Waveform Arithmetic:

*A single member of a parametrized plot family may be selected for display by suffixing a voltage or current within an expression with "@n" where n is the ordinal step number, e.g. V(1)@2, I(S3)@4.

The following operations, grouped in order of precedence of evaluation, are available (r = real data only, c = complex data only, * = undocumented):

```
* Symbol | Operation
 --------+-----------------------------------------------------
     !   r convert succeeding expression to Boolean then invert
    **   | floating point exponentiation
     ^   c floating point exponentiation
 --------+-----------------------------------------------------
     /   | floating point division
     *   | floating point multiplication
*    %   r floating point modulus
 --------+-----------------------------------------------------
     -   | floating point subtraction
     +   | floating point addition
 --------+-----------------------------------------------------
*   ==   r true if preceding and succeeding expressions are
         |   equal, otherwise false
    >=   r true if preceding expression is greater than or
         |   equal to succeeding expression, otherwise false
    <=   r true if preceding expression is less than or
         |   equal to succeeding expression, otherwise false
     >   r true if preceding expression is greater than
         |   succeeding expression, otherwise false
     <   r true if preceding expression is less than
         |   succeeding expression, otherwise false
 --------+-----------------------------------------------------
     ^   r convert adjacent expressions to Boolean then XOR
     |   r convert adjacent expressions to Boolean then OR
     &   r convert adjacent expressions to Boolean then AND
```

For Boolean operations True is 1 and False is 0. Boolean conversions return True if <expression> evaluates to greater than .5, else False.

The following keywords (global variables and constants) are available:

```
      Name     |     Value      | Description
 --------------+----------------+---------------------------------
      time     |    variable    | time in seconds (real data only)
      freq     |    variable    | freq in Hertz   (cmplx data only)
*       w      |    variable    | freq in radians (cmplx data only)
*       i      |    sqrt(-1)    | imaginary unity (cmplx data only)
        e      |  2.71828182846 |
       pi      |  3.14159265359 |
*       c      |  2.99792  e+08 | speed of light in meters/second
  k or *boltz  |  1.3806503e-23 | Boltzmann constant
     planck    |  6.62620  e-34 | Planck's constant
 q or *echarge |  1.6021765e-19 | charge of an electron
*    kelvin    | -2.73150  e+02 | absolute zero in degrees C
```

*The following data labels are available (case insensitive):

V, A, Ohm, W (watt), J (joule), s (second), Hz, deg

Waveform Arithmetic (continued):

The difference of two voltages, i.e. V(a)-V(b), can equivalently be written as V(a,b). The following functions are available (r = real data only, c = complex data only, * = undocumented):

```
*Status       Name  | Function
 -------------------+--------------------------------------
             sin(x) | sine
             cos(x) | cosine
             tan(x) | tangent
            asin(x) | arc sine
            acos(x) | arc cosine
            atan(x) | arc tangent
         atan2(y,x) r arc tangent of y/x (four quadrant)
*        hypot(y,x) r hypotenuse: sqrt(x*x+y*y)
            sinh(x) | hyperbolic sine
            cosh(x) | hyperbolic cosine
            tanh(x) | hyperbolic tangent
           asinh(x) | arc hyperbolic sine
           acosh(x) | arc hyperbolic cosine
           atanh(x) | arc hyperbolic tangent
             exp(x) | exponential
   *ln(x) or log(x) | natural logarithm
           log10(x) | base 10 logarithm
             sgn(x) r sign (0 if x = 0)
*           fabs(x) r absolute value
             abs(x) | absolute value
            sqrt(x) | square root
*           cbrt(x) | cube root
*         square(x) | x**2
*          pow(x,y) r x**y
*          pwr(x,y) r abs(x)**y
*         pwrs(x,y) r sgn(x)*abs(x)**y
*          round(x) r round to nearest integer
             int(x) r truncate to integer part of x
           floor(x) r integer equal or less than x
            ceil(x) r integer equal or greater than x
          min(x,y) r the lesser of x or y
          max(x,y) r the greater of x or y
      limit(x,y,z) r equivalent to min(max(x,y),z)
  table(x,x1,y1...) r interpolate y(x) per a lookup table
  or *tbl: x1<x2... r   of x-ordered point pairs
           uramp(x) r x if x > 0, else 0
               u(x) r unit step: 1 if x > 0, else 0
             buf(x) r 1 if x > .5, else 0
    *!(x) or inv(x) r 0 if x > .5, else 1
            rand(x) r 0 < random num < 1 at x sharp steps/sec
*         random(x) r 0 < random num < 1 at x soft steps/sec
*          white(x) r -.5 < ran num < .5 at x smooth steps/sec
*             re(x) c real part
*             im(x) c imaginary part
*             ph(x) c phase
*            mag(x) c magnitude
*             db(x) c magnitude in dB
*          invdb(x) c 10**(x/20)
*              d(x) | dx/dt (hint: disable waveform compression)
```

B. Arbitrary behavioral voltage or current sources.

Symbol names: BV, BI, *BR (arbitrary resistor)

Syntax (* denotes undocumented features):

```
    Bxxx n1 n2 V=<expression>
    + [[ic=<value>] tripdv=<value>] [tripdt=<value>]
    + [Laplace=<func(s)> [window=<time>] [nfft=<num>] [mtol=<num>]]
*   + [[units] Freq=<valuelist> [delay=<value>]]

    Bxxx n1 n2 I=<expression> [Rpar=<value>]
    + [ic=<value>] [tripdv=<value>] [tripdt=<value>]
    + [Laplace=<func(s)> [window=<time>] [nfft=<num>] [mtol=<num>]]
*   + [[units] Freq=<valuelist> [delay=<value>]]
```

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.

The Laplace transform is applied to the result of the behavioral current or voltage signal. The Laplace transform must be a function of s. 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 impulse response obtained from the Fourier transform of the frequency domain response. LTspice must guess an appropriate frequency range and resolution. The response must drop at high frequencies or an error is reported. It is recommended that the LTspice first be allowed to make a guess at this and then check the accuracy by reducing reltol and/or mtol (*default=1) or explicitly setting nfft and the window. The reciprocal of the value of the window is the frequency resolution. The value of nfft times this resolution is the highest frequency considered. For Laplace expressions, ^ signifies exponentiation.

*The transfer function of the Freq circuit element is specified by an ordered list of points of freq(Hz), mag(dB) and phase(deg) as follows: <(f1,m1,p1)[(f2,m2,p2)...]> where f1<f2<f3, etc. The following units specifiers may optionally precede the Freq keyword: "rad"=radians, "mag"=non dB, ("dB" and "deg" return the defaults), "r_i"=real and imaginary in place of magnitude and phase. If a delay value is called out, the phases of the table values are modified to reflect the delay (delay is automatically adjusted to maintain causality in any case).

Expressions can contain the following:

o Node voltages and differences, e.g. V(n1) and V(n1,n2).

o Circuit element currents, e.g. 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.

B. Arbitrary behavioral voltage or current sources (continued).

o The following operations, grouped in order of precedence of evaluation (* denotes undocumented features):

```
  Symbol | Operation
 --------+-----------------------------------------------
     !   | convert succeeding expression to Boolean then
*  or ~  |   invert
    **   | floating point exponentiation
     ^   | floating point exponentiation (Laplace only)
 --------+-----------------------------------------------
     /   | floating point division
     *   | floating point multiplication
 --------+-----------------------------------------------
     -   | floating point subtraction
     +   | floating point addition
 --------+-----------------------------------------------
*   ==   | true if preceding expression is equal to
         |   succeeding expression, otherwise false
    >=   | true if preceding expression is greater than or
         |   equal to succeeding expression, otherwise false
    <=   | true if preceding expression is less than or
         |   equal to succeeding expression, otherwise false
     >   | true if preceding expression is greater than
         |   succeeding expression, otherwise false
     <   | true if preceding expression is less than
         |   succeeding expression, otherwise false
 --------+-----------------------------------------------
     ^   | convert adjacent expressions to Boolean then XOR
     |   | convert adjacent expressions to Boolean then OR
     &   | convert adjacent expressions to Boolean then AND
```

For Boolean operations True is 1 and False is 0. Boolean conversions return True if <expression> evaluates to greater than .5, else False.

o The following keywords (global variables and constants):

```
    Name   |     Value      | Description
 ----------+----------------+-------------------------------
    time   |    variable    | time in seconds
     pi    |  3.14159265359 |
*   boltz  |  1.38062  e-23 | Boltzmann constant
*  planck  |  6.62620  e-34 | Planck's constant
*  echarge |  1.6021765e-19 | charge of an electron
*  kelvin  | -2.73150  e+02 | absolute zero in degrees C
```

o Any user defined parameters or functions. Note that the parameter substitution scheme is generally symbolic, but that when curly braces are encountered, the enclosed expression is evaluated immediately. With functions all parameter substitution evaluation is always done before the simulation begins. For details, refer to the .param and the .func simulator directives defined in Help under the subchapter on Dot Commands.

B. Arbitrary behavioral voltage or current sources (continued).

o The following functions (* denotes undocumented functions):

```
*Status       Name  | Function
 -------------------+---------------------------------------
             sin(x) | sine
             cos(x) | cosine
             tan(x) | tangent
            asin(x) | arc sine
            acos(x) | arc cosine
            atan(x) | arc tangent
         atan2(y,x) | arc tangent of y/x (four quadrant)
         hypot(y,x) | hypotenuse: sqrt(x*x+y*y)
            sinh(x) | hyperbolic sine
            cosh(x) | hyperbolic cosine
            tanh(x) | hyperbolic tangent
           asinh(x) | arc hyperbolic sine
           acosh(x) | arc hyperbolic cosine
           atanh(x) | arc hyperbolic tangent
             exp(x) | exponential
    ln(x) or log(x) | natural logarithm
           log10(x) | base 10 logarithm
             sgn(x) | sign (0 if x = 0)
             abs(x) | absolute value
            sqrt(x) | square root
*         square(x) | x**2
*          pow(x,y) | x**y
*          pwr(x,y) | abs(x)**y
*         pwrs(x,y) | sgn(x)*abs(x)**y
*          round(x) | round to nearest integer
             int(x) | truncate to integer part of x
           floor(x) | integer equal or less than x
            ceil(x) | integer equal or greater than x
           min(x,y) | the lesser of x or y
           max(x,y) | the greater of x or y
       limit(x,y,z) | equivalent to min(max(x,y),z)
          if(x,y,z) | if x > .5 then y else z
  table(x,x1,y1...) | interpolate y(x) per a lookup table
  or *tbl: x1<x2... |   of x-ordered point pairs
           uramp(x) | x if x > 0, else 0.
    *stp(x) or u(x) | unit step, 1 if x > 0, else 0
             buf(x) | 1 if x > .5, else 0
     !(x) or inv(x) | 0 if x > .5, else 1
            rand(x) | 0 < random num < 1 at x sharp steps/sec
          random(x) | 0 < random num < 1 at x soft steps/sec
*          white(x) | -.5 < ran num < .5 at x smooth steps/sec
*            fra(x) | white(x), but 0 if not SMPS steady state
*            ddt(x) v time derivative (v = Verilog-A compatible)
*  idt(x) or sdt(x) v time integral: idt(x[,ic[,assert]])
*                   v   ic=initial constant, assert<>0 resets idt
*         idtmod(x) v wrapping idt: idtmod(x[,ic[,mod[,offset]]])
*                   v   offset < idtmod(x) < offset+mod
*        delay(x,y) v delay of x by y seconds
* absdelay(x,y[,z]) v delay of x by min(y,z) seconds
```