation extrapolated bandgap voltage at 0K. | V | 1.206 |
| VJC or PC | Base-collector built in potential. | V | 0.75 |
| VJE or PE | Base-emitter built in potential. | V | 0.75 |
| VJS or PS | Substrate junction built in potential. | V | 0.75 |
| VO | Carrier mobility knee voltage. | V | 10.0 |
| VTF | Voltage giving VBC dependence of TF. | V | infinite |
| XCJC | Fraction of base-collector capacitance connected to internal base. | — | 1.0 |
| XCJC2 | Fraction of base-collector capacitance connected to internal base. | — | 1.0 |
| XCJS | Fraction of substrate-collector capacitance connected to internal collector. | — | 1.0 |
| XTB | Forward and reverse beta temperature exponent. | — | 0.0 |
| XTF | Coefficient for bias dependence of TF. | — | 0.0 |
| XTI | IS temperature effect exponent. | — | 3.0 |

The VBIC BJT model parameters are:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| AFN | B-E Flicker Noise Exponent. | — | 1.0 |
| AJC | B-C capacitance smoothing factor. | — | -0.5 |
| AJE | B-E capacitance smoothing factor. | — | -0.5 |
| AJS | S-C capacitance smoothing factor. | — | -0.5 |
| ART | Smoothing parameter for reach-through. | — | 0.1 |
| AVC1 | B-C weak avalanche parameter 1. | 1/V | 0.0 |

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| AVC2 | B-C weak avalanche parameter 2. | 1/V | 0.0 |
| BFN | B-E Flicker Noise 1/f dependence. | — | 1.0 |
| CBCO | Extrinsic B-C overlap capacitance. | F | 0.0 |
| CBEO | Extrinsic B-E overlap capacitance. | F | 0.0 |
| CCSO | Fixed C-S capacitance. | F | 0.0 |
| CJC | Zero bias B-C depletion capacitance. | F | 0.0 |
| CJCP | Zero bias S-C capacitance. | F | 0.0 |
| CJE | Zero bias B-E depletion capacitance. | F | 0.0 |
| CJEP | B-C extrinsic zero bias capacitance. | F | 0.0 |
| CTH | Thermal capacitance. | J/K | 0.0 |
| DEAR | Delta activation energy for ISRR. | — | 0.0 |
| DTEMP | Local Temperature difference. | ºC | 0.0 |
| EA | Activation energy for IS. | eV | 1.12 |
| EAIC | Activation energy for IBCI/IBEIP. | eV | 1.12 |
| EAIE | Activation energy for IBEI. | eV | 1.12 |
| EAIS | Activation energy for IBCIP. | eV | 1.12 |
| EANC | Activation energy for IBCN/IBENP. | eV | 1.12 |
| EANE | Activation energy for IBEN. | eV | 1.12 |
| EANS | Activation energy for IBCNP. | ev | 1.12 |
| EAP | Exitivation energy for ISP. | eV | 1.12 |
| EBBE | exp(-VBBE/(NBBE*Vtv)). | eV | 0.0 |
| FC | Fwd bias depletion capacitance limit. | — | 0.9 |
| GAMM | Epi doping parameter. | — | 0.0 |
| HRCF | High current RC factor. | — | 1.0 |
| IBBE | B-E breakdown current. | A | $1 \times 10^{-6}$ |
| IBCI | Ideal B-C saturation current. | A | $1 \times 10^{-6}$ |
| IBCIP | Ideal parasitic B-C saturation current. | A | 0.0 |
| IBCN | Non-ideal B-C saturation current. | A | 0.0 |
| IBCNP | Nonideal parasitic B-C saturation current. | A | 0.0 |

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| IBEI | Ideal B-E saturation current. | A | $1 \times 10^{-18}$ |
| IBEIP | Ideal parasitic B-E saturation current. | A | 0.0 |
| IBEN | Non-ideal B-E saturation current. | A | 0.0 |
| IBENP | Non-ideal parasitic B-E saturation current. | A | 0.0 |
| IKF | Forward knee current. | A | 0.0 |
| IKP | Parasitic knee current. | A | 0.0 |
| IKR | Reverse knee current. | A | 0.0 |
| IS | Transport saturation current. | A | $1 \times 10^{-16}$ |
| ISP | Parasitic transport saturation current. | A | 0.0 |
| ISRR | Separate IS for fwd and rev. | A | 1.0 |
| ITF | High current dependence of TF. | A | 0.0 |
| KFN | B-E Flicker Noise Coefficient. | — | 0.0 |
| MC | B-C junction grading coefficient. | — | 0.33 |
| ME | B-E junction grading coefficient. | — | 0.33 |
| MS | S-C junction grading coefficient. | — | 0.33 |
| NBBE | B-E breakdown emission coefficient. | — | 1.0 |
| NCI | Ideal B-C emission coefficient. | — | 1.0 |
| NCIP | Ideal parasitic B-C emission coefficient. | — | 1.0 |
| NCN | Non-ideal B-C emission coefficient. | — | 2.0 |
| NCNP | Nonideal parasitic B-C emission coefficient. | — | 2.0 |
| NIE | Ideal B-E emission coefficient. | — | 1.0 |
| NEN | Non-ideal B-E emission coefficient. | — | 2.0 |
| NF | Forward emission coefficient. | — | 1.0 |
| NFP | Parasitic fwd emission coefficient. | — | 1.0 |
| NKF | High current beta rolloff. | — | 0.5 |
| NR | Reverse emission coefficient. | — | 1.0 |
| PC | B-C built in potential. | V | 0.75 |
| PE | `B-E built in potential`. | V | 0.75 |
| PS | S-C junction built in potential. | V | 0.75 |

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| QBM | Select SGP qb formulation. | — | 0.0 |
| QCO | Epi charge parameter. | C | 0.0 |
| QTF | Variation of TF with base-width modulation. | — | 0.0 |
| RBI | Intrinsic base resistance. | Ω | 0.1 |
| RBP | Parasitic base resistance. | Ω | 0.1 |
| RBX | Extrinsic base resistance. | Ω | 0.1 |
| RCI | Intrinsic collector resistance. | Ω | 0.1 |
| RCX | Extrinsic collector resistance. | Ω | 0.1 |
| RE | Intrinsic emitter resistance. | Ω | 0.1 |
| RS | Intrinsic substrate resistance. | Ω | 0.1 |
| RTH | Thermal resistance. | K/W | 0.0 |
| TAVC | Temperature exponent of AVC2. | 1/K | 0.0 |
| TD | Forward excess-phase delay time. | s | 0.0 |
| TF | Ideal forward transit time. | s | 0.0 |
| TNBBE | Temperature coefficient of NBBE. | — | 0.0 |
| TNF | Temperature exponent of NF. | 1/K | 0.0 |
| TNOM | Parameter measurement temperature. | ºC | 27.0 |
| TR | Ideal reverse transit time. | s | 0.0 |
| TVBBE1 | Linear temperature coefficient of VBBE. | — | 0.0 |
| TVBBE2 | Quadratic temperature coefficient of VBBE. | — | 0.0 |
| VBBE | B-E breakdown voltage. | V | 0.0 |
| VEF | Forward Early voltage. | V | 0.0 |
| VER | Reverse Early voltage. | V | 0.0 |
| VERS | Revision Version. | — | 1.2 |
| VO | Epi drift saturation voltage. | V | 0.0 |
| VREF | Reference Version. | — | 0.0 |
| VRT | Punch-through voltage of internal B-C junction. | V | 0.0 |
| VTF | Voltage giving VBC dependence of TF. | V | 0.0 |

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| WBE | Portion of IBEI from Vbei, 1-WBE from Vbex. | — | 1.0 |
| WSP | Portion of ICCP. | — | 1.0 |
| XII | Temperature exponent of IBEI, IBCI, IBEIP, IBCIP. | — | 3.0 |
| XIKF | Temperature exponent of IKF. | — | 0.0 |
| XIN | Temperature exponent of IBEN, IBCN, IBENP, IBCNP. | — | 3.0 |
| XIS | Temperature exponent of IS. | — | 3.0 |
| XISR | Temperature exponent of ISR. | — | 0.0 |
| XRB | Temperature exponent of RB. | — | 0.0 |
| XRBI | Temperature exponent of RBI. | — | 0.0 |
| XRBP | Temperature exponent of RBP. | — | 0.0 |
| XRBX | Temperature exponent of RBX. | — | 0.0 |
| XRC | Temperature exponent of RC. | — | 0.0 |
| XRCI | Temperature exponent of RCI. | — | 0.0 |
| XRCX | Temperature exponent of RCX. | — | 0.0. |
| XRE | Temperature exponent of RE. | — | 0.0 |
| XRS | Temperature exponent of RS. | — | 0.0 |
| XTF | Coefficient for bias dependence of TF. | — | 0.0 |
| XVO | Temperature exponent of VO. | — | 0.0 |

SPICE implementation of VBIC was developed by Dr. Dietmar Warning, with contributions from authors of the NGSPICE project.

### Additional Notes

The level parameter (level) must be a numerical constant—it may not contain parameters or expressions.

### Examples

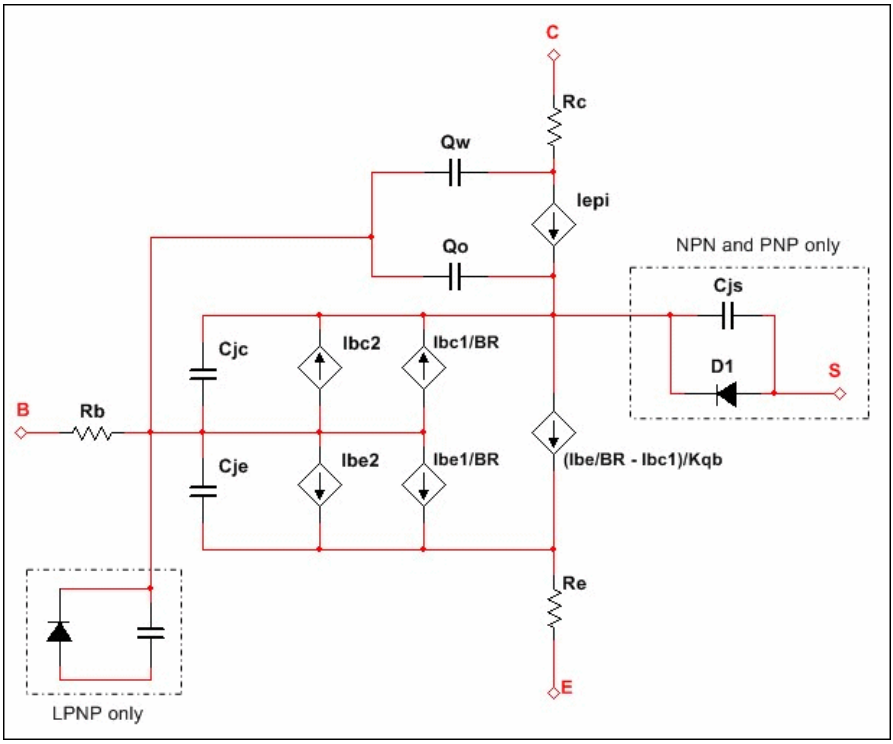
```
Q1 e b c 0 myBJT
.model myBJT NPN(vto=1.3)

Q2 e b c 0 myBJT2
.model myBJT2 PNP(LEVEL=4)
```

# BJT Equations

This section describes the equations governing the operation of BJT Level 1 (model statement Level parameter set to '1' or is left undefined).

### Large signal model




In the following sections, all transistor node voltage references are with respect to the internal nodes (that is, the ohmic resistance pin that is connect the inside of the structure.)

### Static model

Base current: $Ib = area \cdot (Ibe1/BF + Ibe2 + Ibc1/BR + Ibc2)$

Collector current: $Ic = area \cdot (\frac{Ibe1}{Kqb} - \frac{Ibc1}{Kqb} - \frac{Ibc1}{BR} - Ibc2)$

Forward diffusion current: $Ibe1 = IS \times \left(EXP(\frac{Vbc}{NCVt}) - 1\right)$

Non-ideal base-emitter current: $Ibe2 = ISE \cdot \left( EXP(\frac{Vbe}{NE \cdot Vt}) - 1 \right)$

Reverse diffusion current: $Ibc1 = IS \cdot (EXP(\frac{Vbc}{NR \cdot Vt}) - 1)$

Non-ideal base-collector current: $Ibc2 = ISC \cdot (EXP(\frac{Vbc}{NC \cdot Vt}) - 1)$

Base charge factor: $Kqb = Kq1 \cdot \frac{1 + (1 + 4 \cdot Kq2)^{NK}}{2}$

$$Kq1 = \frac{1}{1 - \frac{Vbc}{VAF} - \frac{Vbe}{VAR}}$$

$$Kq2 = \frac{Ibe1}{IKF} + \frac{Ibc1}{IKR}$$

Substrate current: $Is = area \cdot ISS \cdot \left( EXP(\frac{Vjs}{NS \cdot Vt}) - 1 \right)$

**Actual base parasitic resistance**

$$Rb = \begin{cases} \frac{RBM + \frac{RB - RBM}{Kqb}}{area}, & IRB = \infty \ (unspecified) \\ \frac{RBM + 3 \cdot (RB - RBM) \cdot \frac{\tan x - x}{x \cdot tan(x) 2 \cdot}}{area}, & IRB > 0 \end{cases}$$

$$where: \quad x = \frac{\left( 1 + \left( \frac{144}{\pi^2} \right) \cdot \frac{Ib}{area \cdot IRB} \right)^{1/2} - 1}{\left( \frac{24}{\pi^2} \right) \cdot \left( \frac{Ib}{area \cdot IRB} \right)^{1/2}}$$

**Capacitances**

Base-emitter capacitance:

Base-emitter capacitance: $Cbe = Ctbe + area \cdot Cjbe$

Transit time capacitance: $Ctbe = tf \cdot Gbe$

Effective TF: $tf = TF \cdot \left( 1 + XTF \cdot \left( \frac{Ibe1}{Ibe1 + area \cdot ITF} \right) 2 \cdot EXP(\frac{Vbc}{1.44 \cdot VTF}) \right)$

DC base-emitter conductance: $Gbe = (dIbe)/(dVb)$

$$Ibe = Ibe1 + Ibe2$$

$$Cjbe = \begin{cases} Cje \cdot \left(1 - \frac{Vbe}{Vje}\right)^{-Mje}, & Vbe \leq FC \cdot Vje \\ Cje \cdot (1 - FC)^{-(1+Mje)} \cdot \left(1 - FC \cdot (1 + Mje) + Mje \cdot \frac{Vbe}{Vje}\right), & Vbe > FC \cdot Vje \end{cases}$$

Base-collector capacitance:

Base-collector capacitance: $Cbc = Ctbc + area \cdot XCJC \cdot Cjbc$

Transit time capacitance: $Ctbc = TR \cdot Gbc$

DC base-collector conductance: $Gbc = \frac{dIbc}{dVbc}$

$$Cjbc = \begin{cases} Cjc \cdot \left(1 - \frac{Vbc}{Vjc}\right)^{-Mjc}, & Vbc < FC \cdot Vjc \\ Cjc \cdot (1 - FC)^{-(1+Mjc)} \cdot \left(1\ FC \cdot (1 + Mjc) + Mjv \cdot \frac{Vbc}{Vjc}\right), & Vbc > FC \cdot Vjc \end{cases}$$

Extrinsic-base to intrinsic-collector capacitance:

Extrinsic-base to intrinsic-collector capacitance: $Cbx = area \cdot (1 - Xcjc) \cdot Cjbx$

$$Cjbx = \begin{cases} Cjs \cdot \left(1 - \frac{Vbx}{Vjc}\right)^{-Mjc}, & Vbx < FC \cdot Vjc \\ Cjc \cdot (1 - FC) - (1 + Mjc) \cdot \left(1 - FC \cdot (1 + Mjc) + Mjc \cdot \frac{Vbx}{Vjc}\right), & Vbx > FC \cdot Vjc \end{cases}$$

Substrate junction capacitance:

Substrate junction capacitance: $Cjs = area \cdot Cjjs$

$$Cjjs = \begin{cases} Cjs \cdot (1 - \frac{Vjs}{Vjs})^{-Mjs}\ (assumes\ FC = 0), & Vjs < 0 \\ Cjjs = Cjs \cdot \left(1 + Mjs \cdot \frac{Vjs}{Vjs}\right), & Vjs > 0 \end{cases}$$

Quasi-saturation effect:

$$Iepi = area \cdot (VO \cdot (Vt \cdot (K(Vbc) - K(Vbn) - ln((1 + K(Vbc))/(1 + K(Vbn)))) + Vbc - Vbn))/RCO \cdot (|Vbc - Vbn| + VO)$$

$$Qo = area \cdot QCO \cdot (K(Vbc) - 1 - GAMMA/2)$$

$$Qw = area \cdot QCO \cdot \left(K(Vbn) - 1 - \frac{GAMMA}{2}\right) \quad \text{, where } K(v) = \left(1 + GAMMA \cdot EXP\left(\frac{v}{Vt}\right)\right)^{1/2}$$

Temperature effect:

$$IS(T) = IS \cdot EXP\left(\left(\frac{T}{Tnom} - 1\right) \cdot \frac{EG}{N \cdot Vt}\right) \cdot \left(\frac{T}{Tnom}\right)^{\frac{XTI}{N}} \quad \text{, where N = 1}$$

$$ISE(T) = \left(\frac{ISE}{\left(\frac{T}{Tnom}\right)^{XTB}}\right) \cdot EXP\left(\left(\frac{T}{Tnom} - 1\right) \cdot \frac{EG}{NE \cdot Vt}\right) \cdot \left(\frac{T}{Tnom}\right)^{\frac{XTI}{NE}}$$

$$ISC(T) = \left(\frac{ISC}{\left(\frac{T}{Tnom}\right)^{XTB}}\right) \cdot EXP\left(\left(\frac{T}{Tnom} - 1\right) \cdot \frac{EG}{NC \cdot Vt}\right) \cdot \left(\frac{T}{Tnom}\right)^{XTI/NC}$$

$$ISS(T) = \left(\frac{ISS}{\left(\frac{T}{Tnom}\right)^{XTB}}\right) \cdot EXP\left(\left(\frac{T}{Tnom} - 1\right) \cdot \frac{EG}{NS \cdot Vt}\right) \cdot \left(\frac{T}{Tnom}\right)^{XTI/NC}$$

$$BF(T) = BF \cdot \left(\frac{T}{Tnom}\right)^{XTB}$$

$$BR(T) = BR \cdot \left(\frac{T}{Tnom}\right)^{XTB}$$

$$RE(T) = RE \cdot (1 + TRE1 \cdot (T - Tnom) + TRE2 \cdot (T - Tnom)^2)$$

$$RB(T) = RB \cdot (1 + TRE1 \cdot (T - Tnom) + TRE2 \cdot (T - Tnom)^2$$

$$RBM(T) = RBM \cdot (1 + TRM1 \cdot (T - Tnom) + TRM2 \cdot (T - Tnom)^2)$$

$$RC(T) = RC \cdot (1 + TRC1 \cdot (T - Tnom) + TRC2 \cdot (T - Tnom)^2)$$

$$Vje(T) = Vje \cdot \frac{T}{Tnom} - 3 \cdot Vt \cdot ln\left(\frac{T}{Tnom}\right) - Eg(Tnom) \cdot \frac{T}{Tnom} + Eg(T)$$

$$Vjc(T) = Vjc \cdot \frac{T}{Tnom} - 3 \cdot Vt \cdot ln\left(\frac{T}{Tnom}\right) - Eg(Tnom) \cdot \frac{T}{Tnom} + Eg(T)$$

$$Vjs(T) = Vjs \cdot \frac{T}{Tnom} - 3 \cdot Vt \cdot ln\left(\frac{T}{Tnom}\right) - Eg(Tnom) \cdot \frac{T}{Tnom} + Eg(T)$$

where Eg(T) = silicon bandgap energy $= 1.16 - .000702 \cdot \frac{T2}{T+1108}$

$$Cje(T) = Cje \cdot \left(1 + Mje \cdot \left(0.0004 \cdot (T - Tnom) + \left(1 - \frac{Vje(T)}{Vje}\right)\right)\right)$$

$$Cjs(T) = Cjs \cdot \left(1 + Mjs \cdot \left(0.0004 \cdot (T - Tnom) + \left(1 - \frac{Vjs(T)}{Vjs}\right)\right)\right)$$

$$Cjc(T) = Cjc \cdot \left(1 + Mjc \cdot \left(0.0004 \cdot (T - Tnom) + \left(1 - \frac{Vjc(T)}{Vjc}\right)\right)\right)$$

$$GAMMA(T) = GAMMA(Tnom) \cdot \left(\frac{T}{Tnom}\right)^3 \cdot \exp\left(-\frac{qVG}{k} \cdot \left(\frac{1}{T} - \frac{1}{Tnom}\right)\right)$$

$$RCO(T) = RCO(Tnom) \cdot \left(\frac{T}{Tnom}\right)^{CN}$$

$$VO(T) = VO(Tnom) \cdot \left(\frac{T}{Tnom}\right)^{CN-D}$$

Parasitic resistances thermal noise:

$$\text{RC:} \quad Ic^2 = 4 \cdot k \cdot \frac{T}{\left(\frac{RC}{area}\right)}$$

$$\text{RB:} \quad Ib^2 = 4 \cdot k \cdot \frac{T}{RB}$$

$$\text{RE:} \quad Ie^2 = 4 \cdot k \cdot \frac{T}{\frac{RE}{area}}$$

Base and collector currents shot and flicker noise:

$$\text{IB:} \quad Ib^2 = 2 \cdot q \cdot Ib + KF \cdot \frac{Ib^{AF}}{FREQUENCY}$$

$$\text{IC:} \quad Ic^2 = 2 \cdot q \cdot Ic$$

# MOSFET

MOSFET device instance declaration syntax:

Mxxx nodeDrain nodeGate nodeSource nodeBulk Model <<L=>l> <<W=>w> <AD=ad>
<AS=as> <PD=pd> <PS=ps>
+ <NRD=nrd> <NRS=nrs> <M=m> <OFF> <IC=vds <, vgs <, vbs>>>

Mxxx nodeDrain nodeGate nodeSource nodeBulk Model <<L=>l> <<W=>w> <AD=ad> <AS=as> <PD=pd> <PS=ps>
+ <NRD=nrd> <NRS=nrs> <M=m> <OFF> <VDS=vds> <VGS=vgs> <VBS=vbs>

Basic MOSFET instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| 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. |
| M | Device multiplicity factor. |
| OFF | [FLAG] Device is initially off. |
| ICVDS | Initial drain-to-source voltage. |
| ICVGS | Initial gate-to-source voltage. |
| ICVBS | Initial bulk-to-source voltage. |
| IC | Initial voltages. This is a three element vector alternate way of specifying ICVDS, ICVGS, ICVBS. |
| SENS_L | [FLAG] flag to request sensitivity with respect to length. |
| SENS_W | [FLAG] flag to request sensitivity with respect to width. |
| TEMP | Instance temperature. |

MOSFET device NMOS simulation model definition syntax:

.MODEL mymodelname NMOS (<LEVEL=level> <*Other_Model_Parameters*...> )

MOSFET device PMOS simulation model definition syntax:

.MODEL mymodelname PMOS (<LEVEL=level> <*Other_Model_Parameters*...> )

The LEVEL parameter is used to select the appropriate MOSFET simulation model. Multisim provides eight different MOSFET models, which are described below:

| Level Value | Description |
|---|---|
| 1 or MOS1 | Shichman-Hodges model (DEFAULT model). |
| 2 or MOS2 | More complex model than LEVEL 1 based on actual device physics. |
| 3 or MOS3 | Semi-empirical model good for simulating short channel effects. |
| 4 or BSIM1 | BSIM1. |
| 5 or BSIM2 | BSIM2. |
| 6 or MOS6 | N-th power law MOSFET model. |
| 8 or BSIM3 | BSIM3 (version 3.3.0). |
| 10 or B4SOI | BSIMSOI4 (version 4.4). |
| 14 or BSIM | BSIM4 (version 4.8.0). |
| 44 or EKV | EKV (version 2.6) |

Depending on the Level value, different parameters for both instance declarations and model definitions are available.

MOSFET model definition parameters for MOS1 and MOS2:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| AF | Flicker noise exponent. | — | 1.0 |
| CBD | Bulk-to-drain junction capacitance. | F | 0.0 |
| CBS | Bulk-to-source junction capacitance. | F | 0.0 |
| CGBO | Gate-to-bulk overlap capacitance. | F/m | 0.0 |
| CGDO | Gate-to-drain overlap capacitance. | F/m | 0.0 |
| CGSO | Gate-to-source overlap capacitance. | F/m | 0.0 |
| CJ | Zero-bias bulk junction bottom capacitance per area. | $F/(m^2)$ | 0.0 |
| CJSW | Zero-bias bulk junction sidewall capacitance per length. | F/m | 0.0 |
| FC | Forward bias depletion capacitance coefficient. | — | 0.5 |
| GAMMA | Bulk threshold parameter. | $V^{1/2}$ | 0.0 |

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| GDSNOI | Channel shot noise coefficient (use when NLEV=3). | — | 1.0 |
| IS | Bulk junction saturation current. | A | $1 \times 10^{-14}$ |
| JS | Bulk junction saturation current density. | $A/(m^2)$ | 0.0 |
| JSSW | Bulk junction saturation sidewall current per length. | A/m | 0.0 |
| KF | Flicker noise coefficient. | — | 0.0 |
| KP | Transconductance parameter. | $A/(V^2)$ | $2 \times 10^{-5}$ |
| L | Length. | m | DEFW |
| LAMBDA | Channel length modulation. | 1/V | 0.0 |
| LD | Lateral diffusion (length). | m | 0.0 |
| MJ | Bulk junction bottom grading coefficient. | — | 0.5 |
| MJSW | Bulk junction sidewall grading coefficient. | — | 0.33 |
| N | Bulk junction emission coefficient. | — | 1.0 |
| NLEV | Noise equation selector. | — | 2 |
| NSS | Surface state density. | $1/(cm^2)$ | 0.0 |
| NSUB | Substrate doping. | $1/(cm^2)$ | 0.0 |
| PB | Bulk junction potential. | V | 0.8 |
| PBSW | Bulk junction sidewall potential. | V | PB |
| PHI | Surface potential. | V | 0.6 |
| RB | Bulk ohmic resistance. | Ω | 0.0 |
| RD | Drain ohmic resistance. | Ω | 0.0 |
| RDS | Drain-to-source shunt resistance. | Ω | infinite |
| RG | Gate ohmic resistance. | Ω | 0.0 |
| RS | Source ohmic resistance. | Ω | 0.0 |
| RSH | Drain and source diffusion sheet resistance. | Ω/sq. | 0.0 |
| TT | Bulk junction transit time. | s | 0.0 |
| TNOM | Temperature at which model parameters were measured. | ºC | — |
| T_ABS | Absolute temperature. | ºC | — |

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| T_MEASURED | Temperature at which model parameters were measured. | ºC | — |
| T_REL_GLOBAL | Temperature delta relative to global temperature. | ºC | — |
| T_REL_LOCAL | Temperature delta relative to AKO model temperature. | ºC | — |
| TOX | Oxide thickness. | m | Refer to *Additional Notes* |
| TPG | Type of gate material:<br>+1 = opposite to substrate<br>-1 = same as substrate<br>0 = aluminum. | — | 1 |
| U0 | Surface mobility. | $cm^2/Vs$ | 600 |
| VTO | Threshold voltage. | V | 0.0 |
| W | Width. | m | DEFL |
| WD | Lateral diffusion (width). | m | 0.0 |

Additional MOSFET model definition parameters for MOS3:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| AF | Flicker noise exponent. | — | 1.0 |
| ALPHA | Alpha. | — | 0.0 |
| DELTA | Width effect on threshold. | — | 0.0 |
| ETA | Static feedback. | — | 0.0 |
| KAPPA | Saturation field factor. | — | 0.2 |
| KF | Flicker noise coefficient. | — | 0.0 |
| L | Length. | m | DEFW |
| NFS | Fast surface state density. | $1/(cm^2)$ | 0.0 |
| RB | Bulk ohmic resistance. | Ω | 0.0 |
| RG | Gate ohmic resistance. | Ω | 0.0 |
| THETA | Mobility modulation. | 1/V | 0.0 |
| VMAX | Maximum carrier drift velocity. | m/s | 0.0 |

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| W | Width. | m | DEFL |
| XD | Depletion layer width. | — | 0.0 |
| XJ | Metallurgical junction depth. | m | 0.0 |

Additional MOSFET model definition parameters for MOS6:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| GAMMA1 | Bulk threshold parameter 1. | $V^{1/2}$ | 0.0 |
| KC | Saturation current factor. | — | $5 \times 10^{-5}$ |
| KV | Saturation voltage factor. | — | 2.0 |
| LAMBDA0 | Channel length modulation parameter 0. | — | 0.0 |
| LAMBDA1 | Channel length modulation parameter 1. | 1/V | 0.0 |
| NC | Saturation current coefficient. | — | 1.0 |
| NV | Saturation voltage coefficient. | — | 0.5 |
| NVTH | Threshold voltage coefficient. | — | 0.5 |
| PS | Saturation current modification parameter. | — | 0.0 |
| SIGMA | Static feedback effect parameter. | — | 0.0 |

MOSFET model definition for the EKV version 2.6 model:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| AF | Flicker noise exponent. | — | 1.0 |
| BEX | Mobility temperature exponent. | — | -1.5 |
| CBD | Bulk-to-drain PN junction capacitance. | F | 0.0 |
| CBS | Bulk-to-source PN junction capacitance. | F | 0.0 |
| CGBO | Gate-to-bulk overlap capacitance. | F/m | 0.0 |
| CGDO | Gate-to-drain overlap capacitance. | F/m | 0.0 |
| CGSO | Gate-to-source overlap capacitance. | F/m | 0.0 |
| CJ | Bottom PN junction capacitance per area. | $F/m^2$ | 0.0 |
| CJSW | Sidewall PN junction capacitance per length. | F/m | 0.0 |
| COX | Gate oxide capacitance. | $F/(m^2)$ | $7 \times 10^{-4}$ |
| DL | Channel length correction. | m | 0.0 |

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| DW | Channel width correction. | m | 0.0 |
| EO | NEW Mobility reduction coefficient. | V/m | $1 \times 10^{12}$ |
| EKVINT | Interpolation function selector. | — | 0.0 |
| FC | Forward PN junctin capacitance coefficient. | — | 0.5 |
| GAMMA | Body effect parameter. | $V^{1/2}$ | 1.0 |
| IBA | First impact ionization coefficient. | 1/m | 0.0 |
| IBB | Second impact ionization coefficient. | V/m | $3 \times 10^{8}$ |
| IBBT | Temperature coefficient for $I_{BB}$. | 1/K | $9 \times 10^{-4}$ |
| IBN | Saturation voltage factor for impact ionization. | — | 1.0 |
| IS | Bulk PN junction saturation current. | A | $1 \times 10^{-14}$ |
| JS | Bulk PN junction bottom saturation current per area. | $A/(m^2)$ | 0.0 |
| JSW | Bulk PN junction sidewall saturation current per length. | A/m | 0.0 |
| KF | Flicker noise coefficient. | — | 0.0 |
| KP | Transconductance parameter. | $A/(V^2)$ | $5 \times 10^{-5}$ |
| LAMBDA | Depletion length coefficient. | — | 0.5 |
| LETA | Short channel coefficient. | — | 0.1 |
| LK | RSCE characteristic length. | m | $2.9 \times 10^{-7}$ |
| MJ | Bottom PN junction grading coefficient. | — | 0.5 |
| MJSW | Sidewall PN junction grading coefficient. | — | 0.33 |
| N | Emission coefficient. | — | 1.0 |
| NLEVEL | Noise level selector. | — | 1.0 |
| NMOS | N type MOSFET model. | — | — |
| NQS | Non-quasi-static operation switch. | — | 0.0 |
| PB | Bulk PN junction potential. | V | 0.8 |
| PBSW | Bulk sidewall PN junction potential. | V | 0.8 |
| PHI | Bulk Fermi potential. | V | 0.7 |
| PMOS | P type MOSFET model. | — | — |
| Q0 | RSCE excess charge. | $A \cdot s/(m^2)$ | 0.0 |

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| RD | Drain ohmic resistance. | Ω | 0.0 |
| RDC | Drain contact resistance. | Ω | 0.0 |
| RS | Source ohmic resistance. | Ω | 0.0 |
| RSC | Source contact resistance. | Ω | 0.0 |
| RSH | Drain source sheet resistance. | Ω/sq. | 0.0 |
| SATLIM | Ratio defining the saturation limit. | — | 4.0 |
| TCV | Threshold voltage temperature coefficient. | V/K | $1.0e^{-3}$ |
| THETA | Mobility reduction coefficient. | 1/V | 0.0 |
| TNOM | Parameter measurement temperature | ºC | — |
| TR1 | 1st order temperature coefficient. | 1/ºC | 0.0 |
| TR2 | 2nd order temperature coefficient. | $1/(ºC)^2$ | 0.0 |
| TT | Bulk PN junction transit time. | s | 0.0 |
| TYPE | N-channel or P-channel MOSFET. | — | “NMOS” |
| UCEX | Longitudinal critical field temperature coefficient. | — | 0.8 |
| UCRIT | Longitudinal critical field. | V/m | $100 \times 10^6$ |
| VTO | Nominal threshold voltage. | V | 0.5 for NMOS<br>-0.5 for PMOS |
| WETA | Narrow channel effect coefficient. | — | 0.25 |
| XJ | Junction depth. | m | $1x10^{-7}$ |
| XTI | Junction current temperature exponent. | — | 3.0 |

The EKV MOST model used in this software was developed, implemented and tested by the Electronics Laboratory (LEG) of the Swiss Federal Institute of Technology (EPFL).

### Additional Notes

- The level parameter (level) must be a numerical constant - it may not contain parameters or expressions.
- The oxide thickness parameter (TOX) has default value of 1x10-7 for LEVEL 2 and LEVEL 3 models. For the LEVEL 1 the default value is unspecified, and if TOX is not provided the model does not use the process parameters, for example, TOX, NSUB, COX, UO).
- Please consult external documentation at `www-device.eecs.berkeley.edu/bsim` for additional details on BSIM models.

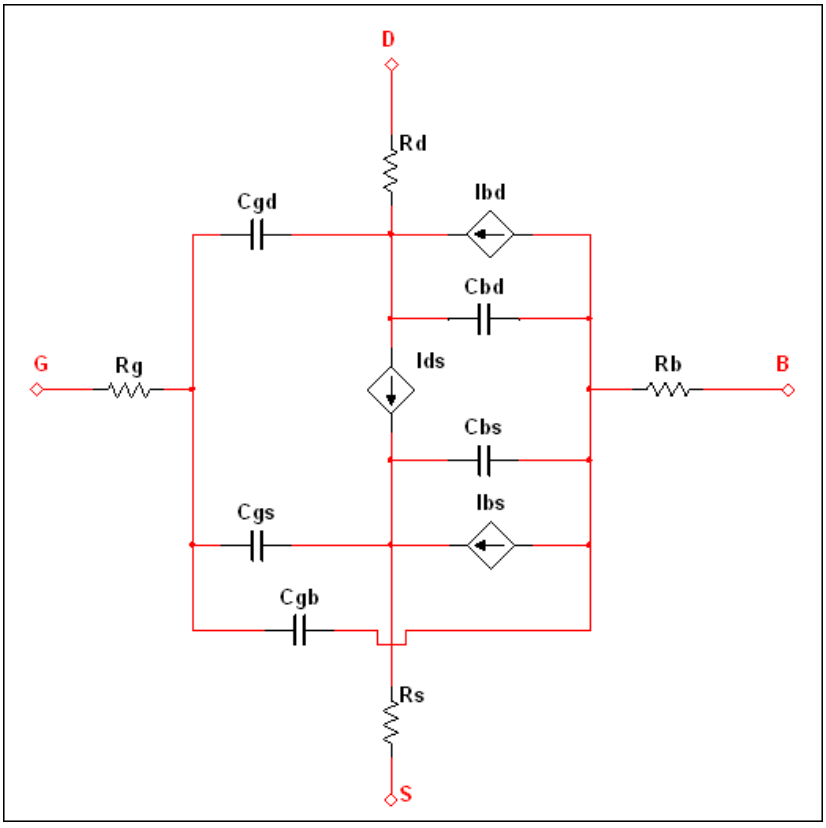
**Example**

M1 d g s s myMOS w=200u l=100u m=3
.model myMOS NMOS(vto=1.3)

## MOSFET Equations

This section describes the equations governing the operation of MOSFET Level 1 (model statement Level parameter set to ‘1’ or is left undefined).

### Large signal model




In the following sections, all transistor node voltage references are with respect to the internal nodes (that is, the ohmic resistance pin that is connect the inside of the structure.)

**Static model**

Drain-source current:

The forward current is the sum of the normal and recombination currents:

$$\textbf{\textit{Linear region}}: V_{GS} > V_{TH} \text{ and } V_{DS} < V_{GS} - V_{TH}$$

$$I_{DS} = KP \cdot \frac{W}{Leff} \cdot \left(V_{GS} - V_{TH} - \frac{V_{DS}}{2}\right) \cdot V_{DS} \cdot (1 + LAMBDA \cdot V_{DS})$$

$$\textbf{\textit{Saturation region}}: V_{GS} > V_{TH} \text{ and } V_{DS} < V_{GS} - V_{TH}$$

$$I_{DS} = \frac{KP}{2} \cdot \frac{W}{Leff} \cdot (V_{GS} - V_{TH})^2 \cdot V_{DS} \cdot (1 + LAMBDA \cdot V_{DS})$$

$$\textbf{\textit{Cut} - \textit{off region}}: \boldsymbol{V_{GS} < V_{TH}}$$

$$I_{DS} = 0$$

*where*

$$V_{TH} = VTO + GAMMA \cdot \left(\sqrt{PHI - V_{BS}} - \sqrt{PHI}\right)$$

$$Leff = L - 2 \cdot LD$$

Substrate currents:

$$I_{BS} = \begin{cases} IS \cdot \left(EXP\left(\frac{V_{BS}}{V_t} - 1\right)\right), & V_{BS} > 0 \\ \frac{ISS}{V_t} \cdot V_{BS}, & V_{BS} \leq 0 \end{cases}$$

$$I_{BD} = \begin{cases} IS \cdot \left(EXP\left(\frac{V_{BD}}{V_t} - 1\right)\right), & V_{BD} > 0 \\ \frac{ISS}{V_t} \cdot V_{BD}, & V_{BD} \leq 0 \end{cases}$$

**Capacitances**

Gate capacitances:

$$\textbf{\textit{Accumulation region}}: V_{GS} < V_{ON} - PHI$$

$$C_{GB} = Cox + CGBO \cdot Leff$$
$$C_{GS} = CGSO \cdot W$$
$$C_{GD} = CGDO \cdot W$$

$$\textbf{\textit{Depletion region}}: V_{ON} - PHI < V_{GS} < V_{ON}$$

$$C_{GB} = Cox \cdot \frac{V_{ON} - V_{GS}}{PHI} + CGBO \cdot Leff$$
$$C_{GS} = \frac{2}{3} \cdot Cox \cdot \left(\frac{V_{ON} - V_{GS}}{PHI} + 1\right) + CGSO \cdot W$$
$$C_{GD} = CGDO \cdot W$$

$$\textbf{\textit{Saturation region}}: V_{ON} < V_{GS} < V_{ON+}\mathrm{V_{DS}}$$

$$C_{GB} = CGBO \cdot Leff$$
$$C_{GS} = \frac{2}{3} \cdot Cox + CGSO \cdot W$$
$$C_{GD} = CGDO \cdot W$$

$$\textbf{\textit{Linear region}}: V_{GS} > V_{ON} + V_{DS}$$

$$C_{GB} = CGBO \cdot Leff$$
$$C_{GS} = Cox \cdot \left(1 - \left(\frac{V_{GS} - V_{DS} - V_{ON}}{2 \cdot (V_{GS} - V_{ON}) - V_{DS}}\right)^2\right) + CGSO \cdot W$$
$$C_{GD} = Cox \cdot \left(1 - \left(\frac{V_{GS} - V_{DS} - V_{ON}}{2 \cdot (V_{GS} - V_{ON}) - V_{DS}}\right)^2\right) + CGDO \cdot W$$

*where*

$$Cox = Cox' \cdot W \cdot Leff$$
$$Cox' = \frac{\varepsilon ox \cdot \varepsilon 0}{TOX} = \frac{3.9 \cdot 8.854e - 12}{TOX}$$

Junction capacitances:

$C_{BD}$ and $C_{BS}$ represent the non-linear bulk-drain and bulk-source capacitances.

*if* $CBD = CBS = 0$ *or unspecified*

$$C_{BS} = AS \cdot CJ \cdot C_{BSJ} + PS \cdot CJSW \cdot C_{BSS} + TT \cdot \frac{dI_{BS}}{dV_{BS}}$$

$$C_{BD} = AS \cdot CJ \cdot C_{BDJ} + PS \cdot CJSW \cdot C_{BDS} + TT \cdot \frac{dI_{BD}}{dV_{BD}}$$

*else*

$$C_{BS} = CBS \cdot C_{BSJ} + PS \cdot CJSW \cdot C_{BSS} + TT \cdot \frac{dI_{BS}}{dV_{BS}}$$

$$C_{BD} = CBD \cdot C_{BDJ} + PS \cdot CJSW \cdot C_{BDS} + TT \cdot \frac{dI_{BD}}{dV_{BD}}$$

*For* $V_{BS}, V_{BD} < FC \cdot PB$:

$$C_{BSJ} = \frac{1}{\left(1 - \frac{V_{BS}}{PB}\right)^{MJ}}$$

$$C_{BSS} = \frac{1}{\left(1 - \frac{V_{BS}}{PB}\right)^{MJSW}}$$

$$C_{BDJ} = \frac{1}{\left(1 - \frac{V_{BD}}{PB}\right)^{MJ}}$$

$$C_{BDS} = \frac{1}{\left(1 - \frac{V_{BD}}{PB}\right)^{MJSW}}$$

For $V_{BS}, V_{BD} > FC \cdot PB$:

$$C_{BSJ} = \frac{\left(1 - FC \cdot (1 + MJ) + MJ \cdot \frac{V_{BS}}{PB}\right)}{(1 - FC)^{1+MJ}}$$

$$C_{BSS} = \frac{(1 - FC \cdot (1 + MJSW) + MJSW \cdot \frac{V_{BS}}{PBSW})}{(1 - FC)^{1+MJSW}}$$

$$C_{BDJ} = \frac{\left(1 - FC \cdot (1 + MJ) + MJ \cdot \frac{V_{BD}}{PB}\right)}{(1 - FC)^{1+MJ}}$$

$$C_{BDS} = \frac{(1 - FC \cdot (1 + MJSW) + MJSW \cdot \frac{V_{BD}}{PBSW})}{(1 - FC)^{1+MJSW}}$$

**Temperature dependent parameters**

The following parameters are functions of temperature. T is the operating temperate and TNOM is the nominal (or measured temperature). T and TNOM can be adjusted in a number of ways.

$$IS(T) = IS \cdot EXP\left(\frac{\frac{TNOM}{T} \cdot Eg(TNOM) - Eg(T)}{V_t}\right)$$

$$JS(T) = JS \cdot EXP\left(\frac{\frac{TNOM}{T} \cdot Eg(TNOM) - Eg(T)}{V_t}\right)$$

$$JSSW(T) = JSSW \cdot EXP\left(\frac{\frac{TNOM}{T} \cdot Eg(TNOM) - Eg(T)}{V_t}\right)$$

$$PB(T) = PB \cdot \frac{T}{TNOM} - 3 \cdot Vt \cdot ln(\frac{T}{TNOM}) - Eg(TNOM) \cdot \frac{T}{TNOM} + Eg(T)$$

$$PBSW(T) = PBSW \cdot \frac{T}{TNOM} - 3 \cdot Vt \cdot ln(\frac{T}{TNOM}) - Eg(TNOM) \cdot \frac{T}{TNOM} + Eg(T)$$

$$PHI(T) = PHI \cdot \frac{T}{TNOM} - 3 \cdot Vt \cdot ln(\frac{T}{TNOM}) - Eg(TNOM) \cdot \frac{T}{TNOM} + Eg(T)$$

$$CJ0(T) = CJ0 \cdot \left(1 + M \cdot \left(0.004 \cdot (T - TNOM) + \left(1 - \frac{VJ(T)}{VJ}\right)\right)\right)$$

$$KP(T) = KP \cdot \left(\frac{TNOM}{T}\right)^{1.5}$$

$$U0(T) = U0 \cdot \left(\frac{TNOM}{T}\right)^{1.5}$$

$$CBD(T) = CBD \cdot \left(1 + MJ \cdot \left(0.0004 \cdot (T - Tnom) + \frac{(1 - PB(T))}{PB}\right)\right)$$

$$CBS(T) = CBS \cdot \left(1 + MJ \cdot \left(0.0004 \cdot (T - Tnom) + \frac{(1 - PB(T))}{PB}\right)\right)$$

$$C_{BS}(T) = C_{BS} \cdot \left(1 + MJ \cdot \left(0.0004 \cdot (T - Tnom) + \frac{(1 - PB(T))}{PB}\right)\right)$$

*where*

$V_{t=}\frac{K \cdot T}{q}$, *where K is Boltzmann's constant and q is electron charge*

$$Eg(T) = 1.16 - \frac{0.000702 \cdot T^2}{T + 1108}$$

**Noise model**

The MOSFET has six noise generators; four thermal noise generators associated with the four ohmic resistances, a shot noise generator and a flicker noise generator associated with the channel.

Ohmic resistance noise:

$$\overline{Ird}^2 = \frac{4kT\Delta f}{RD}$$

$$\overline{Irs}^2 = \frac{4kT\Delta f}{RS}$$

$$\overline{Irg}^2 = \frac{4kT\Delta f}{RG}$$

$$\overline{Irb}^2 = \frac{4kT\Delta f}{RB}$$

Shot noise:

$$if\ NLEV < 3$$

$$\overline{Ishot}^2 = \frac{8 \cdot k \cdot T \cdot gm}{3}$$

$$else$$

$$\overline{Ishot}^2 = \frac{8 \cdot k \cdot T \cdot \beta \cdot (V_{GS} - V_{T0})}{3} \cdot \frac{1 + a + a^2}{1 + a}$$

$$where$$

$$\beta = KP \cdot \left(\frac{W}{Leff}\right)$$

$$a = f(x) = \begin{cases} 1 - \frac{V_{DS}}{V_{DSAT}}, & linear\ region \\ 0, & saturation\ region \end{cases}$$

Flicker noise:

$$if\ NLEV = 0$$

$$\overline{Iflicker}^2 = \frac{KF \cdot {I_{DS}}^{AF}}{Cox \cdot Leff^2 \cdot f}$$

$$else\ if\ NLEV = 1$$

$$\overline{Iflicker}^2 = \frac{KF \cdot {I_{DS}}^{AF}}{Cox \cdot Weff \cdot Leff \cdot f}$$

$$else$$

$$\overline{Iflicker}^2 = \frac{KF \cdot gm^2}{Cox \cdot Weff \cdot Leff \cdot f^{AF}}$$

### References


1. G. Massobrio and P. Antognetti, Semiconductor Device Modeling with SPICE, 2nd edition, McGraw-Hill, 1993.
2. A. Vladimirescu, The SPICE Book, Wiley, 1994.


### Related Information

*Device Temperature Parameters*

# Independent Voltage Source

Independent Voltage Source instance declaration syntax:

Vxxx node+ node- <<DC> dc_mag> <AC <ac_mag <ac_phase>>
<source_type(source_params)>

Independent Voltage Source instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| DC | DC voltage identifier. |
| AC | AC analysis input voltage identifier. |
| dc_mag | DC magnitude voltage value. |
| ac_mag | AC signal magnitude value. |
| ac_phase | AC signal phase value. |
| source_type | Type of source waveform (see table below). |
| source_params | Parameters for source waveform. |

The types of voltage source waveforms allowed by Multisim are:

| source_type | Description |
|---|---|
| PULSE | Pulse source. |
| SIN | Sinusoidal source. |
| SFFM | Single frequency FM source. |
| EXP | Exponential source. |
| PWL | Piecewise linear source. |
| PWLREPEAT | Repeating piecewise linear source. |
| PWLFILE | PWL source generated using data stored in a file. |
| PWLFILEREPEAT | Repeating PWL source generated using data stored in a file. |
| XM | Agilent 33120A Arbitrary Waveform Generator modulation. |
| XAM | Agilent 33120A Arbitrary Waveform Generator AM modulation. |
| XFM | Agilent 33120A Arbitrary Waveform Generator FM modulation. |
| XFSK | Agilent 33120A Arbitrary Waveform Generator FSK modulation. |
| XBST | Agilent 33120A Arbitrary Waveform Generator Burst modulation. |
| XSWP | Agilent 33120A Arbitrary Waveform Generator Sweep modulation. |
| XNOISE | Agilent 33120A Arbitrary Waveform Generator Noise modulation. |
| XARB | Agilent 33120A Arbitrary Waveform Generator ARB modulation. |

### Description

An independent voltage source is an ideal voltage source, fully driving the voltage between its nodes. Every voltage source statement has three optional portions describing its output behavior: DC portion which describes its behavior in DC analysis, AC portion which describes its behavior in AC analysis, and source_type which describes its behavior in Transient Analysis.

### Additional Notes

- In DC analysis, if the DC portion is not specified, but the source_type portion is specified, the output voltage is that of the source_type at time=0.0.
- In transient analysis, if the source_type portion is not specified, but the DC portion is specified, the output voltage for the duration of the analysis is the DC value.
- If neither the DC portion or the source_type portion are specified, the output is zero volts.
- Sources have special parameters that are enclosed in brackets; these brackets are optional.
- Due to the ambiguity of some source parameter names (SIN, for example) expressions used in source statements should always have curly brackets, {}, around them.
- The files used in PWLFILE and PWLFILEREPEAT should be plain text, with one comma-separated pair of numbers on every line of the file. For more details on the file format, refer to the *Piecewise Linear Source* section.

### Examples

V1 in 0 10

*1khz, 10V, 50% duty cycle voltage signal
V1 in ref pulse (10 0 0 1n 1n 0.5m 1m)

*5V DC source with a 1V voltage for AC analysis
V1 1 0 ac 1 dc 5

### Related Information

*Pulse Source*
*Sinusoidal Source*
*Single Frequency FM Source*
*Exponential Source*
*Piecewise Linear Source*
*Piecewise Linear File Source*
*Modulation Source*
*AM Modulation Source*
*FM Modulation Source*
*FSK Modulation Source*
*BST Modulation Source*
*SWP Modulation Source*
*Noise Source*
*XARB Source*

## Pulse Source

Pulse source instance declaration syntax:

```
PULSE( vi vp <td <tr <tf <pw <per>>>>> )
```

PULSE instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| vi | Initial value (voltage/current). |
| vp | Pulsed value voltage/current). |
| td | Time delay. |
| tr | Rise time. |
| tf | Fall time. |
| pw | Pulse width. |
| per | Period. |

## Sinusoidal Source

Sinusoidal source instance declaration syntax:

```
SIN( vo va <freq <td <df <theta>>>> )
```

SIN instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| vo | Offset value. |
| va | Peak value. |
| freq | Frequency. |
| td | Delay. |
| df | Damping factor. |
| theta | Phase. |

## Single Frequency FM Source

Single frequency FM source instance declaration syntax:

```
SFFM( vo va <fc <mdi <fs>>> )
```

SFFM instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| vo | Offset value. |
| va | Peak value. |
| fc | Carrier frequency. |
| mdi | Modulation index. |
| fs | Signal frequency. |

## Exponential Source

Exponential source instance declaration syntax:

```
EXP( vi vp <td1 tau1 td2 <tau2>> )
```

EXP instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| vi | Initial value. |
| vp | Peak value. |
| td1 | Rise/fall delay time. |
| tau1 | Rise/fall time constant. |
| td2 | Fall/rise delay time. |
| tau2 | Fall/rise time constant. |

## Piecewise Linear Source

Piecewise linear source instance declaration syntax:

```
PWL( t1 v1 <t2 v2 <t3 v3 <...>>> )
```

```
PWL( t1, v1 ) <( t2, v2 ) <( t3, v3 ) <...>>> )
```

Repeating piecewise linear source instance declaration syntax:

```
PWLREPEAT( t1 v1 <t2 v2 <t3 v3 <...>>> )
```

```
PWLREPEAT( t1, v1 ) <( t2, v2 ) <( t3, v3 ) <...>>> )
```

PWL and PWLREPEAT instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| tn where n=0,1,2… | Time component of data point *n*. |
| vn where n=0,1,2… | Voltage component of data point *n*. |

## Piecewise Linear File Source

File-based piecewise linear source instance declaration syntax:

`PWLFILE( filename_string )`

Repeating File-based piecewise linear source instance declaration syntax:

`PWLFILEREPEAT( filename_string )`

PWLFILE and PWLFILEREPEAT instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| filename_string | A fully qualified filename in quotes (for example, `c:\mypath\myfile.ext`). |

## Modulation Source

This source models the modulation available on the Agilent 33120A Arbitrary Waveform Generator.

XM instrument voltage source instance declaration syntax:

`XM( vo va fc td dt cwid )`

| Parameter Name | Parameter Description |
|---|---|
| vo | Offset value. |
| va | Amplitude value. |
| fc | Carrier frequency. |
| td | Time delay. |
| dt | Duty value for square wave. |
| cwid | Carrier wave ID: 0=sine, 1=square, 2=triangle, 3=ramp. |

## AM Modulation Source

This source models the AM modulation available on the Agilent 33120A Arbitrary Waveform Generator.

AM modulation voltage source instance declaration syntax:

`XAM( vo va fc td dt depth fs cwid wsid )`

| Parameter Name | Parameter Description |
|---|---|
| vo | Offset value. |
| va | Amplitude value. |
| fc | Carrier frequency. |
| td | Time delay. |
| dt | Duty value for square wave. |
| depth | Modulation depth. |
| fs | Signal frequency. |
| cwid | Carrier wave ID: 0=sine, 1=square, 2=triangle, 3=ramp. |
| wsid | Signal wave ID: 0=sine, 1=square, 2=triangle, 3=ramp. |

## FM Modulation Source

This source models the FM modulation available on the Agilent 33120A Arbitrary Waveform Generator.

FM modulation voltage source instance declaration syntax:

`XFM( vo va fc mdi fs cwid wsid )`

| Parameter Name | Parameter Description |
|---|---|
| vo | Offset value. |
| va | Amplitude value. |
| fc | Carrier frequency. |
| mdi | Modulation index. |
| fs | Signal frequency. |
| cwid | Carrier wave ID: 0=sine, 1=square, 2=triangle, 3=ramp. |
| wsid | Signal wave ID: 0=sine, 1=square, 2=triangle, 3=ramp. |

## FSK Modulation Source

This source models the FSK modulation available on the Agilent 33120A Arbitrary Waveform Generator.

FSK modulation voltage source instance declaration syntax:

`XFSK(` vo va fc td dt fh fskr `)`

| Parameter Name | Parameter Description |
|---|---|
| vo | Offset value. |
| va | Amplitude value. |
| fc | Carrier frequency. |
| td | Time delay. |
| dt | Duty value for square wave. |
| fh | Hop frequency. |
| fskr | FSK rate. |

## BST Modulation Source

This source models the Burst modulation available on the Agilent 33120A Arbitrary Waveform Generator.

BST modulation voltage source instance declaration syntax:

`XBST(` vo va fc td dt bstr nc ph cwid tm te `)`

| Parameter Name | Parameter Description |
|---|---|
| vo | Offset value. |
| va | Amplitude value. |
| fc | Carrier frequency. |
| td | Time delay. |
| dt | Duty value for square wave. |
| bstr | BST rate. |
| nc | BST counter. |
| ph | Phase (in degrees). |
| cwid | Carrier wave ID: 0=sine, 1=square, 2=triangle, 3=ramp. |
| tm | Trigger mode: 0=auto, 1=single. |
| te | Trigger enable: 0=disabled, 1=enabled. |

## SWP Modulation Source

This source models the Sweep modulation available on the Agilent 33120A Arbitrary Waveform Generator.

SWP modulation voltage source instance declaration syntax:

`XSWP( vo va td dt fb fe ts sm cwid tm te)`

| Parameter Name | Parameter Description |
|---|---|
| vo | Offset value. |
| va | Amplitude value. |
| td | Time delay. |
| dt | Duty value for square wave. |
| fb | Start frequency. |
| fe | End frequency. |
| ts | Sweep time. |
| sm | Sweep mode: 1=linear, 0=log. |
| cwid | Signal wave ID: 0=sine, 1=square, 2=triangle, 3=ramp. |
| tm | Trigger mode: 0=auto, 1=single. |
| te | Trigger enable: 0=disabled, 1=enabled. |

## Noise Source

This source models the Noise modulation available on the Agilent 33120A Arbitrary Waveform Generator.

NOISE modulation voltage source instance declaration syntax:

`XNOISE( vo va fc bf)`

| Parameter Name | Parameter Description |
|---|---|
| vo | Offset value. |
| va | Amplitude value. |
| fc | Carrier frequency. |
| bf | Bandwidth frequency. |

## XARB Source

This source models the ARB modulation available on the Agilent 33120A Arbitrary Waveform Generator.

XARB voltage source instance declaration syntax:

```
XARB( tt td t0 v0 t1 v1 <t2 v2 <...>>)
```

| Parameter Name | Parameter Description |
|---|---|
| tt | Period time (1/F). |
| td | Time delay. |
| tn where n=0,1,2… | Time component of data point *n*. |
| vn where n=0,1,2… | Voltage component of data point *n*. |

# Independent Current Source

Independent Current Source instance declaration syntax:

```
Ixxx node+ node- <<DC> dc_mag> <AC <ac_mag <ac_phase>>
<source_type(source_params)>
```

Independent Current Source instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| DC | DC current identifier. |
| AC | AC analysis input current identifier. |
| dc_mag | DC magnitude current value. |
| ac_mag | AC signal magnitude value. |
| ac_phase | AC signal phase value. |
| source_type | Type of source waveform (see table below). |
| source_params | Parameters for source waveform. |

The types of current source waveforms allowed by Multisim are:

| source_type | Description |
|---|---|
| PULSE | Pulse source. |
| SIN | Sinusoidal source. |
| SFFM | Single frequency FM source. |

| source_type | Description |
|---|---|
| EXP | Exponential source. |
| PWL | Piecewise linear source. |
| PWLREPEAT | Repeating piecewise linear source. |
| PWLFILE | PWL source generated using data stored in a file. |
| PWLFILEREPEAT | Repeating PWL source generated using data stored in a file. |

### Description

An independent current source is an ideal current source which drives current in its branch in the direction from `node+` to `node-`. Every current source statement has three optional portions describing its output behavior: DC portion which describes its behavior in DC analysis, AC portion which describes its behavior in AC analysis, and `source_type` which describes its behavior in Transient Analysis.

### Additional Notes

- In DC analysis, if the DC portion is not specified, but the source_type portion is specified, the output voltage is that of the `source_type` at time=0.0.
- In transient analysis, if the `source_type` portion is not specified, but the DC portion is specified, the output voltage for the duration of the analysis is the DC value.
- If neither the DC portion nor the `source_type` portion are specified, the output is zero.
- Sources have special parameters that are enclosed in brackets; these brackets are optional.
- Due to the ambiguity of some source parameter names (SIN, for example) expressions used in source statements should always have curly brackets, {}, around them.
- The files used in PWLFILE and PWLFILEREPEAT should be plain text, with one comma-separated pair of numbers on every line of the file.

### Examples

I1 0 in 10

*1khz, 5Apk current source
I1 0 in sin (0 5 1k)

*5A DC source with a 1A for AC analysis
I1 0 1 ac 1 dc 5

### Related Information

*Pulse Source*
*Sinusoidal Source*
*Single Frequency FM Source*
*Exponential Source*
*Piecewise Linear Source*
*Piecewise Linear File Source*

# Arbitrary Sources

Arbitrary sources generate an output voltage or current that is the result of a mathematical expression, and, in some types of Arbitrary sources, of further processing.

The ablity to describe both non-linear behaviour, using a vast amount of functions and operators, and dynamic behaviour, using differential functions or special source types, truly makes this collection of devices “arbitrary”. They are the essential building blocks of macro and behavioral models.

Their advanced functionality obsoletes the limited functionality provided by the specific controlled source devices (for example, voltage controlled voltage source), which are primarily supported for legacy and compatibility reasons.

### Related Information

*VALUE Type Source*
*TABLE Type Source*
*FLUX and Q Type Sources*
*LAPLACE and FREQ Type Sources*

## VALUE Type Source

Instance declaration syntax:

```
Exxx o_node_p  o_node_n VALUE  = {expression}
Gxxx o_node_p  o_node_n VALUE  = {expression}

Bxxx o_node_p  o_node_n V  = expression
Bxxx o_node_p  o_node_n I  = expression
```

Instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| expression | A mathematical expression operating on circuit variables (voltages and currents). |

### Description

In these sources the output is simply the result of the `expression`.

The B-source with the declaration "`Bxxx o_node_p  o_node_n V  = expression`" is a controlled voltage source and is functionality equivalent to the E-source of the VALUE type with the declaration "`Exxx o_node_p  o_node_n VALUE  = expression`".

The B-style with the declaration "`Bxxx o_node_p  o_node_n` I = `expression`" is a controlled current source and is functionality equivalent to the G-source of the VALUE source with the declaration "`Gxxx o_node_p  o_node_n` VALUE = `expression`".

### Examples

E99 out 0 value={v(in)/5}
Gout 33 0 value={ddt(v(33))*1u}
B99 5 0 i=limit(log10(v(5)),-100,10)

### Related Information

*Mathematical Expressions*

## TABLE Type Source

Instance declaration syntax:

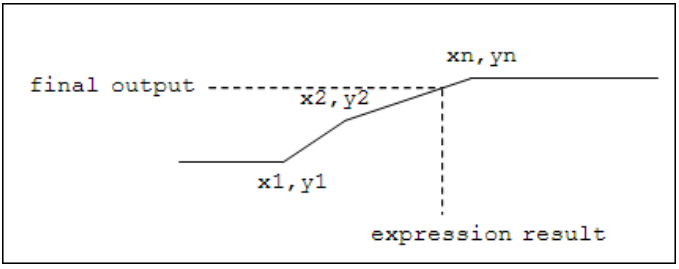
```
Exxx o_node_p o_node_n TABLE{expression} < = > (x1,y1) (x2,y2) <
(x3,y3) <...>>
```

Instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| expression | A mathematical expression operating on circuit variables (voltages and currents). |
| xN where N=0,1,2… | Nth x value for the TABLE source. |
| yN where N=0,1,2… | Nth y value for the TABLE source. |

### Description

The TABLE type source has the functionality of the VALUE type source, but adds the ability to further process the value of `expression` by mapping it to a piece-wise-linear function described by co-ordinates (x1,y1) (x2,y2)… (xn,yn).




The E-source has voltage output, whereas the G-source has current output.

### Example

E1 5 0 TABLE(V(1)**2) (-5,-5)(5,5)
GD99 2 13 TABLE {V(2,13)}=((-100,-1p)(0,0)(1m,1n)(2m,1m)(3m,1))

### Related Information

*Mathematical Expressions*

## FLUX and Q Type Sources

Instance declaration syntax:

```
Exxx o_node_p o_node_n FLUX = {expression}

Gxxx o_node_p o_node_n Q = {expression}
```

Instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| expression | A mathematical expression operating on circuit variables (voltages and currents). |

The FLUX type source takes the derivative of `expression`. It can only be specified in E-style and has a voltage output.

The Q source's takes the derivative of `expression`. It can only be specified in G-style and has a current output.

These sources are useful for modeling non-linear inductors and capacitors.

### Examples

E1 out1 out2 FLUX={v(InductorValNode)*I(E1)}
Q1 out1 out2 Q={1u*V(out1,out2)}

### Related Information

*Mathematical Expressions*

## LAPLACE and FREQ Type Sources

Instance declaration syntax:

```
Exxx o_node_p o_node_n LAPLACE{expression} < = > {s-expression} <
fmax= fmax > < fres= fres > <db> <mag> <deg> <rad> <R_I> < delay=
delay >
Exxx o_node_p o_node_n FREQ{expression} < = >
(f1,ga1,gb1)(f2,ga2,gb2)<(f3,ga3,gb3)<...>> < fmax= fmax > < fres=
fres > <mag> <db> <deg> <rad> <R_I> < delay= delay >

Gxxx o_node_p o_node_n LAPLACE{expression} < = > {s-expression} <
fmax= fmax > < fres= fres > <db> <mag> <deg> <rad> <R_I> < delay=
delay >
Gxxx o_node_p o_node_n FREQ{expression} < = >
(f1,ga1,gb1)(f2,ga2,gb2)<(f3,ga3,gb3)<...>> < fmax= fmax > < fres=
fres > <mag> <db> <deg> <rad> <R_I> < delay= delay >
```

Instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| expression | A mathematical expression operating on circuit variables (voltages and currents). |
| s-expression | A mathematical expression describing the frequency response of the LAPLACE source. It operates on the complex frequency variable “s”. |
| fN where N=0,1,2… | Nth frequency value for the FREQ source frequency response. |
| gaN where N=0,1,2… | Nth frequency gain value (real or magnitude) for the FREQ source frequency response. |
| gbN where N=0,1,2… | Nth frequency gain value (imaginary or phase) for the FREQ source frequency response. |
| fmax | Maximum frequency response frequency sampled. Applicable to LAPLACE and FREQ sources. |
| fres | Frequency response sampling interval. Applicable to LAPLACE and FREQ sources. |
| db | Flag which indicates that the units of the `gaN` values in the FREQ sources are decibels. |
| mag | Flag which indicates that the units of the `gaN` values in the FREQ sources are absolute. |
| deg | Flag which indicates that the units of the `gbN` values in the FREQ sources are degrees. |
| rad | Flag which indicates that the units of the `gbN` values in the FREQ sources are radians. |
| r_i | Flag which indicates that gain components of the FREQ source are specified using real, imaginary format rather than magnitude phase format. That is, `gaN` values are the real components of the gain and `gbN` are the imaginary components of the gain. |
| delay | Group delay value for the frequency response of FREQ source. It has the effect of reducing the slope of the phase. |

### Description

LAPLACE type source:

The LAPLACE source is much more unique than the Arbitrary Sources in previous sections. It is a device with an frequency response, which is typically used to describe dynamic characteristics. The frequency response is accounted for in addition to the value of `expression`.

The frequency response is specified as a mathematical expression operating on the complex frequency variable “s”. The allowed mathematical operators and functions are all those permitted by Multisim’s mathematical expression engine. The source’s most basic use may be to model frequency selective macro models such as filters. But because the source supports more than just operators needed to describe rational polynomials, it can be used to describe more advanced linear systems behaviour, such as advanced inductors and transmission lines.

FREQ type source:

The major difference between the FREQ source and the LAPLACE source is that in the FREQ source the frequency response is specified as a set of discrete data instead of as an equation.

The optional flags `db`, `mag`, `deg`, `rad`, and `r_i` indicate to Multisim the format and units of the gain components in the data set. If not specified, Multisim assumes that the gain is in `decibels,degrees` format. It is expected that a sensical combination of these flags is used: if r_i is used, do not use any other flags; do not use `db` and `mag` together; and do not use `deg` and `rad` together.

### Implementation Overview

Both source types obtain the impulse response from a sampled version of the frequency response (using an inverse Fourier Transform) and then use the impulse response to perform a convolution with the input (result of `expression`). Due to this sampling, the overall result is prone to inaccuracies.

Two parameters control the sampling: The maximum frequency sampled and the frequency sampling resolution. By default, Multisim automatically chooses these parameters based on both the simulation settings (`TSTOP` and `TMAX`) and the frequency response characteristics. However, there may be times when there is disagreement between what Multisim determines is reasonable and what is acceptable to you. In such cases, you can override these parameters using the `Fmax` (Maximum Frequency) and `Fres` (Frequency Resolution) parameters. However, due to computational constraints, the following restriction exists:

```
Frequency Resolution > 2*Maximum Frequency / 65536
```

If the above is not obeyed, then the Frequency Resolution is increased to `2*Maximum Frequency / 65536`.

The sampling values which Multisim actually uses to calculate the impulse response are always reported in Multisim’s **Simulation Error Log/Audit Trail**.

The E-source has voltage output, whereas the G-source has current output.

### Examples

```
e1 2 0 laplace v(1) = {250/(0.0008*s+3e-8*s^2+250)} fmax=200k fres=100
gax ai a2 laplace {i(VB)} = {-exp(-2*sqrt((1.6m*s+10e3)*(1u*s+0.03)))}

E1 out 0 FREQ {v(in)} = (1hz, 0, 0)(100hz, -3, -30)(1k, -60,-90)(100k, -100, -180) db deg
e_Y00_X1 PORT1 0 FREQ {V(1,0)} =
+(1.0000000000000000e-003,-4.4036231270621791e+001,8.9639990604640673e+001)
```

+(1.0055196478890240e-003,-4.3988418055769841e+001,8.9637905176544564e+001)
+(1.0110392957780479e-003,-4.3940866586755206e+001,8.9635842519681660e+001)
+(1.0165589436670721e-003,-4.3893574013523697e+001,8.9633802263116522e+001)

**Related Information**
*Mathematical Expressions*

# Controlled Sources

The functionality of the Controlled Sources is a subset of that provided by the Arbitrary Sources. The main limitation of Controlled Sources is that the controlling entity is restricted to an equation that is of polynomial form and may reference only voltages or only currents. They are supported primarily for compatability reasons.

**Related Information**
*Voltage Controlled Voltage Source*
*Current Controlled Voltage Source*
*Voltage Controlled Current Source*
*Current Controlled Current Source*
*Polynomial Specifications*

## Voltage Controlled Voltage Source

Voltage Controlled Voltage Source instance declaration syntax:

Exxx o_node_p o_node_n c_node0_p c_node0_n gain

Exxx o_node_p o_node_n <POLY(ndim) > c_node0_p c_node0_n < c_node1_p c_node1_n <...>> p0 p1 <p2 <p4 <...>>>

Voltage Controlled Voltage Source instance declaration terminals:

| Parameter Name | Parameter Description |
|---|---|
| o_node_p | Output node +. |
| o_node_n | Output node -. |
| c_node*N*_p where N=0,1,2… | *N*th controlling node +. |
| c_node*N*_n where N=0,1,2… | *N*th controlling node -. |

Voltage Controlled Voltage Source instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| gain | Voltage gain. This is not applicable to the POLY source. |
| ndim | The number of pairs of controlling nodes. This must be a simple integer, not an expression. |
| p*N* where *N*=0,1,2… | *N*th polynomial term. |

### Additional Notes

The dimension parameter of the polynomial source (ndim) must be a numerical constant. It may not contain parameters or expressions.

### Examples

E1 ampout 0 ampin 0 1e6
E1 out ref poly(2) node1 0 input+ input- 0.5 1 1

### Related Information

*Polynomial Specifications*

## Current Controlled Voltage Source

Current Controlled Voltage Source instance declaration syntax:

Hxxx o_node_p o_node_n Vname1 gain

Hxxx o_node_p o_node_n < POLY(ndim) > Vname1 < Vname2 <...>> p0 p1 <p2 <p4 <...>>>

Current Controlled Voltage Source instance declaration terminals:

| Parameter Name | Parameter Description |
|---|---|
| o_node_p | Output node +. |
| o_node_n | Output node -. |
| c_node*N*_p where *N*=0,1,2… | *N*th controlling node +. |
| c_node*N*_n where *N*=0,1,2… | *N*th controlling node -. |
| Vname*N* where *N*=0,1,2… | *N*th controlling current source name. |

Current Controlled Voltage Source instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| gain | Voltage gain. This is not applicable to the POLY source. |
| ndim | The number of pairs of controlling nodes. This must be a simple integer. |
| vo | Voltage offset. |
| p*N* where *N*=0,1,2… | Nth polynomial term. |

### Additional Notes

- The dimension parameter of the polynomial source (ndim) must be a numerical constant - it may not contain parameters or expressions.

### Example

H1 out 0 Vsense 1k

### Related Information

*Polynomial Specifications*

## Voltage Controlled Current Source

Voltage Controlled Current Source instance declaration syntax:

Gxxx o_node_p o_node_n c_node0_p c_node0_n gain

Gxxx o_node_p o_node_n < POLY(ndim) > c_node0_p c_node0_n < c_node1_p c_node1_n <...>> p0 p1 <p2 <p4 <...>>>

Voltage Controlled Current Source instance declaration terminals:

| Parameter Name | Parameter Description |
|---|---|
| o_node_p | Output node +. |
| o_node_n | Output node -. |
| c_node*N*_p where *N*=0,1,2… | *N*th controlling node +. |
| c_node*N*_n where *N*=0,1,2… | *N*th controlling node -. |

Voltage Controlled Voltage Source instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| gain | Voltage gain. This is not applicable to the POLY source. |
| ndim | The number of pairs of controlling nodes. This must be a simple integer, not an expression. |
| vo | Voltage offset. |
| p*N* where *N*=0,1,2… | Nth polynomial term. |

### Additional Notes

- The dimension parameter of the polynomial source (ndim) must be a numerical constant—it may not contain parameters or expressions.

### Example

G1 0 out 1 2 100k
G1 out ref poly(2) node_a 0 node_b 0 0.5 1 1 2 2

### Related Information

*Polynomial Specifications*

## Current Controlled Current Source

Current Controlled Current Source instance declaration syntax:

Fxxx o_node_p o_node_n Vname1 gain

Fxxx o_node_p o_node_n < POLY(ndim) > Vname1 < Vname2 <...>> p0 p1 <p2 <p4 <...>>>

Current Controlled Current Source instance declaration terminals:

| Parameter Name | Parameter Description |
|---|---|
| o_node_p | Output node +. |
| o_node_n | Output node -. |
| c_node*N*_p where *N*=0,1,2… | *N*th controlling node +. |
| c_node*N*_n where *N*=0,1,2… | *N*th controlling node -. |
| Vname*N* where *N*=0,1,2… | *N*th controlling current source name. |

Current Controlled Voltage Source instance declaration parameters:

| Parameter Name | Parameter Description |
|---|---|
| gain | Voltage gain. This is not applicable to the POLY source. |
| ndim | The number of pairs of controlling nodes. This must be a simple integer, not an expression. |
| vo | Voltage offset. |
| p*N* where *N*=0,1,2… | Nth polynomial term. |

### Additional Notes

- The dimension parameter of the polynomial source (`ndim`) must be a numerical constant—it may not contain parameters or expressions.

### Example

F1 0 out V1 10

### Related Information

*Polynomial Specifications*

## Polynomial Specifications

In polynomial controlled sources the `ndim` value provided is the number of controlling inputs, which are either pairs of nodes or names of current sources. The number of inputs must match the `ndim` value, however the number of polynomial terms is independent.

There can be any number of polynomial terms which are used to generate an expression of the following form:

For POLY(1):

```
p0*a + p1*a*a + p2*a*a*a + p3*a*a*a*a + ...
```

where `a` is the value of `V(c_node0_p, c_node0_n)` or `I(c_isrc0)`

For POLY(2):

```
p0*a + p1*b + p2*a*a + p3*a*b + p4*b*b + p5*a*a*a + p6*a*a*b +
p7*a*b*b + p8*b*b*b + p9*a*a*a*a + p10*a*a*a*b +...
```

where `a` is the value of `V(c_node0_p, c_node0_n)` or `I(c_isrc0)`
and `b` is the value of `V(c_node1_p, c_node1_n)` or `I(c_isrc1)`

For POLY(3):

```
p0*a + p1*b + p2*c + p3*a*a + p4*a*b + p5*a*c + p6*b*b + p7*b*c
+ p8*c*c + p9*a*a*a + p10*a*a*b + p11*a*a*c + p12*a*b*b +
p13*a*b*c + p14*a*c*c + p15*b*b*b + p16*b*b*c + p17*b*c*c +
p18*c*c*c + p19*a*a*a*a + ....
```

where `a` is the value of `V(c_node0_p, c_node0_n)` or `I(c_isrc0)`

and b is the value of V(c_node1_p, c_node1_n) or I(c_isrc1)

and c is the value of V(c_node2_p, c_node2_n) or I(c_isrc2)

etc…

### Additional Notes

The dimension parameter of the polynomial source (ndim) must be a numerical constant—it may not contain parameters or expressions.

### Example

**The line below is the relationship v(out)=0.5 + 2*v(1) + 3*v(2) + 4*v(2)^2
E1 out 0 poly(2) 1 0 2 0 0.5 2 3 4

# Digital and Mixed-Mode Devices Library

Digital devices are devices (lowest level modeling elements) that are simulated by the digital simulation engine. This section describes all such devices supported in Multisim.

### Related Information

*AND Gate*
*OR Gate*
*NOT Gate*
*NOR Gate*
*NAND Gate*
*XOR Gate*
*XNOR Gate*
*Tri-State*
*Buffer*
*Delay*
*Digital Chip*
*D-Latch*
*SR-Latch*
*D Flip-Flop*
*Toggle Flip-Flop*
*JK Flip-Flop*
*Constant Source*
*Clock Source*
*Arbitrary Digital Source*
*Analog to Digital Bridges*
*Digital to Analog Bridges*

# AND Gate

AND gate instance declaration syntax:

```
Axxxx [in_1 in_2 <in_3…in_n>] out MyModel
```

AND gate model definition syntax:

```
.MODEL MyModel d_and (<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| RISE_DELAY | LOW to HIGH rise delay. | s | 1e-9 |
| FALL_DELAY | HIGH to LOW rise delay. | s | 1e-9 |

**Example**

A1 [1 22] 4 myAND
.model myAND d_and(rise_delay=10n fall_delay=10n)

# OR Gate

OR gate instance declaration syntax:

```
Axxxx [in_1 in_2 <in_3…in_n>] out MyModel
```

OR gate model definition syntax:

```
.MODEL MyModel d_or(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| RISE_DELAY | LOW to HIGH rise delay. | s | 1e-9 |
| FALL_DELAY | HIGH to LOW rise delay. | s | 1e-9 |

**Example**

A1 [1 22 33] 4 myOR
.model myOR d_or(rise_delay=10n fall_delay=10n)

# NOT Gate

NOT gate instance declaration syntax:

```
Axxxx in out MyModel
```

NOT gate model definition syntax:

```
.MODEL MyModel d_inverter (<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| RISE_DELAY | LOW to HIGH propagation delay. | s | 1e-9 |
| FALL_DELAY | HIGH to LOW propagation delay. | s | 1e-9 |

**Example**
A1 1 2 not
.model not d_inverter(rise_delay=10n fall_delay=10n)

# NOR Gate

NOR gate instance declaration syntax:

```
Axxxx [in_1 in_2 <in_3…in_n>] out MyModel
```

NOR gate model definition syntax:

```
.MODEL MyModel d_nor (<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| RISE_DELAY | LOW to HIGH propagation delay. | s | 1e-9 |
| FALL_DELAY | HIGH to LOW propagation delay. | s | 1e-9 |

**Example**
A1 [1 22 33] 4 myNOR
.model myNOR d_nor(rise_delay=10n fall_delay=10n)

# NAND Gate

NAND gate instance declaration syntax:

```
Axxxx [in_1 in_2 <in_3…in_n>] out MyModel
```

NAND gate model definition syntax:

```
.MODEL MyModel d_nand (<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| RISE_DELAY | LOW to HIGH propagation delay. | s | 1e-9 |
| FALL_DELAY | HIGH to LOW propagation delay. | s | 1e-9 |

**Example**

A1 [1 22 33 44] 5 myNAND
.model myNAND d_nand(rise_delay=10n fall_delay=10n)

# XOR Gate

XOR gate instance declaration syntax:

```
Axxxx [in_1 in_2 <in_3…in_n>] out MyModel
```

XOR gate model definition syntax:

```
.MODEL MyModel d_xor (<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| RISE_DELAY | LOW to HIGH propagation delay. | s | 1e-9 |
| FALL_DELAY | HIGH to LOW propagation delay. | s | 1e-9 |

**Description**

An XOR gate generates an output HIGH if the number of inputs in state HIGH is odd.

**Example**

A1 [1 22 33 44] 5 myXOR
.model myXOR d_xor(rise_delay=10n fall_delay=10n)

# XNOR Gate

XNOR gate instance declaration syntax:

```
Axxxx [in_1 in_2 <in_3…in_n>] out MyModel
```

XNOR gate model definition syntax:

```
.MODEL MyModel d_xnor (<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| RISE_DELAY | LOW to HIGH propagation delay. | s | 1e-9 |
| FALL_DELAY | HIGH to LOW propagation delay. | s | 1e-9 |

### Description

An XNOR gate generates an output HIGH if the number of inputs in state HIGH is even.

### Example

A1 [1 22 33 44] 5 myXNOR
.model myXNOR d_xnor(rise_delay=10n fall_delay=10n)

# Tri-State

Tri-state instance declaration syntax:

```
Axxxx in enable out MyModel
```

Tri-state model definition syntax:

```
.MODEL MyModel d_tristate (<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| DELAY | Propogation delay for all in and enable. | s | 1e-9 |

### Description

If the signal on the `enable` pin is HIGH, then the signal on the input is passed to the output with a digital strength of STRONG. If the signal level on the `enable` pin is LOW, then the signal level on the input is passed to the output with a digital strength of HI-IMPEDANCE.

### Example

A1 1 ctrl 5 myTri
.model myTri d_tristate

# Buffer

Buffer element instance declaration syntax:

```
Axxxx in out MyModel
```

Delay element model definition syntax:

```
.MODEL MyModel d_buffer(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| RISE_DELAY | LOW to HIGH propagation delay. | s | 1e-9 |
| FALL_DELAY | HIGH to LOW propagation delay. | s | 1e-9 |

### Description

This device is used to convert signals with HI-IMPEDANCE strength to those with STRONG strength. Additionally it can be used as a delay element. Note however that, unless otherwise specfied in the Simulation Options, the device exhibit inertial delay characteristics and thus it will swallow pulses that are shorter than the values of its delay parameters. Use the `d_delay` device to provide a transport delay.

### Example

A1 1 2 delay
.model delay d_delay(delay_type="transport")

# Delay

Delay element instance declaration syntax:

```
Axxxx in out MyModel
```

Delay element model definition syntax:

```
.MODEL MyModel d_delay(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| RISE_DELAY | LOW to HIGH propagation delay. | s | 1e-9 |
| FALL_DELAY | HIGH to LOW propagation delay. | s | 1e-9 |
| INPUT_LOAD | Digital load added to the input. | F | 1e-12 |
| DELAY_TYPE | Delay type characteristic—“inertial” or “transport”. | — | inertial |

### Description
This device is used to delay the signal as seen on the input. It can be configured to exhibit transmission line style delay, or Transport Delay, using the `delay_type` parameter.

### Example
A1 1 2 delay
.model delay d_delay(delay_type="transport")

# Digital Chip
Digital Chip instance declaration syntax:

```
Axxxx [in_1 in_2 <in_3…in_n>] [out_1 out_2 <out_3…out_n>] MyModel
```

Digital Chip model definition syntax:

```
.MODEL MyModel d_chip(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| BEHAVIOUR | Text string describing the logical behavior. The format and commands are described below. | — | — |

### Description
This device is used for modeling customized logic behavior using a truth table style approach. The device supports states, and thus allows for the description of sequential logic. The format is specific to Multisim only; it does not follow a standard. The logic description is passed in as text parameter to the *behavior* model parameter.

### Format
The general format is to specify a command and the number of lines in the content that follows the command. Commands are specified using the `/command_name` convention and must be placed at the beginning of a line.

Because the behavior is passed in as string of text, to tell the netlist parser that every new line of text is actually a continuation of the old line, the plus '+' character is needed on every new line. For example:

```
.model myCounter d_chip(behaviour= "
+;We have here the beginning of some counter
+/inputs  ~G B A D0 D1 D2 D3
+/outputs Y ~W
+/table 5
+more …
+")
```

Comments are applied using the semicolon ';' character.

The digital chip device supports three of the six digital signal states that are available in the digital simulator (be careful not to confuse signal states, which are simply digital values correponding to nodes, with logic states, which are representations of some states within a state machine). The three signal states are:

HIGH level with STRONG Strength - denoted as ‘H’

LOW Level with STRONG Strength - denoted as ‘L’

UNKNOWN Level with HI-IMPEDANCE Strength - denoted as ‘Z’

**Related Information**
*/input, /output*
*/table*
*/clock*
*/delay*
*/wires*
*/module/instance*

## /input, /output

The `/input` and `/output` commands map the internal input/output variables to the calling entity’s instance node list (ussually the netlist nodes). These commands must appear prior to any other commands in the logic description.

The `/input` command specifies the first set of variable-node pairs and the `/output` command specifies the remaining set of variable-node pairs. In the example below, ~G is mapped to 4, B is mapped to 6, A is mapped to 7, D0 is mapped to 8, Y is mapped to Y, and ~W is mapped to notY.

```
aU1  [4 6 7 8] [Y notY] MUX_4TO1

.model myCounter d_chip(behaviour= "
+;We have here the beginning of some counter
+/inputs  ~G B A D0
+/outputs Y ~W
```

## /table

The `/table` command is used to specify an output truth table which specifies output variable values as a function of input variables, and, in the case of sequential logic, of state variables. This command is fundamental to the Digital Chip device and must be present in all cases. The `/table` command has a single parameter that describes the number of lines, or rows, in the truth table that follows in the subsequent lines. The columns in the table represent the set of all input and output node variables in the order that they were defined. All input combinations possibilities must be exhausted by the truth table. However, using the symbol ‘X’ to a denote “Don’t Care” conditions can dramatically reduce the number of rows in the table. Below is an example of an AND gate.

```
U1  [in1 in2] [out] myAND
.model myAND d_chip(behaviour= "
+;Simple And gate
+/inputs  A B
+/outputs Y
+/table 2
+;A B Y
+H H H
+X X L
+")
```

During simulation, scanning from top to bottom, the simulator evaluates the first ‘true’ condition in the truth table  As such, when specifying “Dont Care” conditions it is important to specify them in the right order.

## /clock

Description of sequential logic is made possible using the `/clock` command. The description of the logic resembles a state transition table used in the design of finite state machines (FSMs). Both synchrounous and asynchrounous logic is supported. The `/clock` command and its parameters are as follows:

```
/CLOCK  clk_name  edge  NumOfFlags  NumOfSync  NumOfAsync
```

| | |
|---|---|
| `clk_name` | The input variable to which the synchronous truth table is sensitized. |
| `edge` | Positive or negative edge detection (‘+’ or ‘-’). |
| `NumOfFlags` | Number of flags used to symbolize the state machine states. Flags are symbolized using the ‘`Fn`’ notation, where `n` is an integer from `0` to `NumOfFlags-1`. |
| `NumofSync` | Number of lines/rows in the synchronous state truth table. |
| `NumOfAsync` | Number of lines/rows in the asynchronous state truth table. |

The `/clock` command must be followed by a synchronous state transition table. This table is used to assign values to the state flags and is evaluated when the `clk_name` variable makes a transition as defined by the *edge* parameter. The first set of columns correspond to the inputs variables. The next set of columns correspond to the current state flags - the state just before the edge. The current state flags, together with the input variables, are the inputs to the truth table. The last set of columns correspond to the next state flags - the state just after the edge. The next state flags are the outputs of the truth table. The number of columns in each current state and

next state fields must equal `NumOfFlags`. Note that values of the flags are overwritten with every evaluation of the synchronous truth table.

The asyncronous state transition table, which directly follows the synchronous transition table, is identical in every way except one: it is evaluated whenever any of the other inputs (that is, non-clock inputs) make a transition (LOW->HIGH or HIGH->LOW).

Note that neither the synchronous truth table nor the asyncronous truth table actually write values into any of the output variables; the output truth table (`/table` command) must be used for that. In the context of the `/clock` command, since the output is a function of both the current state and input signal, the output truth table has extra column(s) for the current state flags.

The description of sequential logic using the `/clock` command is best illustrated by an example. Consider the following description of a D-type flip-flop.

```
aU1  [D Set Clear Clk] [Out notOut] logic
.model logic d_chip(behaviour= "
+/inputs D SD CD CP
+/outputs O  ~O
+;clock input_number edge{+|-} number_of_flags sync_entries
async_entries
+/clock CP + 1 2 3
+;The synchronous table
+; D  SD  CD  CP CurrentFlagState  NextFlagState
+H X X X X H
+L X X X X L
+;The asynchronous table
+; D  SD  CD  CP CurrentFlagState  NextFlagState
+ X H X X X H
+ X X H X X L
+ X X X X X F0
+;The output table
+/table 1
+; D  SD  CD  CP Flag   O1  ~O1
+  X   X   X     X     X     F0   ~F0
+")
```

Since the Flip-Flop only has two states, a single flag, or bit, is required to represent them. We represent the state when output is LOW using the LOW value and the state when the output is HIGH using a HIGH value. When the signal “CP” makes a LOW->HIGH transition, the synchronous truth table is evaluated, updating the next state flags (i.e the contents of F0). The output truth table is evaluated afterwards, updating the output variables. Note that in this example, the values that were chosen to represent the states correspond directly to the output values. As such, only a single entry, which copies the value of state flags to the outputs variables regardless of the input, is required in the output table.

When any of the non-clocked signals make a transition, the asynchronous table and the output table are evaluated in a similar fashion. Note that when the Set or Clear signals are not individually HIGH, the state is maintained by the copying of the present state flag to the next state flag.

## /delay

The `/delay` command is used to describe the combined propagation and rise/fall delay between an event on a particular input and an event on the affected output. The delay statement is followed by an integer that represents the number of rows in the delay table that follows. The delay table has four columns: input signal name, output signal name, rise delay, and fall delay.

```
aU1  [in1 in2] [out] myAND
.model myAND d_chip(behaviour= "
+;Simple And gate
+/inputs  A B
+/outputs Y
+/table 2
+;A B Y
+H H H
+X X X
+/delay 2
;input  output  rise_delay  fall_delay
A Y 30n 20n
B Y 20n 10n
+")
```

## /wires

The `/wires` command is used define internal nodes which are used to interconnect modules. It has the following format:

```
/WIRES wire_name1 wire_name2 …
```

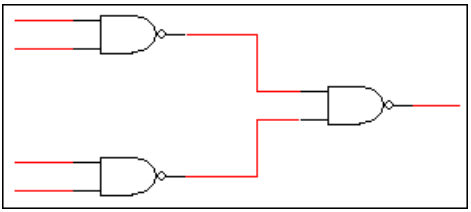
Because `/wires` is used in the context of modules, a practical example is shown in the `/module` command section.

## /module/instance

The `/module` command is used to define self-contained units which may internally contain all of the behaviour commands described about. This is useful in building hierarchical circuits and in simplifying designs that reuse a block of logic. Modules are analagous in function to subcircuits in SPICE. The `/module` command must be accompanied by an `/endmodule` command which signifies the end of scope.

The `/instance` statement is used to instantiate the modules into the logic.

Suppose you would like to describe the logical behavior of the following circuit.



One convenient way of doing so is to create a module for the NAND gate and instantiate it three times, as shown below. Note that because we need to create internal nodes, we must use the `/wires` command.

```
aU1 [in1 in2 in3 in4] [out] logic
.model logic d_chip(behaviour= "
+/inputs A B C D
+/outputs Y
+/wires node1 node2

+/module NAND
+/inputs  in1 in2
+/outputs out
+/table 2
+L L H
+X X L
+/endmodule

+/instance NAND A B node1
+/instance NAND C D node2
+/instance NAND node1 node2 Y
+")
```

## Additional D_Chip Device Functionality

- Input Variable Names

  The names of the input variables and state flags can be used directly in the fields corresponding to the output variables. For example, the asynchronous table of the Flip-Flop example given in the `/clock` section, can be modified to the following:

  ```
  +/clock CP + 1 1 3
  +;SYNC
  +; D  SD  CD  CP CurrentFlagState  NextFlagState
  +X X X X X D
  ```

  Note that instead of specifying table entries corresponding to specific values of the signal of ‘D’, simply copy it over by name.

- Omitting white space

  When specifying input columns in tables, it is valid to omit white space by concatenating the input values into one string. For example:

```
+;SYNC
+; D  SD  CD  CP CurrentFlagState  NextFlagState
+X X X X X D
```

  is the same as

```
+;SYNC
+; D  SD  CD  CP CurrentFlagState  NextFlagState
+XXXXX D
```

  is the same as

```
+;SYNC
+; D  SD  CD  CP CurrentFlagState  NextFlagState
+XX XXX D
```

  Many models in Multisim's database take advantage of this functionality.

- Flag value increment/decrement

  To ease the description of state transition tables in which the states transition in an orderly manner, such as those of a counter, the 'F+n' and 'F-n' notation can be used to assign the next state values. A next state value assigment of 'F+n', increments the current state flag value (i.e toggles bit) if a carry-over bit is passed in and sends a carry over bit to the next flag assigment in case of overflow. If the 'F+n' notation is used in a leading flag (i.e no carry over can be passed into it) such as F0, the flag is always incremented by one.

  In other words, the 'F+n' notation allows you to set the next state value by forming a binary number of present state flags,FnFn-1..F1F0, adding a one to it, and assigning each bit of the result to the respective flag.

  The 'F-n' notation is analogous to 'F+n'. It is used to decrement the next state flags.

  Let's consider the following 3-bit synchronous binary counter as an example.

```
.MODEL 3bit_counter d_chip
+    ( behaviour= "
+/inputs  MR  CLK
+/outputs O0 O1 O2
+/clock CLK + 3 1 2
+;SYNC
+;MR CLK   Present State Flags    Next State Flags
+ X   X    XXX                   F+0 F+1 F+2
+;ASYNC
+;MR CLK   Present State Flags    Next State Flags
+ L   X    XXX                   F0  F1  F2
+ H   X    XXX                   L   L   L
+/TABLE 1
+;MR CLK   Present State Flags    O0  O1  O2
+ X   X    XXX                   F0  F1  F2
+")
```

On every positive clock edge, the 4-bit number represented by the present state flags is incremented by one, the result is assigned to the next state flags, and the next state flags are then directly assigned to the output variables. For instance, if the current state is F0=1 F1=0 F2=0, the next state is F0=0 F1=1 F=0.

Note that without this functionality, we would have had to explicitly specify every possible present state condition and assign a corresponding next state value, as shown in the example below:

```
.MODEL 3bit_counter d_chip
+   ( behaviour= "
+/inputs  MR  CLK
+/outputs O0 O1 O2
+/clock CLK + 3 8 2
+;SYNC
+;MR CLK   Present State Flags     Next State Flags
+ X   X    LLL                    HLL
+ X   X    HLL                    LHL
+ X   X    LHL                    HHL
+ X   X    HHL                    LLH
+ X   X    LLH                    HLH
+ X   X    HLH                    LHH
+ X   X    LHH                    HHH
+ X   X    HHH                    LLL
+;ASYNC
+;MR CLK   Present State Flags     Next State Flags
+ L   X    XXX                    F0  F1  F2
+ H   X    XXX                    L   L   L
+/TABLE 1
+;MR CLK   Present State Flags     O0  O1  O2
+ X   X    XXX                    F0  F1  F2
+")
```

- Toggle

  It’s possible to toggle the value of an input variable or a state variable(flag) and use the result for an assignment. For example:

  ```
  +/clock CLK + 1 1 2
  +;SYNC
  +;MR ~CP   Present State Flags      Next State Flags
  + X   X    X                      ~F0
  ```

  Note that it is valid for input variable names to contain the ‘~’ character. As such, the string “~somename” does not imply a toggle if “~somename” is an actual variable name. Using the ~ character to toggle these types of variables is disallowed.

# D-Latch

D-Latch instance declaration syntax:

```
Axxxx Data Enable Set Reset Q notQ MyModel
```

D-Latch model definition syntax:

```
.MODEL MyModel d_latch(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| DATA_DELAY | Data Propagation delay. | s | 1e-9 |
| ENABLE_DELAYS | Enable propagation delay. | s | 1e-9 |
| SET_DELAYS | Set propagation delay. | F | 1e-9 |
| RESET_DELAYS | Reset propagation delay. | s | 1e-9 |
| IC | Initial Condition on the output (in case enable=LOW) 0 – LOW, 1-HIGH, 2-UNKNOWN. | — | 0 |

### Description

This is D-Latch with asyncronous set and reset. The Enable signal is also known as the Clock signal. It has the following behavior:

| Data | Enable | Set | Reset | Q |
|---|---|---|---|---|
| D | 1 | 0 | 0 | D |
| — | 0 | 0 | 0 | Last State |
| — | — | 1 | 0 | 1 |
| — | — | 0 | 1 | 0 |
| — | — | 1 | 1 | X |

Where '-' denotes Don't Care, '1' denotes HIGH, '0' denotes LOW, and 'X' denotes UNKNOWN.

### Example

```
aU4 1
+     2
+     5
+     6
+    dU4.Q
+     U4_OPEN_notQ Latch
.model latch d_latch(rise_delay=10n fall_delay=10n)
```

# SR-Latch

SR-Latch instance declaration syntax:

```
Axxxx S R Enable Set Reset Q notQ MyModel
```

D-Latch model definition syntax:

```
.MODEL MyModel d_srlatch(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| S_DELAY | Synchronous Set Propagation delay. | s | 1e-9 |
| R_DELAY | Synchronous Reset Propagation delay. | s | 1e-9 |
| ENABLE_DELAY | Enable propagation delay. | s | 1e-9 |
| SET_DELAY | Set propagation delay. | F | 1e-9 |
| RESET_DELAY | Reset propagation delay. | s | 1e-9 |
| IC | Initial Condition on the output (in case enable=LOW)<br>0 – LOW, 1-HIGH, 2-UNKNOWN. | — | 0 |
| RISE_DELAY | Rise Delay. | s | 1e-9 |
| FALL_DELAY | Fall Delay. | s | 1e-9 |

### Description

This is an SR-Latch with asyncronous set and reset. The S and the R signal are synchronous with respect to Enable. The Enable signal is also known as the Clock signal. The device has the following behavior:

| S | R | Enable | Set | Reset | Q |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | Last State |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | X |
| — | — | 0 | 0 | 0 | Last State |
| — | — | — | 1 | 0 | 1 |
| — | — | — | 0 | 1 | 0 |
| — | — | — | 1 | 1 | X |

Where ‘-’ denotes Don’t Care, ‘1’ denotes HIGH, ‘0’ denotes LOW, and ‘X’ denotes UNKNOWN.

**Example**

```
aU4 1
+     2
+     5
+     6
+    dU4.Q
+     U4_OPEN_notQ Latch
.model latch d_latch(rise_delay=10n fall_delay=10n)
```

# D Flip-Flop

D flip-flop instance declaration syntax:

```
Axxxx Data CLK Set Reset Q notQ MyModel
```

D flip-flop model definition syntax:

```
.MODEL MyModel d_dff(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| DATA_DELAY | Data Propagation delay. | s | 1e-9 |
| CLK_DELAY | Enable propagation delay. | s | 1e-9 |
| SET_DELAY | Set propagation delay. | F | 1e-9 |
| RESET_DELAY | Reset propagation delay. | s | 1e-9 |
| IC | Initial Condition on the output (in case enable=LOW) 0 – LOW, 1-HIGH, 2-UNKNOWN. | — | 0 |
| RISE_DELAY | Rise Delay. | s | 1e-9 |
| FALL_DELAY | Fall Delay. | s | 1e-9 |

**Description**

| Data | CLK | Set | Reset | Q |
|---|---|---|---|---|
| D |  | 0 | 0 | D |
| — | — | 0 | 0 | Last State |
| — | — | 1 | 0 | 1 |
| — | — | 0 | 1 | 0 |
| — | — | 1 | 1 | X |

Where  denotes rising edge, '-' denotes Don't Care, '1' denotes HIGH, '0' denotes LOW, and 'X' denotes UNKNOWN.

**Example**

```
aU4 1
+     2
+     5
+     6
+    dU4.Q
+     U4_OPEN_notQ D_FF__TIL
.model D_FF__TIL d_dff(rise_delay=10n fall_delay=10n)
```

# Toggle Flip-Flop

Toggle Flip-Flop instance declaration syntax:

```
Axxxx T CLK Set Reset Q notQ MyModel
```

Toggle Flip-Flop model definition syntax:

```
.MODEL MyModel d_tff(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| T_DELAY | Toggle Propagation delay. | s | 1e-9 |
| CLK_DELAY | Clock propagation delay. | s | 1e-9 |
| SET_DELAY | Set propagation delay. | F | 1e-9 |
| RESET_DELAY | Reset propagation delay. | s | 1e-9 |
| IC | Initial Condition on the output (in case enable=LOW) 0 – LOW, 1-HIGH, 2-UNKNOWN. | — | 0 |
| RISE_DELAY | Rise Delay. | s | 1e-9 |
| FALL_DELAY | Fall Delay. | s | 1e-9 |

**Description**

| T | CLK | Set | Reset | Q |
|---|---|---|---|---|
| 1 |  | 0 | 0 | ~(Last State) |
| 0 |  | 0 | 0 | Last State |
| — | — | 1 | 0 | 1 |
| — | — | 0 | 1 | 0 |
| — | — | 1 | 1 | X |

Where  denotes rising edge, '-' denotes Don't Care, '1' denotes HIGH, '0' denotes LOW, and 'X' denotes UNKNOWN.

**Example**

```
aU4 1
+     2
+     5
+     6
+    dU4.Q
+     U4_OPEN_notQ D_FF__TIL
.model D_FF__TIL d_tff(rise_delay=10n fall_delay=10n)
```

# JK Flip-Flop

SR flip-flop instance declaration syntax:

```
Axxxx J K CLK Set Reset Q notQ MyModel
```

Toggle flip-flop model definition syntax:

```
.MODEL MyModel d_jkff(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| J_DELAY | J Propagation delay. | s | 1e-9 |
| K_DELAY | K Propagation delay. | s | 1e-9 |
| CLK_DELAY | Clock propagation delay. | s | 1e-9 |
| SET_DELAY | Set propagation delay. | F | 1e-9 |
| RESET_DELAY | Reset propagation delay. | s | 1e-9 |
| IC | Initial Condition on the output (in case enable=LOW) 0 – LOW, 1-HIGH, 2-UNKNOWN. | — | 0 |
| RISE_DELAY | Rise Delay. | s | 1e-9 |
| FALL_DELAY | Fall Delay. | s | 1e-9 |

**Description**

This is a JK flip-flop with asynchronous set and reset.

| J | K | CLK | Set | Reset | Q |
|---|---|---|---|---|---|
| 0 | 0 |  | 0 | 0 | Last State |
| 1 | 0 |  | 0 | 0 | 1 |

| J | K | CLK | Set | Reset | Q |
|---|---|---|---|---|---|
| 0 | 1 |  | 0 | 0 | 0 |
| 1 | 1 |  | 1 | 0 | ~(Last State) |
| — | — | — | 1 | 0 | 1 |
| — | — | — | 0 | 1 | 0 |
| — | — | — | 1 | 1 | X |

Where  denotes rising edge, '-' denotes Don't Care, '1' denotes HIGH, '0' denotes LOW, and 'X' denotes UNKNOWN.

**Example**

```
aU4 1
+     2
+     5
+     6
+    dU4.Q
+     U4_OPEN_notQ D_FF__TIL
.model D_FF__TIL d_tff(rise_delay=10n fall_delay=10n)
```

# Constant Source

Constant Source instance declaration syntax:

```
Axxxx out MyModel
```

OR gate model definition syntax:

```
.MODEL MyModel d_constsource(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| STATE | Output Logic level: 1 - HIGH, 0 - LOW, -1 - UNKNOWN. The Strength is always STRONG. | — | 1 |

**Description**

This device is used to generate a constant digital signal.

**Example**

```
A1 out dsig
.model dsig d_constsource(state=0)
```

# Clock Source

Constant Source instance declaration syntax:

```
Axxxx out MyModel
```

Constant Source model definition syntax:

```
.MODEL MyModel d_clock(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| FREQUENCY | Frequency of oscillation. | Hz | 1000 |
| DUTY | Duty Cycle ratio. HIGH time=DUTY/FREQUENCY. | — | 0.5 |
| DELAY | Time before oscillation begins. | s | 0 |

### Description

This device is used to generate a periodic HIGH/LOW pulse signal.

### Example

A2 out_clock DigClock
.model DigClock d_clock (frequency=100k duty=0.5 delay=5m)

# Arbitrary Digital Source

Arbitrary Digital Source instance declaration syntax:

```
Axxxx [out1 <out2>…<outN>] MyModel
```

Arbitrary Digital Source model definition syntax:

```
.MODEL MyModel d_source(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| INPUT_FILE | Full path to file containing signal description. This is a string parameter and must be contained in quotes. | — | — |

### Description

This device is used to generate an arbitrary, multi-output digital signal. The signal description is stored in an external ASCII file and follows a table-like format. The columns in the file are white-space separated.

The first column describes event times when the outputs are allowed to change. The device supports scientific notation suffix characters (e.g 1n, 5.98m).

The subsequent columns describe, for each output specified in the instace line, the signal values at the event times. Signal values are two characters long. The first character is 1, 0, or X which correspond to HIGH, LOW, and UNKNOWN levels respectively. The second character is either S or Z, which correspond to STRONG and HI-IMPEDANCE signal strengths, respectively. The * character is used for comments. Consider the following example:

A2 [digOut1 digOut2] myDIG
.model myDIG d_source (input_file="C:/digital_signal.txt")

where `c:/digital_signal.txt` has the following contents:

```
**time dig_out1 dig_out2
0.0    0s     0s
100n   1s     0s
500n   1s     1s
1.4u   1s     0s
3e-6   0s     1z
```

# Analog to Digital Bridges

Analog to Digital bridges are mixed-mode devices: they contain analog input ports and digital output ports. They are used to interface analog devices to the inputs of digital devices.

**Related Information**
*Ideal A/D Bridge*
*Real A/D Bridge*

## Ideal A/D Bridge

Ideal A/D Bridge instance declaration syntax:

`Axxxx [Ain1 <Ain2>…<AinN>] [Dout1 <Dout2>…<DoutN>] MyModel`

Ideal A/D Bridge model definition syntax:

`.MODEL MyModel adc_bridge(<Model_Parameters>)`

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| IN_LOW | LOW voltage threshold. | v | 0.1 |
| IN_HIGH | HIGH voltage threshold. | v | 0.9 |
| SCHMITT | Flag that indicates whether the input has hysteresis behavior. | — | false |
| RISE_DELAY | Rise Delay. | s | 1e-9 |
| FALL_DELAY | Fall Delay. | s | 1e-9 |

### Description

This device senses the voltage on the input port and generates correponding digital signal values on the output port. The device acts as an ideal voltage sensor, presenting no load to the input. The input voltage is measured with respect to SPICE ground (node 0).

### Example

```
aADC1in [1] [2] ADC1
.MODEL ADC1 adc_bridge (in_low= 1 in_high = 3.5 schmitt=true)
```

## Real A/D Bridge

Real A/D Bridge instance declaration syntax:

```
Axxxx [Ain1 <Ain2>…<AinN>] [Dout1 <Dout2>…<DoutN>] power ground
MyModel
```

Real A/D Bridge model definition syntax:

```
.MODEL MyModel adc_bridge(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| TECHNOLOGY | Selects the logic technology. | — | 0 |
| SCHMITT | Flag that indicates whether the input has hysteresis behavior. | — | false |
| RISE_DELAY | Rise Delay. | s | 1e-9 |
| FALL_DELAY | Fall Delay. | s | 1e-9 |

### Description

This device is designed for internal use for a very specific application: interacing the inputs of digital components that belong a specific logic technology to analog components. This device is crucial to real pin models.

This device senses the voltage on the input port and generates a correponding digital signal values on the output port. The device acts as an ideal voltage sensor, presenting zero load to the input (the load is simulated by a separate, purely analog device). However, unlike in the Ideal A/D Bridge, the switching thresholds are determined by the power and ground voltages rather than by the hardcoded values. Additionally, the input voltage is measured with respect to the ground connection node, rather than SPICE ground.

### Example

aADC1 [1] 2 3 4 ADC1
.model ADC1 adc_bridge_real(technology=1)

# Digital to Analog Bridges

Digital to Analog bridge are mixed-mode devices with digital input ports and analog output ports. They are used to interface ouputs of digital devices to analog devices.

**Related Information**

*Ideal D/A Bridge*
*Real D/A Bridge*

## Ideal D/A Bridge

Ideal D/A Bridge instance declaration syntax:

Axxxx [Din1 <Din2>…<DinN>] [Aout1 <Aout2>…<AoutN>] MyModel

Ideal D/A Bridge model definition syntax:

```
.MODEL MyModel dac_bridge(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| OUT_LOW | Output Voltage for LOW. | v | 0 |
| OUT_HIGH | Output Voltage for HIGH. | v | 1 |
| OUT_UNDEF | Output Voltage for UNKNOWN. | v | 0.5 |
| T_RISE | Analog Rise Time. | s | 1e-9 |
| T_FALL | Analog Fall Time. | s | 1e-9 |

**Description**

This device generates a voltage on the output according to the digital signal on the input node. The output is seen as an ideal voltage source with respect to SPICE ground (node 0).

**Example**

```
aDAC1in  [1]  [2] aDAC1
.MODEL aDAC1 dac_bridge (out_low= 0 out_high = 5.0 out_undef = 2.5)
```

## Real D/A Bridge

Real D/A Bridge instance declaration syntax:

```
Axxxx [Din1 <Din2>…<DinN>] [Aout1 <Aout2>…<AoutN>] power ground
MyModel
```

Real D/A Bridge model definition syntax:

```
.MODEL MyModel dac_bridge_real(<Model_Parameters>)
```

Model definition parameters:

| Parameter Name | Parameter Description | Units | Default |
|---|---|---|---|
| TECHNOLOGY | Selects the logic technology. | — | 0 |
| OPEN_C | Flag that indicates whether the output has Open-collector or Open-drain behavior. | — | false |

**Description**

This device is designed for internal use for a very specific application: interacing the outputs of digital components that belong to a specific logic technology to analog components. This device is crucial to real pin models.

The device changes its output chacteristic depending on the value of the digital input signal. However, unlike the output of Ideal A/D Bridge device, the output of this device is seen as a source with a non-linear resistance whose I/V behavior closely matches the I/V behaviour of the logic technology designed by the TECHNOLOGY parameter.

**Example**

aADC1 [1] 2 3 4 ADC1
.model ADC1 adc_bridge_real(technology=1 open_c=false)

# Device Temperature Parameters

A device is temperature-dependent if it has at least one parameter that is temperature-dependent.

The temperature-dependent devices are:

- Resistor
- Capacitor
- Inductor
- Diode
- MOSFET (all levels)
- BJT (all levels)
- JFET
- MESFET
- Arbitrary sources

Refer to the documentation for each device for the list of all temperature-dependent parameters and the temperature equations affecting them.

Multisim includes parameterizable components with models that are based on a single such device (for example, the BJT_NPN component in the TRANSISTORS_VIRTUAL family is a wrapper for the N-type BJT device) so that you can easily access and modify the temperature parameters without having to create your own components.

## Instance Temperature Parameters

The operating temperature can be specified on the instance statement of the temperature-dependent device using the TEMP parameter.

**Example**

R1 node1 node2 47k TEMP=35

D1 node3 node4 myDiode TEMP=38

This setting overrides the global operating temperature setting.

## Model Temperature Parameters

Both the operating and nominal temperature can be specified on the model statement of temperature-dependent devices using variety of parameters outlined below.

**Note** The nominal temperature is the temperature at which the element's parameters are measured. It is typically set by the model designer and not the circuit designer.

| Parameter | Device Simulation Temperature | Description |
|---|---|---|
| TNOM | — | Nominal temperature.<br>Example:<br>R1 44 50 res1 4.7k TEMP=35<br>.model res1 R (TNOM=30) |
| T_MEASURED | — | Nominal temperature. Same as TNOM. Cadence® PSpice® compliant. |
| T_ABS | T_ABS | Operating temperature. This overrides any settings of the operating temperature from the instance line and from the global setting. Cadence® PSpice® compliant.<br>Example:<br>D1 node1 node2 Mydiodes<br>D2 node2 node3 Mydiodes<br>.model Mydiodes d( T_ABS=41 T_MEASURED=0 |

| Parameter | Device Simulation Temperature | Description |
|---|---|---|
| T_REL_GLOBAL | <global temperature> + T_REL_GLOBAL | Operating temperature relative to the global operating temperature. Cadence® PSpice® compliant.<br>Example:<br>.MODEL PNP_MOS<br>(T_REL_GLOBAL=-17) |
| T_REL_LOCAL | <T_ABS in AKO model> + T_REL_LOCAL | Operating temperature relative to T_ABS specified inside the AKO model. Cadence® PSpice® compliant.<br>Example:<br>.MODEL PNP MOD1 T_ABS=60<br>.MODEL MOD2 AKO:MOD1 PNP T_REL_LOCAL=10 |

The Cadence® PSpice® temperature parameters (T_MEASURED, T_ABS, T_REL_LOCAL, T_REL_GLOBAL) can only be set for the following devices:

- Resistor
- Capacitor
- Inductor
- BJT (level 1)
- JFET
- MESFET
- MOSFET (levels 1-3)
- Diode

**Note** The Cadence® PSpice® temperature parameters have precedence over the TEMP instance and TNOM model parameters.

**Note** When using these parameters, a maximum of one device temperature customization (T_ABS, T_REL_LOCAL, T_REL_GLOBAL) can co-exist with the T_MEASURED parameter.

# XSPICE Syntax Reference

**Related Information**
*XSPICE Code Model*

## XSPICE Code Model

XSPICE syntax for a terminal:

<%TerminalType> <(> nodename <, referencenodename> <)>

<%TerminalType> [ arraynodename1 <, arraynodename2 <, ...>> ]

XSPICE Terminal types:

| Symbol | Description |
|---|---|
| %D | Digital terminal. |
| %V | Voltage terminal. |
| %I | Current terminal. |
| %G | Voltage in/current out terminal. |
| %H | Current in/voltage out terminal. |
| %VD | Differential voltage terminal. |
| %ID | Differential current terminal. |
| %GD | Differential voltage in/current out terminal. |
| %HD | Differential current in/voltage out terminal. |
| %VNAM | Named vsource “terminal”. |

XSPICE code model instance line syntax:

Axxx xspiceterminal1 < xspiceterminal2 <. . .>> Model <*Instance_Parameters...*>

XSPICE code-model model definition syntax:

`.MODEL` mymodelname `CODEMODELNAME` ( <*Model_Parameters...*> )

Please refer to XSPICE documentation at:
`http://users.ece.gatech.edu/~mrichard/Xspice/` for further details.

# Unusual Forms of Device Syntax

The syntax of the various devices listed above are given with the most common forms. However, there are several other variations which are also acceptable for compatibility reasons. In general, the following rules can be used:

- Brackets that are not part of a mathematical expression are optional.
- Commas between parameters are optional (except in mathematical functions).
- Equal signs on certain parameters are optional; they can be left out of places where they would normally go and added in after parameters names that don't normally need them. The parameters that are optional are V, I, DC, AC, PWL, SIN, TABLE, and so on, in sources.
- Parameter names that are implied can also be stated explicitly (for example, "Resistance" on a resistor.)
- The order of explicitly stated parameters doesn't matter.
- Except in mathematical expressions, whitespace doesn't matter.
- Putting curly brackets, {}, around expressions is recommended but is not required in most cases. Inside expressions curly brackets, {}, can be used like parenthesis, ().

  In sources of the form similar to SIN and PWL, the brackets don't matter—these can be set with a line like SIN = 1 2 3 … in the same way that any other parameter, or the brackets, can be used to make logical pairs for PWL = (1,2),(3,4),(5,6).

**Examples**

Thus the following statements are acceptable input to the Multisim SPICE simulator:

```
R1 1 0 RESISTANCE=1k

V1 1 0 AC 2, 1 DC = 10

H1 1 2 POLY 1 V2 3.40k

I2 1 0 PWL =  0 0 1.3  2.0, 5.0,   5.0

E2 5 8 TABLE V(4)**2 0.0, 0.0, 0.5,0.5, 1.0,   0.0
```

# Compatibility Modes

Some other forms of SPICE have slight differences which cannot be automatically detected by Multisim. In order for Multisim to correctly handle these cases it is necessary to add the following special command:

.SYNTAX `mode`

where `mode` can be PS for Cadence® PSpice®, XS for XSPICE, and MS for Multisim.

When this command is used, all other elements of the netlist that occur below it (including nested `.subckt` definitions) will be interpreted with special compatibility mode until the end of the current subckt. The default compatibility mode is MULTISIM unless otherwise specified.

Default mode occurs when no .SYNTAX statements are present:

- The mathematical function log is the natural logarithm, except in 'B' arbitrary sources where it is the base 10 logarithm.

MS and XS mode are currently the same:

- The mathematical function log is the base 10 logarithm.
- No absolute value of the base is taken in the mathematical operator **.
- Mathematical expressions do not need to be surrounded by `{}`, however this is still recommended for clarity.

The PS mode has a few differences.

- The mathematical function log is the natural logarithm.
- The mathematical operator `**` is the same as the function PWR, which is equivalent to `abs(x)**y` in Multisim compatibility mode.
- The mathematical operator ^ is xor whereas in Multisim compatibility mode it is another way of writing `x**y`.
- The step function `STEP(x)` or `STP(x)` or `U(x)` is defined as `if(x>=0,1,0)`.
- Mathematical expressions other than numbers must be surrounded by {}.
- Equal signs are optional between all parameters and their values.
- When the character `*` is not inside {} it marks the start of a comment, even if it is in the middle of a line. Therefore in following example R1 will have a value of 1k in Cadence® PSpice® compatibility mode, but 2k in all other modes.

  R1 1 2 1k*2