Avant!

Chapter 5

Using Sources and Stimuli

This chapter describes element and model statements for independent sources, dependent sources, analog-to-digital elements, and digital-to-analog elements. It also provides explanations of each type of element and model statement. Explicit formulas and examples show how various combinations of parameters affect the simulation.

The chapter covers the following topics:

Independent Source Elements

Use source element statements to specify either DC, AC, transient, or mixed independent voltage and current sources. Some types of analysis use the associated analysis sources. For example, in a DC analysis, if both DC and AC sources are specified in one independent source element statement, the AC source is taken out of the circuit for the DC analysis. If an independent source is specified for an AC, transient, and DC analysis, transient sources are removed for the AC analysis and DC sources are removed after the performance of the operating point. Initial transient value always overrides the DC value.

Source Element Conventions

Voltage sources need not be grounded. Positive current is assumed to flow from the positive node through the source to the negative node. A positive current source forces current to flow out of the N+ node through the source and into the N- node.

You can use parameters as values in independent sources. Do not identify these parameters using any of the following keywords:

AC	ACI	AM	DC	EXP	PE	PL
PU	PULSE	PWL	R	RD	SFFM	SIN

Independent Source Element Statements

Syntax

```
Vxxx n+ n- <<DC=> dcval> <tranfun> <AC=acmag, <acphase>>
```

or

```
Iyyy n+ n- <<DC=> dcval> <tranfun> <AC=acmag, <acphase>> <M=val>
```

where:

Vxxx independent voltage source element name. Must begin with a “V”, which can be followed by up to 1023 alphanumeric characters.

Iyyy	independent current source element name. Must begin with an “I”, which can be followed by up to 1023 alphanumeric characters.
n+	positive node
n-	negative node
DC	DC source value, *dcval*. The “tranfun” value at time zero overrides the DC value (default=0.0).
tranfun	transient source function (AC, ACI, AM, DC, EXP, PE, PL, PU, PULSE, PWL, R, RD, SFFM, SIN)
AC	indicates source is to be used in an AC small-signal analysis
acmag	AC magnitude
acphase	AC phase (default=0.0)
M	multiplier used for simulating multiple parallel current sources. The current value is multiplied by M (default=1.0).

Examples

```
VX 1 0 5V
VB 2 0 DC=VCC
VH 3 6 DC=2 AC=1,90
IG 8 7 PL(1MA 0S 5MA 25MS)
VCC 10 0 VCC PWL 0 0  10NS VCC  15NS VCC  20NS 0
VIN 13 2 0.001 AC 1 SIN (0 1 1MEG)
ISRC 23 21 AC 0.333 45.0 SFFM (0 1 10K 5 1K)
VMEAS 12 9
```

DC Sources

For a DC source, you can specify the DC current or voltage in different ways:

```
V1 1 0 DC=5V
V1 1 0 5V
I1 1 0 DC=5mA
I1 1 0 5mA
```

The first two examples specify a DC voltage source of 5 V connected between node 1 and ground. The third and fourth examples specify a 5 mA DC current source between node 1 and ground. The direction of current is from node 1 to ground.

AC Sources

AC current and voltage sources are impulse functions used for an AC analysis. Specify the magnitude and phase of the impulse with the AC keyword.

```
V1 1 0 AC=10V,90
VIN 1 0 AC 10V 90
```

The above two examples specify an AC voltage source with a magnitude of 10 V and a phase of 90 degrees. Specify the frequency sweep range of the AC analysis in the .AC analysis statement. The AC or frequency domain analysis provides the impulse response of the circuit.

Transient Sources

For transient analysis, you can specify the source as a function of time. The functions available are pulse, exponential, damped sinusoidal, single frequency FM, and piecewise linear function.

Mixed Sources

Mixed sources specify source values for more than one type of analysis. For example, you can specify a DC source specified together with an AC source and transient source, all of which are connected to the same nodes. In this case, when specific analyses are run, Star-Hspice selects the appropriate DC, AC, or transient source. The exception is the zero-time value of a transient source, which overrides the DC value, and is selected for operating-point calculation for all analyses.

```
VIN 13 2 0.5 AC 1 SIN (0 1 1MEG)
```

The above example specifies a DC source of 0.5 V, an AC source of 1 V, and a transient damped sinusoidal source, each of which are connected between nodes 13 and 2. For DC analysis, the program uses zero source value since the sinusoidal source is zero at time zero.

Star-Hspice Independent Source Functions

Star-Hspice provides the following types of independent source functions:

- Pulse (PULSE function)
- Sinusoidal (SIN function)
- Exponential (EXP function)
- Piecewise linear (PWL function)
- Single-frequency FM (SFFM function)
- Single-frequency AM (AM function)

PWL also comes in a data driven version. The data driven PWL allows the results of an experiment or a previous simulation to provide one or more input sources for a transient simulation.

The independent sources supplied with Star-Hspice permit the designer to specify a variety of useful analog and digital test vectors for either steady state, time domain, or frequency domain analysis. For example, in the time domain, both current and voltage transient waveforms can be specified as exponential, sinusoidal, piecewise linear, single-sided FM functions, or AM functions.

Pulse Source Function

Star-Hspice has a trapezoidal pulse source function, which starts with an initial delay from the beginning of the transient simulation interval to an onset ramp. During the onset ramp, the voltage or current changes linearly from its initial value to the pulse plateau value. After the pulse plateau, the voltage or current moves linearly along a recovery ramp, back to its initial value. The entire pulse repeats with a period per from onset to onset.

Syntax

```
PULSE <(>v1 v2 <td <tr <tf <pw <per>>>>> <)>
```

or

```
PU <(>v1 v2 <td <tr <tf <pw <per>>>>> <)>
```

where:

v1	initial value of the voltage or current, before the pulse onset
v2	pulse plateau value
td	delay time in seconds from the beginning of transient interval to the first onset ramp (default=0.0 and for td < 0.0, td=0.0 is used as well)
tr	duration of the onset ramp, from the initial value to the pulse plateau value (reverse transit time) (default=TSTEP)
tf	duration of the recovery ramp, from the pulse plateau back to the initial value (forward transit time) (default=TSTEP)
pw	pulse width (the width of the pulse plateau portion of the pulse) (default=TSTEP)
per	pulse repetition period in seconds (default=TSTEP, the transient timestep)

Example 1

```
VIN 3 0 PULSE (-1 1 2NS 2NS 2NS 50NS 100NS)
V1 99 0 PU lv hv tdlay tris tfall tpw tper
```

The first example specifies a pulse source connected between node 3 and node 0. The pulse has an output high voltage of 1 V, an output low voltage of -1 V, a delay of 2 ns, a rise and fall time of 2 ns, a high pulse width of 50 ns, and a period of 100 ns. The second example specifies pulse value parameters in the .PARAM statement.

Single pulse

Time	***Value***
0	*v1*
td	*v1*
td + tr	*v2*
td + tr + pw	*v2*
td + tr + pw + tf	*v1*
tstop	*v1*

Intermediate points are determined by linear interpolation.

Note: TSTEP is the printing increment, and TSTOP is the final time.

Example 2

```
file pulse.sp test of pulse
.option post
.tran .5ns 75ns
vpulse 1 0 pulse( v1 v2 td tr tf pw per )
r1 1 0 1
.param v1=1v v2=2v td=5ns tr=5ns tf=5ns pw=20ns
+ per=50ns
.end
```

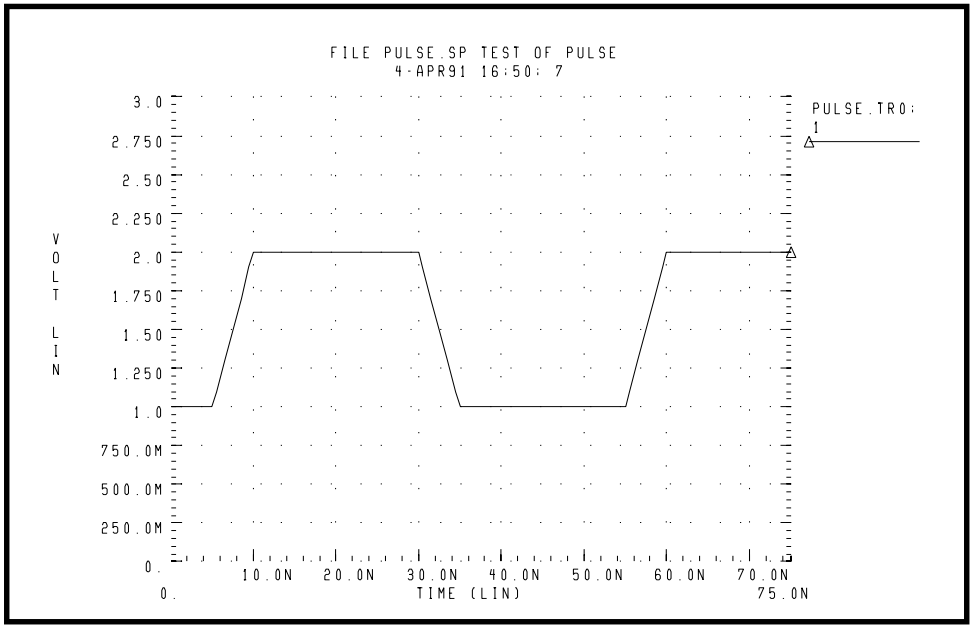

Figure 5-1: Pulse Source Function

Sinusoidal Source Function

Star-Hspice has a damped sinusoidal source that is the product of a dying exponential with a sine wave. Application of this waveform requires the specification of the sine wave frequency, the exponential decay constant, the beginning phase, and the beginning time of the waveform, as explained below.

Syntax

```
SIN <(>vo va <freq <td <θ<φ>>>> <)>
```

where:

vo	voltage or current offset in volts or amps
va	voltage or current amplitude in volts or amps
freq	frequency in Hz (default=1/TSTOP)
td	delay in seconds (default=0.0)
θ	damping factor in 1/seconds (default=0.0)
φ	phase delay in degrees (default=0.0)

Example

```
VIN 3 0 SIN (0 1 100MEG 1NS 1e10)
```

The example specifies a damped sinusoidal source connected between nodes 3 and 0. The waveform has a peak value of 1 V, an offset of 0 V, a 100 MHz frequency, a time delay of 1 ns, a damping factor of 1e10, and a phase delay of zero degrees.

Waveform shape

Time	***Value***
0 to td	$vo + va \cdot SIN\left(\frac{2 \cdot \Pi \cdot \varphi}{360}\right)$
	$vo + va \cdot Exp[-(Time - td) \cdot \theta] \cdot SIN\left\{2 \cdot \Pi \cdot \left[freq \cdot (time - td) + \frac{\varphi}{360}\right]\right\}$
td to tstop	

TSTOP is the final time; see the .TRAN statement for a detailed explanation.

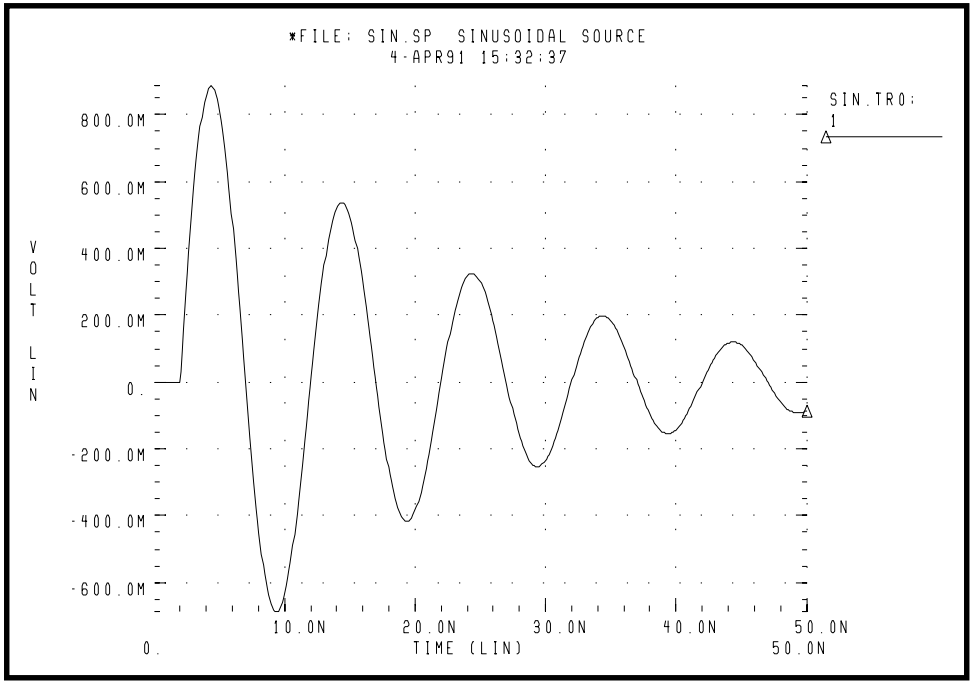

Figure 5-2: Sinusoidal Source Function

Example

```
*File: SIN.SP THE SINUSOIDAL WAVEFORM
*<decay envelope>
```

```
.OPTIONS POST
.PARAM V0=0 VA=1 FREQ=100MEG DELAY=2N THETA=5E7
+ PHASE=0
V  1  0  SIN (V0 VA FREQ DELAY THETA PHASE)
R  1  0  1
.TRAN .05N 50N
.END
```

Exponential Source Function

Syntax

```
EXP <(>v1 v2 <td1 <τ1 <td2 <τ2>>>> <)>
```

where:

v1	initial value of voltage or current in volts or amps
v2	pulsed value of voltage or current in volts or amps
td1	rise delay time in seconds (default=0.0)
td2	fall delay time in seconds (default=td1+TSTEP)
τ1	rise time constant in seconds (default=TSTEP)
τ2	fall time constant in seconds (default=TSTEP)

Example

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

The above example describes an exponential transient source that is connected between nodes 3 and 0. It has an initial t=0 voltage of -4 V and a final voltage of -1 V. The waveform rises exponentially from -4 V to -1 V with a time constant of 30 ns. At 60 ns it starts dropping to -4 V again, with a time constant of 40 ns.

TSTEP is the printing increment, and TSTOP is the final time.

Waveform shape

Time	***Value***

0 to td1 $v1$

td1 to td2 $v1 + (v2 - v1) \cdot \left[1 - Exp\left(-\frac{Time - td1}{\tau_1}\right)\right]$

td2 to tstop $v1 + (v2 - v1) \cdot \left[1 - Exp\left(-\frac{td2 - td1}{\tau_1}\right)\right] \cdot Exp\left[\frac{-(Time - td2)}{\tau_2}\right]$

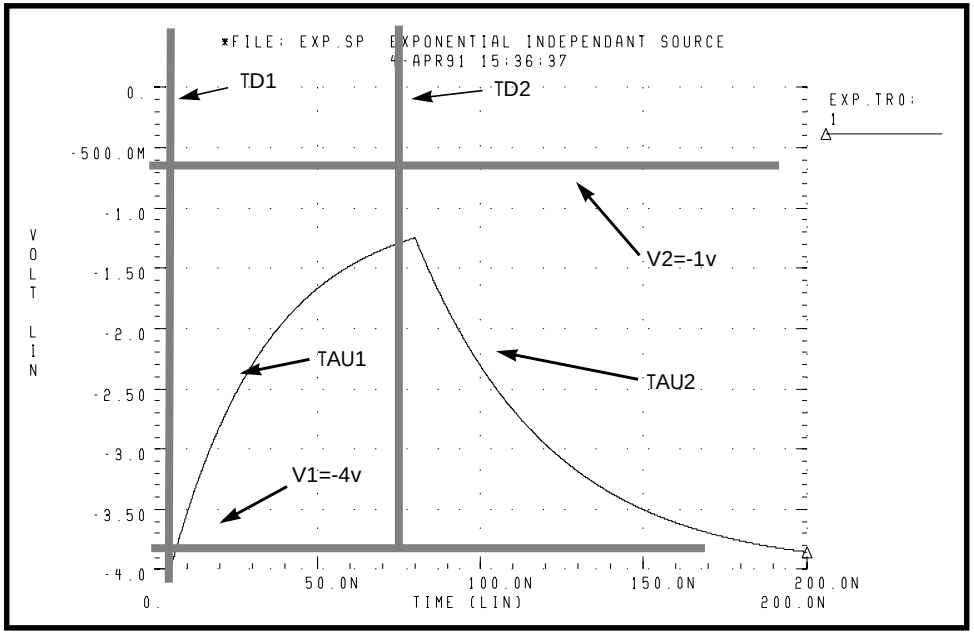

Figure 5-3: Exponential Source Function

Example

```
*FILE: EXP.SP THE EXPONENTIAL WAVEFORM
.OPTIONS POST
.PARAM V1=-4 V2=-1 TD1=5N TAU1=30N TAU2=40N TD2=80N
```

```
V 1 0 EXP (V1 V2 TD1 TAU1 TD2 TAU2)
R 1 0 1
.TRAN .05N 200N
.END
```

Piecewise Linear Source Function

Syntax

```
PWL <(>t1 v1 <t2 v2 t3 v3…> <R <=repeat>> <TD=delay> <)>
```

MSINC and ASPEC

```
PL <(>v1 t1 <v2 t2 v3 t3…> <R <=repeat>> <TD=delay> <)>
```

where

v1 …	specifies current or voltage values
t1 …	specifies segment time values
R	causes the function to repeat
repeat	specifies the start point of the waveform which is to be repeated
TD	is keyword for time delay before piecewise actually starts
delay	specifies the length of time to delay the piecewise linear function

Each pair of values (t1, v1) specifies that the value of the source is v1 (in volts) at time t1. The value of the source at intermediate values of time is determined by linear interpolation between the time points. ASPEC style formats are accommodated by the “PL” form of the function, which reverses the order of the time-voltage pairs to voltage-time pairs. Star-Hspice uses the DC value of the source as the time-zero source value if no time-zero point is given. Also, Star-Hspice does not force the source to terminate at the TSTOP value specified in the .TRAN statement.

Specify “R” to cause the function to repeat. You can specify a value after this “R” to indicate the beginning of the function to be repeated: the repeat time must equal a breakpoint in the function. For example, if t1 = 1, t2 = 2, t3 = 3, and t4 = 4, “repeat” can be equal to 1, 2, or 3.

Specify TD=val to cause a delay at the beginning of the function. You can use TD with or without the repeat function.

Example

```
*FILE: PWL.SP THE REPEATED PIECEWISE LINEAR SOURCE
*ILLUSTRATION OF THE USE OF THE REPEAT FUNCTION “R”
.file pwl.sp REPEATED PIECEWISE LINEAR SOURCE
.OPTION POST
.TRAN 5N 500N
V1 1 0 PWL 60N 0V, 120N 0V, 130N 5V, 170N 5V, 180N 0V, R 0N
R1 1 0 1
V2 2 0 PL  0V 60N, 0V 120N, 5V 130N, 5V 170N, 0V 180N, R 60N
R2 2 0 1
.END
```

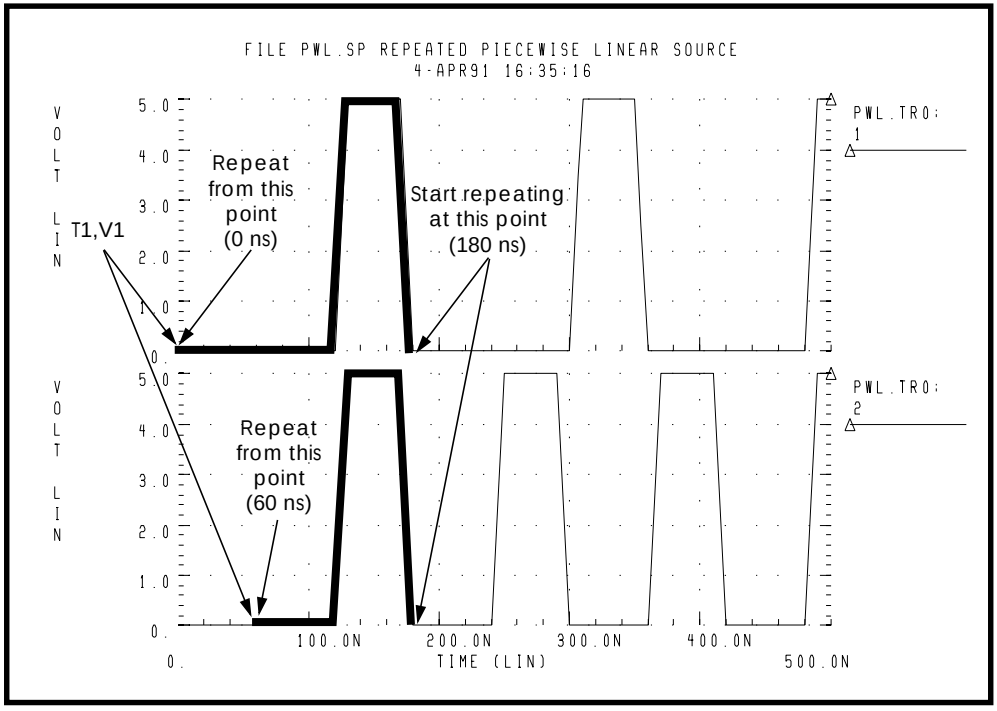

Figure 5-4: Results of Using the Repeat Function

Data Driven Piecewise Linear Source Function

Syntax

```
PWL (TIME, PV)
.DATA datanam
TIME PV
t1   v1
t2   v2
t3   v3
t4   v4
...  ...
.ENDDATA
```

```
.TRAN DATA=datanam
```

where

TIME	parameter for a time value provided in a .DATA statement
PV	parameter for an amplitude value provided in a .DATA statement

You must use this source with a .DATA statement that contains time-value pairs. For each t*n*-v*n* (time-value) pair given in the .DATA block, the data driven PWL function outputs a current or voltage of the given t*n* duration and with the given v*n* amplitude.

This source allows you to use the results of one simulation as an input source in another simulation. The transient analysis must be data driven.

Example

```
*DATA DRIVEN PIECEWISE LINEAR SOURCE
V1 1 0 PWL(TIME, pv1)
R1 1 0 1
V2 2 0 PWL(TIME, pv2)
R2 2 0 1
.DATA dsrc
TIME pv1pv2
0    5v 0v
5n   0v 5v
10n  0v 5v
.ENDDATA
.TRAN DATA=dsrc
.END
```

Single-Frequency FM Source Function

Syntax

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

where

vo	output voltage or current offset, in volts or amps
va	output voltage or current amplitude, in volts or amps
fc	carrier frequency in Hz (default=1/TSTOP)
mdi	modulation index (default=0.0)
fs	signal frequency in Hz (default=1/TSTOP)

Waveform shape

$$sourcevalue = vo + va \cdot [SIN(2 \cdot \pi \cdot fc \cdot Time) + mdi \cdot SIN(2 \cdot \pi \cdot fs \cdot Time)]$$

Note: TSTOP is discussed in the .TRAN statement description.

Example

```
*FILE: SFFM.SP THE SINGLE FREQUENCY FM MODULATION
.OPTIONS POST
V 1 0 SFFM (0, 1M, 20K. 10, 5K)
R 1 0 1
.TRAN .0005M .5MS
.END
```

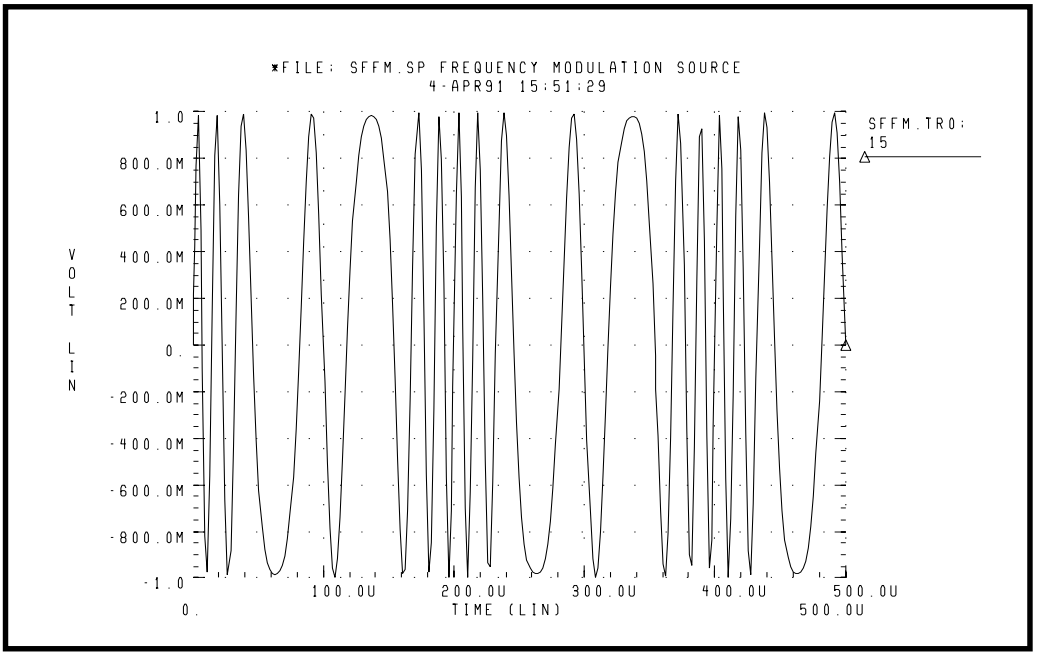

Figure 5-5: Single Frequency FM Modulation

Amplitude Modulation Source Function

Syntax

```
AM (sa oc fm fc td)
```

where

sa	signal amplitude (default=0.0)
fc	carrier frequency (default=0.0)
fm	modulation frequency (default=1/TSTOP)
oc	offset constant (default=0.0)
td	delay time before start of signal (default=0.0)

$$sourcevalue = sa \cdot \{oc + SIN[2 \cdot \pi \cdot fm \cdot (Time - td)]\} \cdot SIN[2 \cdot \pi \cdot fc \cdot (Time - td)]$$

Example

```
.OPTION POST
.TRAN .01M 20M
V1 1 0 AM(10 1 100 1K 1M)
R1 1 0 1
V2 2 0 AM(2.5 4 100 1K 1M)
R2 2 0 1
V3 3 0 AM(10 1 1K 100 1M)
R3 3 0 1
.END
```

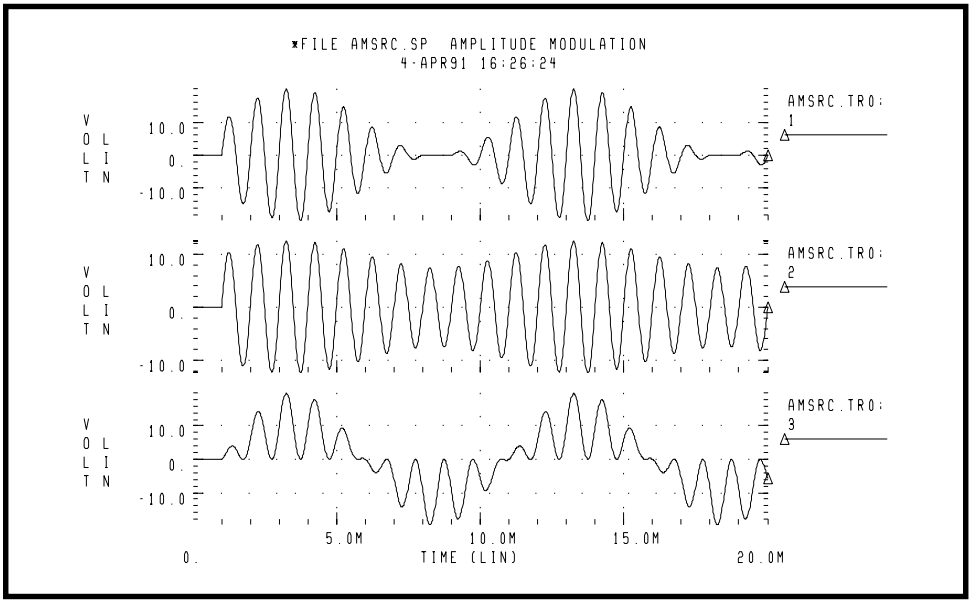

Figure 5-6: Amplitude Modulation Plot

Voltage and Current Controlled Elements

Star-Hspice has four voltage and current controlled elements, known as E, F, G and H elements. You can use these controlled elements in Star-Hspice to model both MOS and bipolar transistors, tunnel diodes, SCRs, as well as analog functions such as operational amplifiers, summers, comparators, voltage controlled oscillators, modulators, and switched capacitor circuits. The controlled elements are either linear or nonlinear functions of controlling node voltages or branch currents, depending on whether you use the polynomial or piecewise linear functions. Each controlled element has different functions:

- The E element is a voltage and/or current controlled voltage source, an ideal op-amp, an ideal transformer, an ideal delay element, or a piecewise linear voltage controlled multi-input AND, NAND, OR, and NOR gate.
- The F element is a current controlled current source, an ideal delay element, or a piecewise linear current controlled multi-input AND, NAND, OR, and NOR gate.
- The G element is a voltage and/or current controlled current source, a voltage controlled resistor, a piecewise linear voltage controlled capacitor, an ideal delay element, or a piecewise linear multi-input AND, NAND, OR, and NOR gate.
- The H element is a current controlled voltage source, an ideal delay element, or a piecewise linear current controlled multi-input AND, NAND, OR, and NOR gate.

The following sections discuss the polynomial and piecewise linear functions and describe element statements for linear or nonlinear functions.

Polynomial Functions

The controlled element statement allows the definition of the controlled output variable (current, resistance, or voltage) as a polynomial function of one or more voltages or branch currents. You can select three polynomial equations through the POLY(NDIM) parameter.

POLY(1)	one-dimensional equation
POLY(2)	two-dimensional equation
POLY(3)	three-dimensional equation

The POLY(1) polynomial equation specifies a polynomial equation as a function of one controlling variable, POLY(2) as a function of two controlling variables, and POLY(3) as a function of three controlling variables.

Along with each polynomial equation are polynomial coefficient parameters (P0, P1 … Pn) that can be set to explicitly define the equation.

One-Dimensional Function

If the function is one-dimensional (a function of one branch current or node voltage), the function value FV is determined by the following expression:

$$FV = P0 + (P1 \cdot FA) + (P2 \cdot FA^2) + (P3 \cdot FA^3) + (P4 \cdot FA^4) + (P5 \cdot FA^5) + \ldots$$

FV	controlled voltage or current from the controlled source
P0. . .PN	coefficients of polynomial equation
FA	controlling branch current or nodal voltage

Note: If the polynomial is one-dimensional and exactly one coefficient is specified, Star-Hspice assumes it to be P1 (P0 = 0.0) to facilitate the input of linear controlled sources.

One-Dimensional Example

The following controlled source statement is an example of a one-dimensional function:

```
E1  5  0  POLY(1)  3  2  1  2.5
```

The above voltage-controlled voltage source is connected to nodes 5 and 0. The single dimension polynomial function parameter, POLY(1), informs Star-Hspice that E1 is a function of the difference of one nodal voltage pair, in this

case, the voltage difference between nodes 3 and 2, hence FA=V(3,2). The dependent source statement then specifies that P0=1 and P1=2.5. From the one-dimensional polynomial equation above, the defining equation for V(5,0) is

$$V(5, 0) = 1 + 2.5 \cdot V(3,2)$$

Two-Dimensional Function

Where the function is two-dimensional (a function of two node voltages or two branch currents), FV is determined by the following expression:

$$FV = P0 + (P1 \cdot FA) + (P2 \cdot FB) + (P3 \cdot FA^2) + (P4 \cdot FA \cdot FB) + (P5 \cdot FB^2) + (P6 \cdot FA^3) + (P7 \cdot FA^2 \cdot FB) + (P8 \cdot FA \cdot FB^2) + (P9 \cdot FB^3) + \ldots$$

For a two-dimensional polynomial, the controlled source is a function of two nodal voltages or currents. To specify a two-dimensional polynomial, set POLY(2) in the controlled source statement.

Two-Dimensional Example

For example, generate a voltage controlled source that gives the controlled voltage, V(1,0), as:

$$V(1, 0) = 3 \cdot V(3,2) + 4 \cdot V(7,6)^2$$

To implement this function, use the following controlled source element statement:

```
E1  1  0  POLY(2)  3  2   7  6   0  3  0  0  0  4
```

This specifies a controlled voltage source connected between nodes 1 and 0 that is controlled by two differential voltages: the voltage difference between nodes 3 and 2 and the voltage difference between nodes 7 and 6, that is, FA=V(3,2) and FB=V(7,6). The polynomial coefficients are P0=0, P1=3, P2=0, P3=0, P4=0, and P5=4.

Three-Dimensional Function

For a three-dimensional polynomial function with arguments FA, FB, and FC, the function value FV is determined by the following expression:

$$\begin{aligned} FV = {} & P0 + (P1 \cdot FA) + (P2 \cdot FB) + (P3 \cdot FC) + (P4 \cdot FA^2) \\ & + (P5 \cdot FA \cdot FB) + (P6 \cdot FA \cdot FC) + (P7 \cdot FB^2) + (P8 \cdot FB \cdot FC) \\ & + (P9 \cdot FC^2) + (P10 \cdot FA^3) + (P11 \cdot FA^2 \cdot FB) + (P12 \cdot FA^2 \cdot FC) \\ & + (P13 \cdot FA \cdot FB^2) + (P14 \cdot FA \cdot FB \cdot FC) + (P15 \cdot FA \cdot FC^2) \\ & + (P16 \cdot FB^3) + (P17 \cdot FB^2 \cdot FC) + (P18 \cdot FB \cdot FC^2) \\ & + (P19 \cdot FC^3) + (P20 \cdot FA^4) + \ldots \end{aligned}$$

Three-Dimensional Example

For example, generate a voltage controlled source that gives the voltage as:

$$V(1, 0) = 3 \cdot V(3,2) + 4 \cdot V(7,6)^2 + 5 \cdot V(9,8)^3$$

from the above defining equation and the three-dimensional polynomial equation:

$$FA = V(3,2)$$

$$FB = V(7,6)$$

$$FC = V(9,8)$$

$$P1 = 3$$

$$P7 = 4$$

$$P19 = 5$$

Substituting these values into the voltage controlled voltage source statement yields the following:

V(1, 0) POLY(3) 3 2 7 6 9 8 0 3 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 5

The above specifies a controlled voltage source connected between nodes 1 and 0 that is controlled by three differential voltages: the voltage difference between nodes 3 and 2, the voltage difference between nodes 7 and 6, and the voltage difference between nodes 9 and 8, that is, FA=V(3,2), FB=V(7,6), and FC=V(9,8). The statement gives the polynomial coefficients as P1=3, P7=4, P19=5, and the rest are zero.

Piecewise Linear Function

The one-dimensional piecewise linear function allows you to model some special element characteristics, such as those of tunnel diodes, silicon controlled rectifiers, and diode breakdown regions. The piecewise linear function can be described by specifying measured data points. Although the device characteristic is described by some data points, Star-Hspice automatically smooths the corners to ensure derivative continuity and, as a result, better convergence.

A parameter DELTA is provided to control the curvature of the characteristic at the corners. The smaller the DELTA, the sharper the corners are. The maximum DELTA is limited to half of the smallest breakpoint distance. If the breakpoints are quite separated, specify the DELTA to a proper value. You can specify up to 100 point pairs. At least two point pairs (four coefficients) must be specified.

In order to model bidirectional switch or transfer gates, the functions NPWL and PPWL are provided for G elements. The NPWL and PPWL function like NMOS and PMOS transistors.

The piecewise linear function also models multi-input AND, NAND,OR, and NOR gates. In this case, only one input determines the state of the output. In AND / NAND gates, the input with the smallest value is used in the piecewise linear function to determine the corresponding output of the gates. In the OR / NOR gates, the input with the largest value is used to determine the corresponding output of the gates.

Voltage Dependent Current Sources — G Elements

G element syntax statements are described in the following pages. The parameters are defined in the following section.

Voltage Controlled Current Source (VCCS)

Syntax

Linear

```
Gxxx n+ n- <VCCS> in+ in- transconductance <MAX=val> <MIN=val> <SCALE=val>
+          <M=val> <TC1=val> <TC2=val> <ABS=1> <IC=val>
```

Polynomial

```
Gxxx  n+ n- <VCCS> POLY(NDIM) in1+ in1- ... <inndim+ inndim-> MAX=val>
+          <MIN=val> <SCALE=val> <M=val> <TC1=val> <TC2=val> <ABS=1> P0
+          <P1…> <IC=vals>
```

Piecewise Linear

```
Gxxx  n+ n- <VCCS> PWL(1) in+ in- <DELTA=val> <SCALE=val> <M=val>
+          TC1=val> <TC2=val> x1,y1 x2,y2 ... x100,y100 <IC=val>
+          <SMOOTH=val>
Gxxx  n+ n- <VCCS> NPWL(1) in+ in- <DELTA=val> <SCALE=val> <M=val>
+          <TC1=val><TC2=val> x1,y1 x2,y2 ... x100,y100 <IC=val> <SMOOTH=val>
Gxxx  n+ n- <VCCS> PPWL(1) in+ in- <DELTA=val> <SCALE=val> <M=val>
+          <TC1=val> <TC2=val> x1,y1 x2,y2 ... x100,y100 <IC=val> <SMOOTH=val>
```

Multi-Input Gates

```
Gxxx  n+ n- <VCCS> gatetype(k) in1+ in1- ... ink+ ink- <DELTA=val> <TC1=val>
+          <TC2=val> <SCALE=val> <M=val> x1,y1 ... x100,y100<IC=val>
```

Delay Element

```
Gxxx n+ n- <VCCS> DELAY in+ in- TD=val <SCALE=val> <TC1=val> <TC2=val>
+          NPDELAY=val
```

Behavioral Current Source

Syntax

```
Gxxx n+ n- CUR='equation' <MAX>=val> <MIN=val> <M=val> <SCALE=val>
```

Voltage Controlled Resistor (VCR)

Syntax

Linear

```
Gxxx n+ n- VCR in+ in- transfactor <MAX=val> <MIN=val> <SCALE=val> <M=val>
+          <TC1=val> <TC2=val> <IC=val>
```

Polynomial

```
Gxxx  n+ n- VCR POLY(NDIM) in1+ in1- ... <inndim+ inndim-> <MAX=val>
+          <MIN=val><SCALE=val> <M=val> <TC1=val> <TC2=val>    P0 <P1…>
+          <IC=vals>
```

Piecewise Linear

```
Gxxx  n+ n- VCR PWL(1) in+ in- <DELTA=val> <SCALE=val> <M=val> <TC1=val>
+          <TC2=val> x1,y1 x2,y2 ... x100,y100 <IC=val> <SMOOTH=val>

Gxxx  n+ n- VCR NPWL(1) in+ in- <DELTA=val> <SCALE=val> <M=val> <TC1=val>
+          <TC2=val> x1,y1 x2,y2 ... x100,y100 <IC=val> <SMOOTH=val>

Gxxx  n+ n- VCR PPWL(1) in+ in- <DELTA=val> <SCALE=val> <M=val> <TC1=val>
+          <TC2=val> x1,y1 x2,y2 ... x100,y100 <IC=val> <SMOOTH=val>
```

Multi-Input Gates

```
Gxxx  n+ n- VCR gatetype(k) in1+ in1- ... ink+ ink- <DELTA=val>
+          <TC1=val> <TC2=val> <SCALE=val> <M=val> x1,y1 ... x100,y100 <IC=val>
```

Voltage Controlled Capacitor (VCCAP)

Syntax (Piecewise Linear)

```
Gxxx  n+ n- VCCAP PWL(1) in+ in-   <DELTA=val> <SCALE=val> <M=val>
+          <TC1=val><TC2=val> x1,y1 x2,y2 ... x100,y100 <IC=val> <SMOOTH=val>
```

The two functions NPWL and PPWL allow the interchange of the “n+” and “n-” nodes while keeping the same transfer function. This action is summarized as follows:

NPWL Function

For node “in-” connected to “n-”:

If v(n+,n-) > 0, then the controlling voltage would be v(in+,in-). Otherwise, the controlling voltage is v(in+,n+).

For node “in-” connected to “n+”:

If v(n+,n-) < 0, then the controlling voltage would be v(in+,in-). Otherwise, the controlling voltage is v(in+,n+).

PPWL Function

For node “in-” connected to “n-”:

If v(n+,n-) < 0, then the controlling voltage would be v(in+,in1-). Otherwise, the controlling voltage is v(in+,n+).

For node “in-” connected to “n+”:

If v(n+,n-) > 0, then the controlling voltage would be v(in+,in-). Otherwise, the controlling voltage is v(in+,n+).

Parameter Definitions

ABS	Output is absolute value if ABS=1.
CUR, VALUE	current output that flows from n+ to n-. The equation that you define can be a function of node voltages, branch currents, TIME, temperature (TEMPER), and frequency (HERTZ).
DELAY	keyword for the delay element. The delay element is the same as voltage controlled current source except it is associated by a propagation delay TD. This element facilitates the adjustment of propagation delay in the macromodel process. **Note:** Because DELAY is an Star-Hspice keyword, it should not be used as a node name.
DELTA	used to control the curvature of the piecewise linear corners. The parameter defaults to 1/4 of the smallest breakpoint distances. The maximum is limited to 1/2 of the smallest breakpoint distances.
Gxxx	voltage controlled element name. This parameter must begin with a “G” followed by up to 1023 alphanumeric characters.

gatetype(k) can be one of AND, NAND, OR, or NOR. The parameter (k) represents the number of inputs of the gate. The x's and y's represents the piecewise linear variation of output as a function of input. In the multi-input gates, only one input determines the state of the output.

IC initial condition. The initial estimate of the value(s) of the controlling voltage(s). If IC is not specified, the default=0.0.

in +/- positive or negative controlling nodes. Specify one pair for each dimension.

M number of element in parallel

MAX maximum current or resistance value. The default is undefined and sets no maximum value.

MIN minimum current or resistance value. The default is undefined and sets no minimum value.

n+/- positive or negative node of controlled element

NDIM polynomial dimensions. If POLY(NDIM) is not specified, a one-dimensional polynomial is assumed. NDIM must be a positive number.

NPDELAY sets the number of data points to be used in delay simulations. The default value is the larger of 10 or the smaller of TD/tstep and tstop/tstep

That is,

$$NPDELAY_{default} = max\left[\frac{min\langle TD, tstop\rangle}{tstep}, 10\right]$$

The values of tstep and tstop are specified in the .TRAN statement.

NPWL models the symmetrical bidirectional switch or transfer gate, NMOS

P0, P1 … the polynomial coefficients. When one coefficient is specified, Star-Hspice assumes it to be P1 (P0=0.0), and the element is linear. When more than one polynomial coefficient is specified, the element is nonlinear, and P0, P1, P2 ... represent them (see *Polynomial Functions, page -20 in , Using Sources and Stimuli*).

POLY polynomial keyword function

PWL piecewise linear keyword function

PPWL models the symmetrical bidirectional switch or transfer gate, PMOS

SCALE element value multiplier

SMOOTH For piecewise linear dependent source elements, SMOOTH selects the curve smoothing method.

A curve smoothing method simulates exact data points you provide. This method can be used to make Star-Hspice simulate specific data points that correspond to measured data or data sheets, for example.

Choices for SMOOTH are 1 or 2:

1 Selects the smoothing method used in Hspice releases prior to release H93A. Use this method to maintain compatibility with simulations done using releases older than H93A.

2 Selects the smoothing method that uses data points you provide. This is the default for Hspice releases starting with 93A.

TC1,TC2 first and second order temperature coefficients. The SCALE is updated by temperature:

$$SCALEeff = SCALE \cdot (1 + TC1 \cdot \Delta t + TC2 \cdot \Delta t^2)$$

TD time delay keyword

transconductance	voltage-to-current conversion factor
transfactor	voltage-to-resistance conversion factor
VCCAP	the keyword for voltage controlled capacitance element. VCCAP is a reserved word and should not be used as a node name.
VCCS	the keyword for voltage controlled current source. VCCS is a reserved word and should not be used as a node name.
VCR	the keyword for voltage controlled resistor element. VCR is a reserved word and should not be used as a node name.
x1,...	controlling voltage across nodes in+ and in- . The x values must be in increasing order.
y1,...	corresponding element values of x

Examples

Switch

A voltage controlled resistor represents a basic switch characteristic. The resistance between nodes 2 and 0 varies linearly from 10 meg to 1 m ohms when voltage across nodes 1 and 0 varies between 0 and 1 volt. Beyond the voltage limits, the resistance remains at 10 meg and 1 m ohms, respectively.

```
Gswitch 2 0 VCR PWL(1) 1 0  0v,10meg 1v,1m
```

Switch-Level MOSFET

Model a switch level n-channel MOSFET by the N-piecewise linear resistance switch. The resistance value does not change when the node d and s positions are switched.

```
Gnmos d s VCR NPWL(1) g s LEVEL=1 0.4v,150g
+  1v,10meg 2v,50k 3v,4k 5v,2k
```

Voltage Controlled Capacitor

The capacitance value across nodes (out,0) varies linearly from 1 p to 5 p when voltage across nodes (ctrl,0) varies between 2 v and 2.5 v. Beyond the voltage limits, the capacitance value remains constant at 1 picofarad and 5 picofarads respectively.

```
Gcap out 0 VCCAP PWL(1) ctrl 0  2v,1p 2.5v,5p
```

Zero Delay Gate

Implement a two-input AND gate using an expression and a piecewise linear table. The inputs are voltages at nodes a and b, and the output is the current flow from node out to 0. The current is multiplied by the SCALE value, which in this example is specified as the inverse of the load resistance connected across the nodes (out,0).

```
Gand out 0 AND(2) a 0 b 0 SCALE='1/rload' 0v,0a 1v,.5a
+  4v,4.5a 5v,5a
```

Delay Element

A delay is a low-pass filter type delay similar to that of an opamp. A transmission line, on the other hand, has an infinite frequency response. A glitch input to a G delay is attenuated similarly to a buffer circuit. In this example, the output of the delay element is the current flow from node *out* to node *1* with a value equal to the voltage across nodes (*in*, *0*) multiplied by SCALE value and delayed by TD value.

```
Gdel out 0 DELAY in 0 TD=5ns SCALE=2 NPDELAY=25
```

Diode Equation

Model forward bias diode characteristic from node 5 to ground with a runtime expression. The saturation current is 1e-14 amp, and the thermal voltage is 0.025 v.

```
Gdio 5 0 CUR='1e-14*(EXP(V(5)/0.025)-1.0)'
```

Diode Breakdown

Model a diode breakdown region to forward region using the following example. When voltage across the diode goes beyond the piecewise linear limit values (-2.2v, 2v), the diode current remains at the corresponding limit values (-1a, 1.2a).

```
Gdiode 1 0 PWL(1) 1 0 -2.2v,-1a -2v,-1pa .3v,.15pa
+  .6v,10ua 1v,1a 2v,1.2a
```

Triode

Both the following voltage controlled current sources implement a basic triode. The first uses the poly(2) operator to multiply the anode and grid voltages together and scale by .02. The next example uses the explicit behavioral algebraic description.

```
gt i_anode cathode poly(2) anode,cathode grid,cathode 0 0
+0 0 .02
gt i_anode cathode
+cur='20m*v(anode,cathode)*v(grid,cathode)'
```

Current Dependent Current Sources — F Elements

F element syntax statements are described in the following paragraphs. The parameter definitions follow.

Current Controlled Current Source (CCCS)

Syntax

Linear

```
Fxxx n+ n- <CCCS> vn1 gain <MAX=val> <MIN=val> <SCALE=val> <TC1=val>
+          <TC2=val>        <M=val> <ABS=1> <IC=val>
```

Polynomial

```
Fxxx n+ n- <CCCS> POLY(NDIM) vn1 <... vnndim> <MAX=val> <MIN=val>
+       <TC1=val> <TC2=val> <SCALE=vals> <M=val> <ABS=1> P0   <P1…>
+       <IC=vals>
```

Piecewise Linear

```
Fxxx n+ n- <CCCS> PWL(1) vn1 <DELTA=val> <SCALE=val><TC1=val> <TC2=val>
+       <M=val>   x1,y1 ... x100,y100 <IC=val>
```

Multi-Input Gates

```
Fxxx  n+ n- <CCCS> gatetype(k) vn1, ... vnk <DELTA=val> <SCALE=val> <TC1=val>
+       <TC2=val> <M=val> <ABS=1> x1,y1   ...   x100,y100    <IC=val>
```

Delay Element

```
Fxxx n+ n- <CCCS> DELAY vn1 TD=val <SCALE=val> <TC1=val><TC2=val>
+          NPDELAY=val
```

Parameter Definitions

ABS	Output is absolute value if ABS=1.
CCCS	the keyword for current controlled current source. Note that CCCS is a reserved word and should not be used as a node name.

DELAY	keyword for the delay element. The delay element is the same as a current controlled current source except it is associated by a propagation delay TD. This element facilitates the adjustment of propagation delay in the macromodel process. **Note:** DELAY is a reserved word and should not be used as a node name.
DELTA	used to control the curvature of the piecewise linear corners. The parameter defaults to 1/4 of the smallest breakpoint distances. The maximum is limited to 1/2 of the smallest breakpoint distances.
Fxxx	current controlled current source element name. The parameter must begin with an “F”, followed by up to 1023 alphanumeric characters.
gain	current gain
gatetype(k)	can be one of AND, NAND, OR, or NOR. (k) represents the number of inputs of the gate. The x’s and y’s represent the piecewise linear variation of output as a function of input. In the multi-input gates, only one input determines the state of the output. The above keyword names should not be used as a node name.
IC	initial condition: the initial estimate of the value(s) of the controlling current(s) in amps. If IC is not specified, the default=0.0.
M	number of element in parallel
MAX	maximum output current value. The default is undefined and sets no maximum value.
MIN	minimum output current value. The default is undefined and sets no minimum value.
n+/-	positive or negative controlled source connecting nodes

NDIM polynomial dimensions. If POLY(NDIM) is not specified, a one-dimensional polynomial is assumed. NDIM must be a positive number.

NPDELAY sets the number of data points to be used in delay simulations. The default value is the larger of 10 or the smaller of TD/tstep and tstop/tstep.

That is,

$$NPDELAY_{default} = max\left[\frac{min\langle TD, tstop\rangle}{tstep}, 10\right]$$

The values of tstep and tstop are specifed in the .TRAN statement.

P0, P1 … when one polynomial coefficient is specified, Star-Hspice assumes it to be P1 (P0=0.0) and the source is linear. When more than one polynomial coefficient is specified, the source is nonlinear, and P0, P1, P2 … represent them.

POLY polynomial keyword function

PWL piecewise linear keyword function

SCALE element value multiplier

TC1,TC2 first and second order temperature coefficients. The SCALE is updated by temperature:

$$SCALEeff = SCALE \cdot (1 + TC1 \cdot \Delta t + TC2 \cdot \Delta t^2)$$

TD time delay keyword

vn1 … names of voltage sources through which the controlling current flows. One name must be specified for each dimension.

x1,...	controlling current through vn1 source. The x values must be in increasing order.
y1,...	corresponding output current values of x

Examples

```
F1 13 5 VSENS MAX=+3 MIN=-3 5
```

This example describes a current controlled current source connected between nodes 13 and 5. The current that controls the value of the controlled source flows through the voltage source named VSENS (to use a current controlled current source, a dummy independent voltage source is often placed into the path of the controlling current). The defining equation is:

$$I(F1) = 5 \cdot I(VSENS)$$

The current gain is 5, the maximum current flow through F1 is 3 A, and the minimum current flow is -3 A. If I(VSENS) = 2 A, I(F1) would be set to 3 amps and not 10 amps as would be suggested by the equation. A user-defined parameter can be specified for the polynomial coefficient(s), as shown below.

```
.PARAM VU = 5
F1 13 5 VSENS MAX=+3 MIN=-3 VU
```

The next example describes a current controlled current source with the value:

$$I(F2)=1e\text{-}3 + 1.3e\text{-}3 \cdot I(VCC)$$

```
F2 12 10 POLY VCC 1MA 1.3M
```

Current flow is from the positive node through the source to the negative node. The direction of positive controlling current flow is from the positive node through the source to the negative node of vnam (linear), or to the negative node of each voltage source (nonlinear).

```
Fd 1 0 DELAY vin TD=7ns SCALE=5
```

This example is a delayed current controlled current source.

```
Filim 0 out PWL(1) vsrc -1a,-1a 1a,1a
```

The final example is a piecewise linear current controlled current source.

Voltage Dependent Voltage Sources — E Elements

E element syntax statements are described in the following paragraphs. The parameters are defined in the following section.

Voltage Controlled Voltage Source (VCVS)

Syntax

Linear

```
Exxx n+ n- <VCVS> in+ in- gain <MAX=val> <MIN=val> <SCALE=val>
+        <TC1=val> <TC2=val><ABS=1> <IC=val>
```

Polynomial

```
Exxx  n+ n- <VCVS> POLY(NDIM) in1+ in1- ... inndim+ inndim-<TC1=val>
+        <TC2=val><SCALE=val><MAX=val><MIN=val> <ABS=1> P0 <P1…>
+        <IC=vals>
```

Piecewise Linear

```
Exxx  n+ n- <VCVS> PWL(1) in+ in-    <DELTA=val> <SCALE=val> <TC1=val>
+        <TC2=val> x1,y1 x2,y2 ...   x100,y100 <IC=val>
```

Multi-Input Gates

```
Exxx  n+ n- <VCVS> gatetype(k) in1+ in1- ... ink+ ink- <DELTA=val> <TC1=val>
+        <TC2=val> <SCALE=val> x1,y1   ...   x100,y100 <IC=val>
```

Delay Element

```
Exxx n+ n- <VCVS> DELAY in+ in- TD=val <SCALE=val> <TC1=val> <TC2=val>
+       <NPDELAY=val>
```

Behavioral Voltage Source

The syntax is:

```
Exxx n+ n- VOL='equation' in+ in- <MAX>=val> <MIN=val>
```

Ideal Op-Amp

The syntax is:

```
Exxx n+ n- OPAMP in+ in-
```

Ideal Transformer

The syntax is:

Exxx n+ n- TRANSFORMER in+ in- k

Parameter Definitions

ABS	Output is absolute value if ABS=1.
DELAY	keyword for the delay element. The delay element is the same as voltage controlled voltage source, except it is associated by a propagation delay TD. This element facilitates the adjustment of propagation delay in the macro-modelling process. **Note:** DELAY is a reserved word and should not be used as a node name.
DELTA	used to control the curvature of the piecewise linear corners. The parameter defaults to one-fourth of the smallest breakpoint distances. The maximum is limited to one-half of the smallest breakpoint distances.
Exxx	voltage controlled element name. The parameter must begin with an “E” followed by up to 1023 alphanumeric characters.
gain	voltage gain
gatetype(k)	can be one of AND, NAND, OR, or NOR. (k) represents the number of inputs of the gate. The x’s and y’s represent the piecewise linear variation of output as a function of input. In the multi-input gates only one input determines the state of the output.
IC	initial condition: the initial estimate of the value(s) of the controlling voltage(s). If IC is not specified, the default=0.0.
in +/-	positive or negative controlling nodes. Specify one pair for each dimension.
k	ideal transformer turn ratio: V(in+,in-) = $k \cdot$ V(n+,n-) or, number of gates input

MAX	maximum output voltage value. The default is undefined and sets no maximum value.
MIN	minimum output voltage value. The default is undefined and sets no minimum value.
n+/-	positive or negative node of controlled element
NDIM	polynomial dimensions. If POLY(NDIM) is not specified, a one-dimensional polynomial is assumed. NDIM must be a positive number.
NPDELAY	sets the number of data points to be used in delay simulations. The default value is the larger of 10 or the smaller of TD/tstep and tstop/tstep That is, $$NPDELAY_{default} = max\left[\frac{min\langle TD, tstop\rangle}{tstep}, 10\right]$$ The values of tstep and tstop are specifed in the .TRAN statement.
OPAMP	the keyword for ideal op-amp element. OPAMP is a reserved word and should not be used as a node name.
P0, P1 …	the polynomial coefficients. When one coefficient is specified, Star-Hspice assumes it to be P1 (P0=0.0), and the element is linear. When more than one polynomial coefficient is specified, the element is nonlinear, and P0, P1, P2 ... represent them (see *Polynomial Functions, page -20 in , Using Sources and Stimuli*).
POLY	polynomial keyword function
PWL	piecewise linear keyword function
SCALE	element value multiplier

TC*1,TC2*	first and second order temperature coefficients. The SCALE is updated by temperature: $\text{SCALEeff} = \text{SCALE} \cdot (1 + TC1 \cdot \Delta t + TC2 \cdot \Delta t^2)$
TD	time delay keyword
TRANSFORMER	the keyword for ideal transformer. TRANS is a reserved word and should not be used as a node name.
VCVS	the keyword for voltage controlled voltage source. VCVS is a reserved word and should not be used as a node name.
x1,...	controlling voltage across nodes in+ and in-. The x values must be in increasing order.
y1,...	corresponding element values of x

Examples

Ideal OpAmp

A voltage amplifier with supply limits can be built with the voltage controlled voltage source. The output voltage across nodes 2,3 = v(14,1) * 2. The voltage gain parameter, 2, is also given. The MAX and MIN parameters specify a maximum E1 voltage of 5 V and a minimum E1 voltage output of -5 V. If, for instance, V(14,1) = -4V, E1 would be set to -5 V and not -8 V, as the equation would produce.

```
Eopamp 2 3 14 1 MAX=+5 MIN=-5  2.0
```

A user-defined parameter can be used in the following format to specify a value for polynomial coefficient parameters:

```
.PARAM CU = 2.0
E1 2 3 14 1 MAX=+5 MIN=-5  CU
```

Voltage Summer

An ideal voltage summer specifies the source voltage as a function of three controlling voltage(s): V(13,0), V(15,0) and V(17,0). It describes a voltage source with the value:

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

This example represents an ideal voltage summer. The three controlling voltages are initialized for a DC operating point analysis to 1.5, 2.0, and 17.25 V, respectively.

```
EX 17 0 POLY(3) 13 0 15 0 17 0 0 1 1 1
+  IC=1.5,2.0,17.25
```

Polynomial Function

The voltage controlled source also can output a nonlinear function using the one-dimensional polynomial. Since the POLY parameter is not specified, a one-dimensional polynomial is assumed— that is, a function of one controlling voltage. The equation corresponds to the element syntax. Behavioral equations replace this older method.

$$V(3,4) = 10.5 + 2.1 * V(21,17) + 1.75 * V(21,17)^2$$

```
E2 3 4 POLY 21 17 10.5 2.1 1.75
```

Zero Delay Inverter Gate

You can build a simple inverter with no delay with a piecewise linear transfer function.

```
Einv out 0 PWL(1) in 0 .7v,5v 1v,0v
```

Ideal Transformer

With the turn ratio 10 to 1, the voltage relationship is V(out)=V(in)/10.

```
Etrans out 0 TRANSFORMER in 0 10
```

Voltage Controlled Oscillator (VCO)

Use the keyword VOL to define a single-ended input that controls the output of a VCO.

In the following example, the frequency of the sinusoidal output voltage at node “out” is controlled by the voltage at node “control”. Parameter “v0” is the DC offset voltage and “gain” is the amplitude. The output is a sinusoidal voltage with a frequency of “freq · control”.

```
Evco out 0 VOL='v0+gain*SIN(6.28 freq*v(control)
+ *TIME)'
```

Dependent Voltage Sources — H Elements

H element syntax statements are described in the following paragraphs. The parameters are defined in the following section.

Current Controlled Voltage Source — CCVS

Syntax

Linear

```
Hxxx n+ n- <CCVS> vn1 transresistance <MAX=val> <MIN=val> <SCALE=val>
+          <TC1=val><TC2=val> <ABS=1> <IC=val>
```

Polynomial

```
Hxxx n+ n- <CCVS> POLY(NDIM) vn1 <... vnndim> <MAX=val>MIN=val>
+          <TC1=val><TC2=val> <SCALE=val> <ABS=1> P0   <P1…> <IC=vals>
```

Piecewise Linear

```
Hxxx n+ n- <CCVS> PWL(1) vn1 <DELTA=val> <SCALE=val> <TC1=val> <TC2=val>
+        x1,y1   ...   x100,y100 <IC=val>
```

Multi-Input Gates

```
Hxxx  n+ n- gatetype(k) vn1, ... vnk <DELTA=val> <SCALE=val> <TC1=val
+          <TC2=val> x1,y1 ...   x100,y100 <IC=val>
```

Delay Element

```
Hxxx n+ n- <CCVS> DELAY vn1 TD=val <SCALE=val> <TC1=val> <TC2=val>
+        <NPDELAY=val>
```

Parameter Definitions

ABS	Output is absolute value if ABS=1.
CCVS	the keyword for current controlled voltage source. CCVS is a reserved word and should not be used as a node name.
DELAY	keyword for the delay element. The delay element is the same as a current controlled voltage source except it is associated by a propagation delay TD. This element

	facilitates the adjustment of propagation delay in the macromodel process. DELAY is a reserved word and should not be used as a node name.
DELTA	used to control the curvature of the piecewise linear corners. The parameter defaults to 1/4 of the smallest breakpoint distances. The maximum is limited to 1/2 of the smallest breakpoint distances.
gatetype(k)	can be one of AND, NAND, OR, NOR. (k) represents the number of inputs of the gate. The x's and y's represent the piecewise linear variation of output as a function of input. In the multi-input gates only one input determines the state of the output.
Hxxx	current controlled voltage source element name. The parameter must begin with an "H" followed by up to 1023 alphanumeric characters.
IC	initial condition. This is the initial estimate of the value(s) of the controlling current(s) in amps. If IC is not specified, the default=0.0.
MAX	maximum voltage value. The default is undefined and sets no maximum value.
MIN	minimum voltage value. The default is undefined and sets no minimum value.
n+/-	positive or negative controlled source connecting nodes
NDIM	polynomial dimensions. If POLY(NDIM) is not specified, a one-dimensional polynomial is assumed. NDIM must be a positive number.

NPDELAY sets the number of data points to be used in delay simulations. The default value is the larger of

10 or the smaller of TD/tstep and tstop/tstep

That is,

$$NPDELAY_{default} = max\left[\frac{min\langle TD, tstop\rangle}{tstep}, 10\right]$$

The values of tstep and tstop are specifed in the .TRAN statement.

P0, P1 . . . When one polynomial coefficient is specified, the source is linear, and the polynomial is assumed to be P1 (P0=0.0). When more than one polynomial coefficient is specified, the source is nonlinear, with the polynomials assumed as P0, P1, P2 …

POLY polynomial keyword function

PWL piecewise linear keyword function

SCALE element value multiplier

TC1,TC2 first and second order temperature coefficients. The SCALE is updated by temperature:

$$\text{SCALEeff} = \text{SCALE} \cdot (1 + TC1 \cdot \Delta t + TC2 \cdot \Delta t^2)$$

TD time delay keyword

transresistance current to voltage conversion factor

vn1 … names of voltage sources through which the controlling current flows. One name must be specified for each dimension.

x1,... controlling current through vn1 source. The x values must be in increasing order.

y1,... corresponding output voltage values of x

Examples

```
HX 20 10 VCUR MAX=+10 MIN=-10 1000
```

The example above selects a linear current controlled voltage source. The controlling current flows through the dependent voltage source called VCUR. The defining equation of the CCVS is:

$$HX = 1000 \cdot VCUR$$

The defining equation specifies that the voltage output of HX is 1000 times the value of current flowing through CUR. If the equation produces a value of HX greater than +10 V or less than -10 V, HX, because of the MAX= and MIN= parameters, would be set to either 10 V or -10 V, respectively. CUR is the name of the independent voltage source that the controlling current flows through. If the controlling current does not flow through an independent voltage source, a dummy independent voltage source must be inserted.

```
.PARAM CT=1000
HX 20 10 VCUR MAX=+10 MIN=-10 CT
HXY 13 20 POLY(2) VIN1 VIN2 0 0 0 0 1 IC=0.5, 1.3
```

The example above describes a dependent voltage source with the value:

$$V = I(VIN1) \cdot I(VIN2)$$

This two-dimensional polynomial equation specifies FA1=VIN1, FA2=VIN2, P0=0, P1=0, P2=0, P3=0, and P4=1. The controlling current for flowing through VIN1 is initialized at .5 mA. For VIN2, the initial current is 1.3 mA.

The direction of positive controlling current flow is from the positive node, through the source, to the negative node of vnam (linear). The polynomial (nonlinear) specifies the source voltage as a function of the controlling current(s).

Digital Files and Mixed Mode — U Elements

The U element can reference digital input and digital output models for mixed mode simulation. Viewlogic's Viewsim mixed mode simulator uses Star-Hspice with digital input from Viewsim. The state information comes from a digital file if Star-Hspice is being run in standalone mode. Digital outputs are handled in a similar fashion. In digital input file mode, the input file is <*design*>*.d2a* and the output file is named <*design*>*.a2d*.

A2D and D2A functions accept the terminal "\" backslash character as a line-continuation character to allow more than 255 characters in a line. This is needed because the first line of a digital file, which contains the signal name list, is often longer than the maximum line length accepted by some text editors.

A digital D2A file must not have a blank first line. If the first line of a digital file is blank, Star-Hspice issues an error message.

The following example demonstrates the use of the "\" line continuation character to format an input file for text editing. The file contains a signal list for a 64-bit bus.

```
...
a00 a01 a02 a03 a04 a05 a06 a07 \
a08 a09 a10 a11 a12 a13 a14 a15 \
...                              * Continuation of signal
names
a56 a57 a58 a59 a60 a61 a62 a63  * End of signal names
...                              * Remainder of file
```

Digital Input Element and Model

Syntax

U<name> <(interface) node> <(lo_ref)node> <(hi_ref)node> <(model) name>
+ SIGNAME = <(digital signal) name>] [IS = (initial state)]

Example

```
UC carry-in VLD2A VHD2A D2A SIGNAME=1 IS=0
VLD2A VLD2A 0 DC lo
VHD2A VHD2A 0 DC hi
```

Model Form

.MODEL <(model) name> U LEVEL=5 [(model parameters)]

Examples

```
.MODEL D2A U LEVEL=5 TIMESTEP=0.1NS,
+ S0NAME=0 S0TSW=1NS S0RLO = 15, S0RHI = 10K,
+ S2NAME=x S2TSW=3NS S2RLO = 1K, S2RHI = 1K
+ S3NAME=z S3TSW=5NS S3RLO = 1MEG,S3RHI = 1MEG
+ S4NAME=1 S4TSW=1NS S4RLO = 10K, S4RHI = 60
```

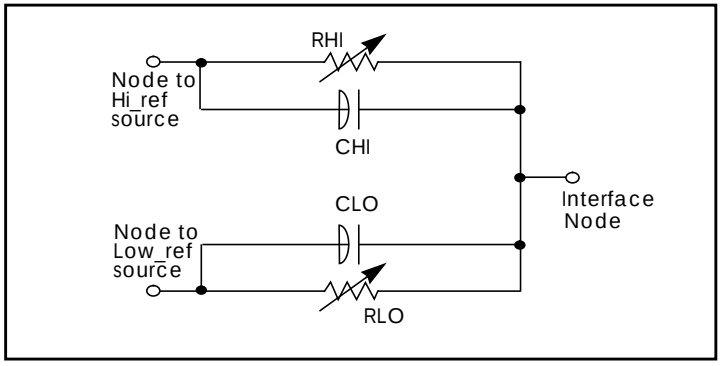

Figure 5-7: Digital-to-Analog Converter Element

Digital-to-Analog Model Parameters

Names(Alias)	Units	Default	Description
CLO	farad	0	Capacitance to low level node
CHI	farad	0	Capacitance to high level node
SONAME			State “O” character abbreviation
SOTSW	sec		State “O” switching time
SORLO	ohm		State “O” resistance to low level node
SORHI	ohm		State “O” resistance to high level node
S1NAME			State “1” character abbreviation
S1TSW	sec		State “1” switching time
S1RLO	ohm		State “1”resistace to low level node
S1RHI	ohm		State “1” resistance to high level node
S2NAME			State “2” character abbreviation
S2TSW	sec		State “2” switching time
S2RLO	ohm		State “2” resistance to low level node
S2RHI	ohm		State “2” resistance to high level node
S19NAME			State “19” character abbreviation
S19TSW	sec		State “19” switching time
S19RLO	ohm		Sstate “19” resistance to low level node
S19RHI	ohm		State “19” resistance to high level node
TIMESTEP	sec		Digital input file step size (digital files only)

Digital Outputs

Syntax: analog-to-digital output

```
U<name> <(interface)node> <(reference) node> <(model) name>
+    [SIGNAME = <(digital signal) name>]
```

Examples

```
vref VREFA2D 0 DC 0.0V
uco carry-out_2 VREFA2D a2d signame=12
```

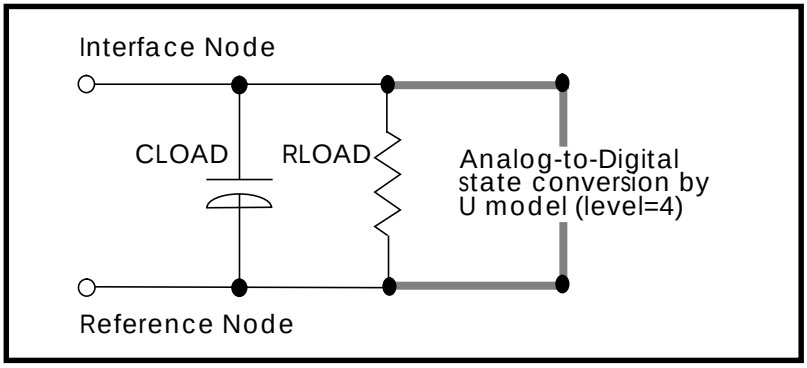

Figure 5-8: Analog-to-Digital Converter Element

Model Form

```
.MODEL < name> U LEVEL=4 [(model parameters)]
```

Examples

```
* DEFAULT DIGITAL OUTPUT MODEL (no “X” value)
.MODEL A2D U LEVEL=4 TIMESTEP=0.1NS TIMESCALE=1
+ S0NAME=0 S0VLO=-1 S0VHI= 2.7
+ S4NAME=1 S4VLO= 1.4 S4VHI=9.0
```

```
+ CLOAD=0.05pf
```

Analog-to-Digital Output Model Parameters

Name(Alias)	Units	Default	Description
RLOAD	ohm	1/gmin	Output resistor
CLOAD	farad	0	Output capacitor
CHGONLY		0	0: write each timestep, 1: write upon change
SONAME			State “0” character abbreviation
SOVLO	volt		State “0” low level voltage
SOVHI	volt		State “0” high level voltage
S1NAME			State “1” character abbreviation
S1VLO	volt		State “1” low level voltage
S1VHI	volt		Sstate “1” high level voltage
S2NAME			State “2” character abbreviation
S2VLO	volt		State “2” low level voltage
S2VHI	volt		State “2” high level voltage
S19NAME			State “19” character abbreviation
S19VLO	volt		State “19” low level voltage
S19VHI	volt		State “19” high level voltage
TIMESTEP	sec	1E-9	Digital input file step size
TIMESCALE			Scale factor for

Two-Bit Adder with Digital I/O

The following two-bit MOS adder uses the digital input file. In the following plot, nodes ‘A[0], A[1], B[0], B[1], and CARRY-IN’ all come from a digital file input. SPICE outputs a digital file.

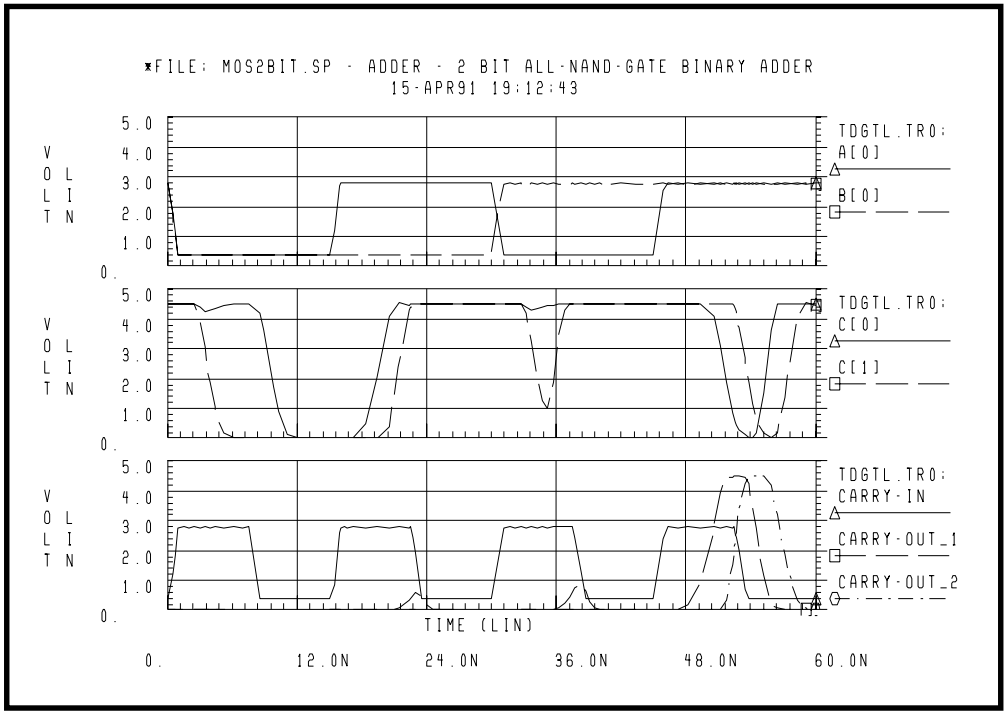

Figure 5-9: Digital Stimulus File Input

Replacing Sources With Digital Inputs

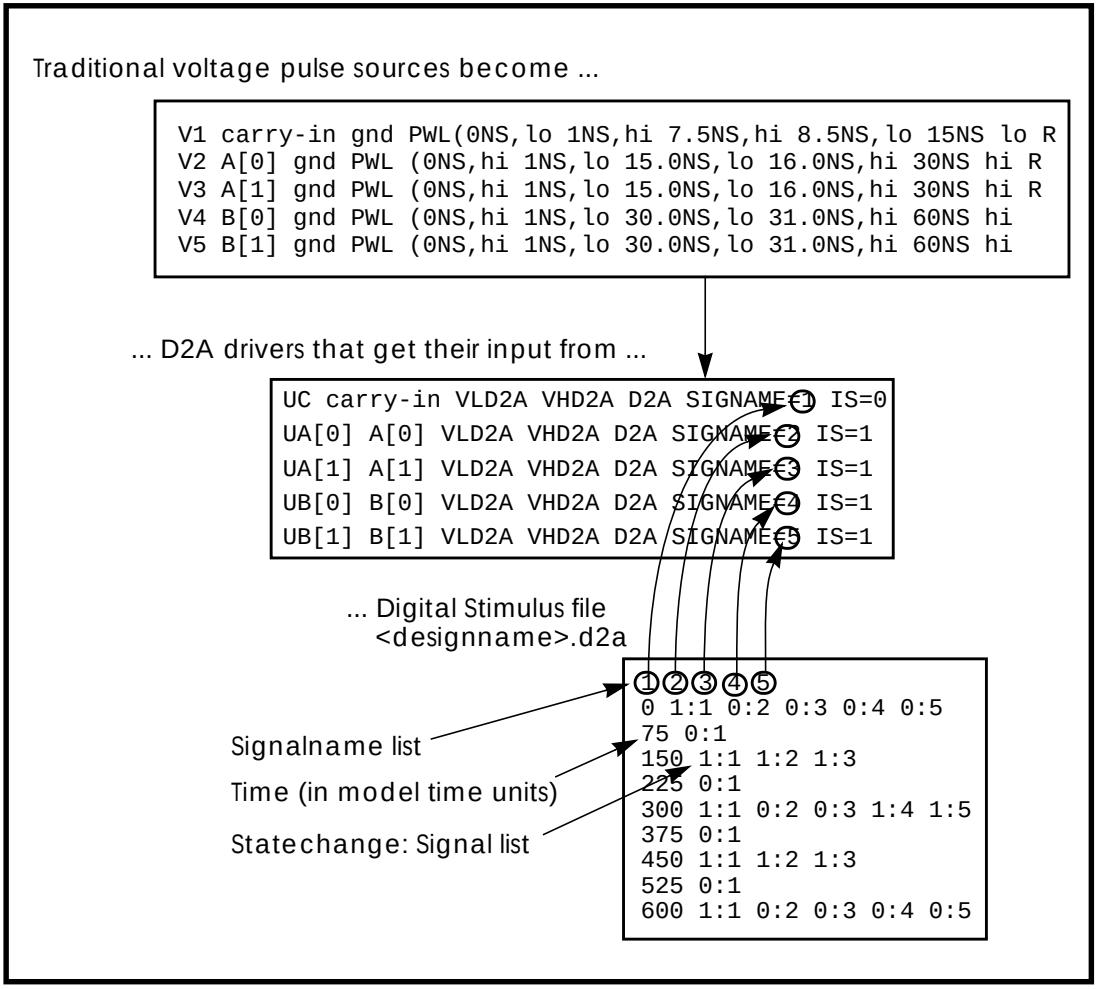

Figure 5-10: Digital File Signal Correspondence

Example of MOS 2 Bit Adder

```
FILE: MOS2BIT.SP - ADDER - 2 BIT ALL-NAND-GATE BINARY ADDER
*
.OPTIONS ACCT NOMOD FAST scale=1u gmindc=100n post
.param lmin=1.25 hi=2.8v lo=.4v vdd=4.5
.global vdd

.TRAN .5NS 60NS
.MEAS PROP-DELAY TRIG V(carry-in) TD=10NS VAL='vdd*.5' RISE=1
+ TARG V(c[1]) TD=10NS VAL='vdd*.5' RISE=3
*
.MEAS PULSE-WIDTH TRIG V(carry-out_1) VAL='vdd*.5' RISE=1
+ TARG V(carry-out_1) VAL='vdd*.5' FALL=1
*
.MEAS FALL-TIME TRIG V(c[1]) TD=32NS VAL='vdd*.9' FALL=1
+ TARG V(c[1]) TD=32NS VAL='vdd*.1' FALL=1

VDD vdd gnd DC vdd
X1 A[0] B[0] carry-in C[0] carry-out_1 ONEBIT
X2 A[1] B[1] carry-out_1 C[1] carry-out_2 ONEBIT
```

Subcircuit Definitions

```
.subckt NAND in1 in2 out wp=10 wn=5
   M1 out in1 vdd vdd P W=wp L=lmin ad=0
   M2 out in2 vdd vdd P W=wp L=lmin ad=0
   M3 out in1 mid gnd N W=wn L=lmin as=0
   M4 mid in2 gnd gnd N W=wn L=lmin ad=0
   CLOAD out gnd 'wp*5.7f'
.ends

.subckt ONEBIT in1 in2 carry-in out carry-out
   X1 in1 in2 #1_nand NAND
   X2 in1 #1_nand 8 NAND
   X3 in2 #1_nand 9 NAND
   X4 8 9 10 NAND
   X5 carry-in 10 half1 NAND
   X6 carry-in half1 half2 NAND
   X7 10 half1 13 NAND
   X8 half2 13 out NAND
   X9 half1 #1_nand carry-out NAND
.ENDS ONEBIT
```

Stimulus

```
UC carry-in VLD2A VHD2A D2A SIGNAME=1 IS=0
UA[0] A[0] VLD2A VHD2A D2A SIGNAME=2 IS=1
UA[1] A[1] VLD2A VHD2A D2A SIGNAME=3 IS=1
UB[0] B[0] VLD2A VHD2A D2A SIGNAME=4 IS=1
UB[1] B[1] VLD2A VHD2A D2A SIGNAME=5 IS=1

*
uc0 c[0] vrefa2d a2d signame=10
uc1 c[1] vrefa2d a2d signame=11
uco carry-out_2 vrefa2d a2d signame=12
uci carry-in vrefa2d a2d signame=13
```

Models

```
.MODEL N NMOS LEVEL=3 VTO=0.7 UO=500 KAPPA=.25 KP=30U
+ ETA=.01 THETA=.04 VMAX=2E5 NSUB=9E16 TOX=400 GAMMA=1.5
+ PB=0.6 JS=.1M XJ=0.5U LD=0.1U NFS=1E11 NSS=2E10
+ RSH=80 CJ=.3M MJ=0.5 CJSW=.1N MJSW=0.3
+ acm=2 capop=4
*
.MODEL P PMOS LEVEL=3 VTO=-0.8 UO=150 KAPPA=.25 KP=15U
+ ETA=.015 THETA=.04 VMAX=5E4 NSUB=1.8E16 TOX=400 GAMMA=.672
+ PB=0.6 JS=.1M XJ=0.5U LD=0.15U NFS=1E11 NSS=2E10
+ RSH=80 CJ=.3M MJ=0.5 CJSW=.1N MJSW=0.3
+ acm=2 capop=4
```

Default Digital Input Interface Model

```
.MODEL D2A U LEVEL=5 TIMESTEP=0.1NS,
+ S0NAME=0 S0TSW=1NS S0RLO = 15, S0RHI = 10K,
+ S2NAME=x S2TSW=5NS S2RLO = 1K, S2RHI = 1K
+ S3NAME=z S3TSW=5NS S3RLO = 1MEG,S3RHI = 1MEG
+ S4NAME=1 S4TSW=1NS S4RLO = 10K, S4RHI = 60
VLD2A VLD2A 0 DC lo
VHD2A VHD2A 0 DC hi
```

Default Digital Output Model (no “X” value)

```
.MODEL A2D U LEVEL=4 TIMESTEP=0.1NS TIMESCALE=1
+ S0NAME=0 S0VLO=-1 S0VHI= 2.7
+ S4NAME=1 S4VLO= 1.4 S4VHI=6.0
+ CLOAD=0.05pf
VREFA2D VREFA2D 0 DC 0.0V

.END
```

Specifying a Digital Vector File

The digital vector file consists of three parts:

- *Vector Pattern Definition* section
- *Waveform Characteristics* section
- *Tabular Data* section.

To incorporate this information into your simulation, you need to include this line in your netlist:

```
.VEC ‘digital_vector_file’
```

Defining Vector Patterns

The *Vector Pattern Definition* section defines the vectors Q — their names, sizes, signal direction, and so on Q — and must occur first in the digital vector file. A sample Vector Pattern Definition section follows:

```
radix 1111 1111
vname a b c d e f g h
io iiii iiii
tunit ns
```

Keywords such as *radix*, *vname* are explained in the "Defining Tabular Data" section later in this chapter.

Defining Waveform Characteristics

The *Waveform Characteristics* section defines various attributes for signals, such as the rise or fall time, thresholds for logic ‘high’ or ‘low’, etc. A sample Waveform Characteristics section follows:

```
trise 0.3 137F 0000
tfall 0.5 137F 0000
vih 5.0 137F 0000
vil 0.0 137F 0000
```

Using Tabular Data

The *Tabular Data* section defines the values of the input signals at specified times. The time is listed in the first column, followed by signal values, in the order specified by the *vname* statement.

Example

An example of tabular data follows:

```
11.0 1000 1000
20.0 1100 1100
33.0 1010 1001
```

Comment Lines

A line beginning with a semi-colon “;” is considered a comment line. Comments may also start at any point along a line. Star-Hspice ignores characters following a semi-colon.

Example

An example of usage follows:

```
; This is a comment line
radix 1 1 4 1234 ; This is a radix line
```

Continuing a Line

Like netlists, a line beginning with a plus sign “+” is a continuation from the previous line.

Digital Vector File Example

An example of a vector pattern definition follows:

```
; specifies # of bits associated with each vector
radix 1 2 444
;*************************************************
*
; defines name for each vector. For multi-bit
; vectors, innermost [] provide the bit index range,
; MSB:LSB
vname v1 va[[1:0]] vb[12:1]
;actual signal names: v1, va[0], va[1], vb1 ... vb12
;*************************************************
*
; defines vector as input, output, or bi-direc
io i o bbb
; defines time unit
tunit ns
;*************************************************
*
; vb12-vb5 are output when ‘v1’ is ‘high’
enable v1 0 0 FF0
; vb4-vb1 are output when ‘v1’ is ‘low’
enable ~v1 0 0 00F
;*************************************************
*
; all signals have delay of 1 ns
; Note: do not put unit (e.g., ns) again here because
; this value will be multiplied by the unit specified
; in the ‘tunit’ line.
tdelay 1.0
```

```
; signals va1 and va0 have delays of 1.5ns
tdelay 1.5 0 3 000
;*************************************************
*
; specify input rise and fall times (if you want
; different rise and fall times, use trise/
; tfallstmt.)
; Note: do not put unit (e.g., ns) again here because
; this value will be multiplied by the unit specified
; in the 'tunit' line.
slope 1.2
;*************************************************
*
; specify the logic 'high' voltage for input signals
vih 3.3 1 0 000
vih 5.0 0 0 FFF
; likewise, may specify logic 'low' with 'vil'
;*************************************************
*
; va & vb switch from 'lo' to 'hi' at 1.75 volts
vth 1.75 0 1 FFF
;*************************************************
*
; tabular data section
10.0 1 3 FFF
20.0 0 2 AFF
30.0 1 0 888
.
.
.
```

Defining Tabular Data

Although this section generally appears last in a digital vector file, following the *Vector Pattern* and *Waveform Characteristics* definitions, we describe it first to introduce the definitions of a *vector*.

The Tabular Data section defines (in *tabular* format) the values of the signals at specified times. Its general format is:

```
time1 signal1_value1 signal2_value1 signal3_value1...
time2 signal1_value2 signal2_value2 signal3_value2...
time3 signal1_value3 signal2_value3 signal3_value3...
.
.
```

The set of values for a particular signal over all times is a *vector*, a vertical column in the tabular data and vector table. Thus, the set of all *signal1_valuex* constitute one vector. Signal values may have the legal states described in the following section.

Rows in the tabular data section must appear in chronological order because row placement carries sequential timing information.

Example

```
10.0 1000 0000
15.0 1100 1100
20.0 1010 1001
30.0 1001 1111
```

This example feature eight signals and therefore eight vectors. The first signal (starting from the left) has a vector [1 1 1 1]; the second has a vector [0 1 0 0]; and so on.

Input Stimuli

Star-Hspice converts each input signal into a PWL (piecewise linear) voltage source and a series resistance. The legal states for an input signal are.:

0	Drive to ZERO (gnd)
1	Drive to ONE (vdd)
Z, z	Floating to HIGH IMPEDANCE
X, x	Drive to ZERO (gnd)
L	Resistive drive to ZERO (gnd)
H	Resistive drive to ONE (vdd)
U, u	Drive to ZERO (gnd)

For the 0, 1, X, x, U, u states, the resistance value is set to zero; for the L, H states, the resistance value is defined by the *out* (or *outz*) statement; and for the Z, z states, the resistance value is defined by the *triz* statement.

Expected Outputs

Star-Hspice converts each output signal into a *.DOUT* statement in the netlist. During simulation, Star-Hspice compares the actual results with the expected output vector(s), and if the states are different, an error message appears. The legal states for expected outputs include:

0	Expect ZERO
1	Expect ONE
X, x	Don’t care
U, u	Don’t care
Z, z	Expect HIGH IMPEDANCE (don’t care)

Z,z are treated as “don’t care” because Star-Hspice cannot detect a high impedance state.

Example

An example of usage follows:

```
...
; start of tabular section data
11.0 1 0 0 1
20.0 1 1 0 0
30.0 1 0 0 0
35.0 x x 0 0
```

Verilog Value Format

Star-Hspice also accepts Verilog *sized* format for number specification:

<size> ’<base format> <number>

The *<size>* specifies (in decimal) the number of bits, and *<base format>* indicates binary (’b or ’B), octal (*’o* or *’O*), or hexadecimal (*’h* or *’H*). Valid *<number>* fields are combinations of the characters *0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F*. Depending on the *<base format>* chosen, only a subset of these characters may be legal.

You may also use unknown values (X) and high impedance (Z) in the *<number>* field. An X or Z sets four bits in the hexadecimal base, three bits in the octal base, and one bit in the binary base.

If the most significant bit of a number is 0, X, or Z, the number is automatically extended (if necessary) to fill the remaining bits with (respectively) 0, X, or Z. If the most significant bit is 1, it is extended with 0.

Examples

```
4’b1111
12’hABx
32’bZ
8’h1
```

Here we specify values for: a 4-bit signal in binary, a 12-bit signal in hexadecimal, a 32-bit signal in binary, and an 8-bit signal in hexadecimal.

Equivalents of these lines in non-Verilog format would be:

```
1111
AB xxxx
ZZZZ ZZZZ ZZZZ ZZZZ ZZZZ ZZZZ ZZZZ ZZZZ
1000 0000
```

Periodic Tabular Data

Very often tabular data is periodic, so it unnecessary to specify the absolute time at every time point. When a user specifies the *period* statement, the *tabular data* section omits the absolute times (see “Using Tabular Data” on page 56 for details).

Example

```
radix 1111 1111
vname a b c d e f g h
io iiii iiii
tunit ns
period 10
; start of vector data section
1000 1000
1100 1100
1010 1001
```

Defining Vector Patterns

The *Vector Pattern Definition* section defines the sequence or order for each vector stimulus, as well as any individual characteristics. The statements in this section (except the *radix* statement) might appear in any order, and all keywords are case-insensitive.

The Radix Statement

The *radix* statement specifies the number of bits associated with each vector. Valid values for the number of bits range from 1 to 4.

# bits	Radix	Number System	Valid Digits
1	2	Binary	0, 1
2	4	–	0 – 3
3	8	Octal	0 – 7
4	16	Hexadecimal	0 – F

Only one *radix* statement must appear in the file, and it must be the first noncomment line.

Example

This example illustrates two 1-bit signals followed by a 4-bit signal, followed by a 1-bit, 2-bit, 3-bit, 4-bit signals, and finally eight 1-bit signals.

```
; start of vector pattern definition section
radix 1 1 4 1234 1111 1111
```

The Vname Statement

The *vname* statement defines the name of each vector. If not specified, a default name will be given to each signal: V1, V2, V3, and so on. If you define more than one *vname* statement, the last one overrules previous one.

```
radix 1 1 1 1 1 1 1 1 1 1 1 1
vname V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12
```

Provide the range of the bit indices with a square bracket [] and a colon syntax:

```
[starting_index : ending index]
```

Tne *vname* name is required for each bit, and a single name may be associated with multiple bits (*such as* bus notation).

The bit order is MSB:LSB. This bus notation syntax may also be nested inside other grouping symbols such as <>, (), [], etc. The name of each bit will be *vname* with the index suffix appended.

Example 1

If you specify:

```
radix 2 4
vname VA[0:1] VB[4:1]
```

the resulting names of the voltage sources generated are:

```
VA0 VA1 VB4 VB3 VB2 VB1
```

where *VA0* and *VB4* are the MSBs and *VA1* and *VB1* are the LSBs.

Example 2

If you specify:

```
vname VA[[0:1]] VB<[4:1]>
```

the resulting names of the voltage sources are:

```
VA[0] VA[1] VB<4> VB<3> VB<2> VB<1>
```

Example 3

This example shows how to specify a single bit of a bus:

```
vname VA[[2:2]]
```

Example 4

This example generates signals A0, A1, A2, ... A23:

```
radix 444444
vname A[0:23]
```

The IO Statement

The *io* statement defines the type of each vector. The line starts with a keyword *io* and followed by a string of i, b, o, or u definitions indicating whether each corresponding vector is an input, bi-directional, output, or unused vector, respectively.

i	Input used to stimulate the circuit.
o	Expected output used to compare with the simulated outputs.
b	Star-Hspice ignores.

Example

If the *io* statement is not specified, all signals are assumed input signals. If you define more than one *io* statements, the last one overrules previous ones.

```
io i i i bbbb iiiioouu
```

The Tunit Statement

The *tunit* statement defines the time unit in digital vector file for *period*, *tdelay*, *slope*, *trise*, *tfall*, and *absolute time*. It must be::

fs	femto-second
ps	pico-second
ns	nano-second
us	micro-second
ms	milli-second

If you do not specify the *tunit* statement, the default time unit value is **ns**. If you define more than one *tunit* statement, the last one will overrule the previous one.

Example

The *tunit* statement in this example specifies that the absolute times in the *tabular data* section are 11.0ns, 20.0ns, and 33.0ns, respectively.

```
tunit ns
11.0 1000 1000
20.0 1100 1100
33.0 1010 1001
```

The Period and Tskip Statements

The *period* statement defines the time interval for the *tabular data* section so that specifying the absolute time at every time point is not necessary. Thus, if a *period* statement is provided alone (without the *tskip* statement), the *tabular data* section contains only signal values, not absolute times. The time unit of *period* is defined by the *tunit* statement.

Example

In this example, the first row of the tabular data (1000 1000) is for time 0ns. The second row of the tabular data (1100 1100) is for time 10ns. The third row of the tabular data (1010 1001) is for time 20ns.

```
radix 1111 1111
period 10
1000 1000
1100 1100
1010 1001
```

The *tskip* statement specifies that the absolute time field in the tabular data is to be ignored. In this way, the absolute time field of each row may be kept in the tabular data (but ignored) when using the *period* statement.

Example

If you have:

```
radix 1111 1111
period 10
tskip
11.0 1000 1000
20.0 1100 1100
33.0 1010 1001
```

the absolute times 11.0, 20.0 and 33.0 are ignored.

The Enable Statement

The *enable* statement specifies the controlling signal(s) of bi-directional signals and is absolutely required for all bi-directional signals. If more than one *enable* statement exists, the last value will overrule the previous ones, with a warning message will be issued.

The syntax is a keyword *enable*, followed by the controlling signal name and the mask that defines the (bi-directional) signals to which *enable* applies.

The controlling signal of bi-directional signals must be an input signal with radix of 1. The bi-directional signals become output when the controlling signal is at state 1 (or high). If you wish to reverse this default control logic, you must start the control signal name with '~'.

Example

In this example, signals *x* and *y* are bi-directional, as defined by the 'b' in the *io* line. The first enable statement indicates that *x* (as defined by the position of 'F') becomes output when signal *a* is 1. The second enable specifies that bi-directional bus *y* becomes output when signal *a* is 0.

```
radix 144
io ibb
vname a x[3:0] y[3:0]
enable a 0 F 0
enable ~a 0 0 F
```

Modifying Waveform Characteristics

This section describes how to modify the waveform characteristics of your circuit.

The Tdelay, Idelay, and Odelay Statements

The *tdelay*, *idelay* and *odelay* statements define the delay time of the signal relative to the absolute time of each row in the *tabular data* section; *idelay* applies to the input signals, *odelay* applies to the output signals, while *tdelay* applies to both input and output signals.

The statement starts with a keyword *tdelay* (or *idelay*, *odelay*) followed by a delay value, and then followed by a *mask,* which defines the signals to which the delay will be applied. If you do not provide a mask, the delay value will be applied to all the signals.

The time unit of *tdelay*, *idelay* and *odelay* is defined by the *tunit* statement. Normally, you only need to use the *tdelay* statement; only use the *idelay* and *odelay* statements to specify different input and output delay times for bi-directional signals. *idelay* settings on output signals (or *odelay* settings on input signals) are ignored with warning message issued.

More than one *tdelay* (*idelay*, *odelay*) statement can be specified. If more than one *tdelay* (*idelay*, *odelay*) statement is applied to a signal, the last value will overrule the previous ones, and a warning will be given. If you do not specify the signal delays by a *tdelay* (*idelay* or *odelay*) statement, Star-Hspice defaults to zero.

Example

The first *tdelay* statement indicates that all signals have the same delay time 1.0. The delay time of some signals are overruled by the subsequent *tdelay* statements. The V2 and Vx signals have delay time -1.2, and V4 V5[0:1] V6[0:2] have a delay of 1.5. The V7[0:3] signals have an input delay time of 2.0 and an output delay time of 3.0.

```
radix 1 1 4 1234 11111111
io i i o iiib iiiiiiii
vname V1 V2 VX[3:0] V4 V5[1:0] V6[0:2] V7[0:3]
+V8 V9 V10 V11 V12 V13 V14 V15
tdelay 1.0
tdelay -1.2 0 1 1 0000 00000000
tdelay 1.5 0 0 0 1370 00000000
idelay 2.0 0 0 0 000F 00000000
odelay 3.0 0 0 0 000F 00000000
```

The Slope Statement

The *slope* statement specifies input signal rise/fall time, with the time unit defined by the *tunit* statement. You can specify the signals to which the *slope* applies using a mask. If the *slope* statement is not provided, the default slope value is 0.1 ns.

If you specify more than one *slope* statement, the last value will overrule the previous ones, and a warning message will be issued. The *slope* statement has no effect on the expected output signals. The rising time and falling time of a signal will be overruled if *trise* and *tfall* are specified.

Examples

The first example indicates that the rising and falling times of all signals are 1.2 ns, whereas the second specifies a rising/falling time of 1.1 ns for the first, second, sixth, and seventh signal.

```
slope 1.2
slope 1.1 1100 0110
```

The Trise Statement

The *trise* statement specifies the rise time of each input signal (for which the mask applies). The time unit of *trise* is defined by the *tunit* statement.

Example

If you do not specify the rising time of the signals by any *trise* statement, the value defined by the *slope* statement is used. If you apply more than one *trise* statements to a signal, the last value will overrule the previous ones, and a warning message will be issued.

```
trise 0.3
trise 0.5 0 1 1 137F 00000000
trise 0.8 0 0 0 0000 11110000
```

The *trise* statements have no effect on the expected output signals.

The Tfall Statement

The *tfall* statement specifies the falling time of each input signal (for which the mask applies). The time unit of *tfall* is defined by the *tunit* statement.

Example

If you do not specify the falling time of the signals by a *tfall* statement, Star-Hspice uses the value defined by the *slope* statement. If you specify more than one *tfall* statement to a signal, the last value will overrule the previous ones, and a warning message will be issued.

```
tfall 0.5
tfall 0.3 0 1 1 137F 00000000
tfall 0.9 0 0 0 0000 11110000
```

The *tfall* statements have no effect on the expected output signals.

The Out /Outz Statements

The keywords *out* and *outz* are equivalent and specify the output resistance of each signal (for which the mask applies); *out* (or *outz*) applies to the input signals only.

Example

If you do not specify the output resistance of a signal by an *out* (or *outz*) statement, Star-Hspice uses the default (zero). If you specify more than one *out* (or *outz*) statement to a signal, Star-Hspice overrules the last value with the previous ones, and issues a warning message.

```
out 15.1
out 150 1 1 1 0000 00000000
outz 50.5 0 0 0 137F 00000000
```

The *out* (or *outz*) statements have no effect on the expected output signals.

The Triz Statement

The *triz* statement specifies the output impedance when the signal (for which the mask applies) is in *tristate*; *triz* applies to the input signals only.

Example

If you do not specify the *tristate* impedance of a signal by a *triz* statement, Star-Hspice assumes 1000M. If you apply more than one *triz* statement to a signal, the last value will overrule the previous ones, and Star-Hspice will issue a warning.

```
triz 15.1M
triz 150M 1 1 1 0000 00000000
triz 50.5M 0 0 0 137F 00000000
```

The *triz* statements have no effect on the expected output signals.

The Vih Statement

The *vih* statement specifies the logic high voltage of each input signal to which the mask applies.

Example

If you specify the logic high voltage of the signals by a *vih* statement, Star-Hspice assumes 3.3. If you apply more than one *vih* statements to a signal, the last value will overrule the previous ones, and Star-Hspice will issue a warning.

```
vih 5.0
vih 5.0 1 1 1 137F 00000000
vih 3.5 0 0 0 0000 11111111
```

The *vih* statements have no effect on the expected output signals.

The Vil Statement

The *vil* statement specifies the logic low voltage of each input signal to which the mask applies.

Example

If you specify the logic low voltage of the signals by a *vil* statement, Star-Hspice assumes 0.0. If you apply more than one *vil* statement to a signal, the last value will overrule the previous ones, and Star-Hspice will issue a warning.

```
vil 0.0
vil 0.0 1 1 1 137F 11111111
```

The *vil* statements have no effect on the expected output signals.

The Vref Statement

Similar to the *tdelay* statement, the *vref* statement specifies the name of the reference voltage for each input vector to which the mask applies; *vref* applies to the input signals only.

Example

If you have:

```
vname v1 v2 v3 v4 v5[1:0] v6[2:0] v7[0:3] v8 v9 v10
vref 0
vref 0 111 137F 000
vref vss 0 0 0 0000 111
```

When Star-Hspice implements it into the netlist, the voltage source realizes *v1*:

```
v1 V1 0 pwl(......)
```

as will *v2*, *v3*, *v4*, *v5*, *v6*, and *v7*. However, *v8* will be realized by

```
V8 V8 vss pwl(......)
```

as will *v9* and *v10*.

If you do not specify the reference voltage name of the signals by a *vref* statement, Star-Hspice assumes 0. If you apply more than one *vref* statement, the last value will overrule the previous ones, and Star-Hspice issues a warning. The *vref* statements have no effect on the output signals.

The Vth Statement

Similar to the *tdelay* statement, the *vth* statement specifies the logic threshold voltage of each signals to which the mask applies; *vth* applies to the output signals only. The threshold voltage is used to decide the logic state of Star-Hspice's output signals for comparison with the expected output signals.

Example

If you do not specify the threshold voltage of the signals by a *vth* statement, Star-Hspice assumes 1.65. If you apply more than one *vth* statements to a signal, the last value will overrule the previous ones, and Star-Hspice issues a warning.

```
vth 1.75
vth 2.5 1 1 1 137F 00000000
vth 1.75 0 0 0 0000 11111111
```

The *vth* statements have no effect on the input signals.

The Voh Statement

The *voh* statement specifies the logic high voltage of each output signal to which the mask applies.

Example

If you do not specify the logic high voltage by a *voh* statement, Star-Hspice assumes 3.3. If you apply more than one *voh* statements to a signal, the last value will overrule the previous ones and Star-Hspice issues a warning.

```
voh 4.75
voh 4.5 1 1 1 137F 00000000
voh 3.5 0 0 0 0000 11111111
```

The *voh* statements have no effect on input signals.

Note: If both voh and vol are not defined, Star-Hspice uses vth (default or defined).

The Vol Statement

The *vol* statement specifies the logic low voltage of each output signal to which the mask applies.

Example

If you do not specify the logic low voltage by a *vol* statement, Star-Hspice assumes 0.0. If you apply more than one *vol* statements to a signal, the last value will overrule the previous ones and Star-Hspice issues a warning.

```
vol 0.5
vol 0.5 1 1 1 137F 11111111
```

The *vol* statements have no effect on input signals.

Note: If both voh and vol are not defined, Star-Hspice uses vth (default or defined)