B. Arbitrary behavioral voltage or current sources.

Symbol names: BV, BI

```
Syntax: Bnnn n001 n002 V=<expression> [ic=<value>]
+ [tripdv=<value>] [tripdt=<value>]
+ [laplace=<expression> [window=<time>]
+ [nfft=<number>] [mtol=<number>]]
```

```
Bnnn n001 n002 I=<expression> [ic=<value>]
+ [tripdv=<value>] [tripdt=<value>] [Rpar=<value>]
+ [laplace=<expression> [window=<time>]
+ [nfft=<number>] [mtol=<number>]]
```

The first syntax specifies a behavioral voltage source and the next is a behavioral current source. For the current source, a parallel resistance may be specified with the Rpar instance parameter.

Tripdv and tripdt control step rejection. If the voltage across a source changes by more than tripdv volts in tripdt seconds, that simulation time step is rejected.

Expressions can contain the following:

o Node voltages, e.g., V(n001)

o Node voltage differences, e.g., V(n001, n002)

o Circuit element currents; for example, I(S1), the current through switch S1 or Ib(Q1), the base current of Q1. However, it is assumed that the circuit element current is varying quasi-statically, that is, there is no instantaneous feedback between the current through the referenced device and the behavioral source output. Similarly, any ac component of such a device current is assumed to be zero in a small signal linear .AC analysis.

o The keyword, "time" meaning the current time in the simulation.

o The keyword, "pi" meaning 3.14159265358979323846.

o The following functions:

Function Name	Description
abs(x)	Absolute value of x
absdelay(x,t[,tmax])	x delayed by t. Optional max delay notification tmax.
acos(x)	Real part of the arc cosine of x, e.g., acos(-5) returns 3.14159, not 3.14159+2.29243i
arccos(x)	Synonym for acos()

acosh(x)	Real part of the arc hyperbolic cosine of x, e.g., acosh(.5) returns 0, not 1.0472i
asin(x)	Real part of the arc sine of x, asin(-5) is -1.57080, not -1.57080+2.29243i
arcsin(x)	Synonym for asin()
asinh(x)	Arc hyperbolic sine
atan(x)	Arc tangent of x
arctan(x)	Synonym for atan()
atan2(y, x)	Four quadrant arc tangent of y/x
atanh(x)	Arc hyperbolic tangent
buf(x)	1 if x > .5, else 0
ceil(x)	Integer equal or greater than x
cos(x)	Cosine of x
cosh(x)	Hyperbolic cosine of x
ddt(x)	Time derivative of x
delay(x,t[,tmax])	Same as absdelay()
exp(x)	e to the x
floor(x)	Integer equal to or less than x
hypot(x,y)	sqrt(x**2 + y**2)
idt(x[,ic[,a]])	Integrate x, optional initial condition ic, reset if a is true.
idtmod(x[,ic[,m[,o]]])	Integrate x, optional initial condition ic, reset on reaching modulus m, offset output by o.
if(x,y,z)	If x > .5, then y else z
int(x)	Convert x to integer
inv(x)	0. if x > .5, else 1.
limit(x,y,z)	Intermediate value of x, y, and z
ln(x)	Natural logarithm of x
log(x)	Alternate syntax for ln()
log10(x)	Base 10 logarithm
max(x,y)	The greater of x or y
min(x,y)	The smaller of x or y
pow(x,y)	Real part of x**y, e.g., pow(-1,.5)=0, not i.
pwr(x,y)	abs(x)**y

pwrs(x,y)	sgn(x)*abs(x)**y
rand(x)	Random number between 0 and 1 depending on the integer value of x.
random(x)	Similar to rand(), but smoothly transitions between values.
round(x)	Nearest integer to x
sdt(x[,ic[,assert]])	Alternate syntax for idt()
sgn(x)	Sign of x
sin(x)	Sine of x
sinh(x)	Hyperbolic sine of x
sqrt(x)	Square root of x
table(x,a,b,c,d,...)	Interpolate a value for x based on a look up table given as a set of pairs of points.
tan(x)	Tangent of x.
tanh(x)	Hyperbolic tangent of x
u(x)	Unit step, i.e., 1 if x > 0., else 0.
uramp(x)	x if x > 0., else 0.
white(x)	Random number between -.5 and .5 smoothly transitions between values even more smoothly than random().
!(x)	Alternative syntax for inv(x)
~(x)	Alternative syntax for inv(x)

o The following operations, grouped in reverse order of precedence of evaluation:

Operand	**Description**
&	Convert the expressions to either side to Boolean, then AND.
|	Convert the expressions to either side to Boolean, then OR.
^	Convert the expressions to either side to Boolean, then XOR.
>	True if expression on the left is greater than the expression on the right, otherwise false.
<	True if expression on the left is less than the expression on the right, otherwise false.

>=	True if expression on the left is less than or equal the expression on the right, otherwise false.
<=	True if expression on the left is greater than or equal the expression on the right, otherwise false.
+	Floating point addition
-	Floating point subtraction
*	Floating point multiplication
/	Floating point division
**	Raise left hand side to power of right hand side. Only the real part is returned, e.g., -1**1.5 gives zero not i.
!	Convert the following expression to Boolean and invert.

True is numerically equal to 1 and False is 0. Conversion to Boolean converts a value to 1 if the value is greater than 0.5, otherwise the value is converted to 0.

Note that LTspice uses the caret character, ^, for Boolean XOR and "**" for exponentiation. Also, LTspice distinguishes between exponentiation, x**y, and the function pwr(x,y). Some 3rd party simulators have an incorrect implementation of behavioral exponentiation, evaluating -3**3 incorrectly to 27 instead of -27, presumably in the interest of avoiding the problem of exponentiating a negative number to a non-integer power. LTspice handles this issue by returning the real part of the result of the exponentiation. E.g., -2**1.5 evaluates to zero which is the real part of the correct answer of 2.82842712474619i. This means that when you import a 3rd party model that was targeted at a 3rd party simulator, you may need to translate the syntax such as x^y to x**y or even pwr(x,y).

If an optional Laplace transform is defined, that transform is applied to the result of the behavioral current or voltage. The Laplace transform must be a function solely of s. The Boolean XOR operator, ^, is understood to mean exponentiation, **, when used in a Laplace expression. The frequency response at frequency f is found by substituting s with sqrt(-1)*2*pi*f. The time domain behavior is found from the sum of the instantaneous current(or voltage) with the convolution of the history of this current(or voltage) with the impulse response. Numerical inversion of a Laplace transfer function to the time domain impulse response is a potentially compute-bound process and a topic of current numerical research. In LTspice, the impulse response is found from the FFT of a discrete set points in frequency domain response. This process is prone to the usual artifacts of FFT's such as spectral leakage and picket fencing that is common to discrete FFT's. LTspice uses a proprietary algorithm that exploits that it has an exact analytical expression for the frequency domain response and chooses points and windows to cause such artifacts to diffract precisely to zero. However, LTspice must guess an appropriate frequency range and resolution. It is recommended that the LTspice first be allowed to make a guess at this. The length of the window and number of FFT data points used will be reported in the .log file. You can then adjust the algorithm's choices by explicitly setting nfft and window length. The reciprocal of the value of the window is the frequency resolution. The value of nfft times this resolution is the highest frequency considered. Note that the convolution of the impulse response with the behavioral source is also potentially a compute bound process.