

NL5

User's Reference

Components
Operators
Functions
Script commands
HTTP functions

Ver.3.19



VERSION

NL5 User's Reference version 3.19.32, 04/26/2025

The latest versions of NL5 documents can be found at sidelinesoft.com/nl5.

LIMITED LIABILITY

NL5, together with all accompanying materials, is provided on a “as is” basis, without warranty of any kind. The author makes no warranty, either expressed, implied, or stationery, including but not limited to any implied warranties of merchantability or fitness for any purpose. In no event will the author be liable to anyone for direct, incidental or consequential damages or losses arising from use or inability to use NL5.

COPYRIGHTS

© 2025, A.Smirnov. The program and User's Manual are copyrighted. No portion of this Manual can be translated or reproduced for commercial purposes without the express written permission from the copyright holder. On publication of results obtained from use of NL5 citation is appreciated.

“**Smith**” is a registered trademark of Analog Instruments Company, New Providence, NJ. **Microsoft**, **Windows**, and **Microsoft Visual C++** are registered trademarks of Microsoft Corporation. **MATLAB** is a registered trademark of The MathWorks, Inc. **PYTHON** is a registered trademark of the Python Software Foundation. **Borland C++ Builder** is a registered trademark of Borland Corporation. **National Instruments** is a registered trademark of National Instruments Corporation. Built with **Indy** (www.indyproject.com). **Verilog** is a registered trademark of Cadence Design Systems. **Xilinx** and **Vivado** are registered trademarks of Xilinx.

Table of Contents

Components.....	8
SubCir model	9
Label	10
A – Amperemeter.....	16
C – Capacitor	17
C – Voltage controlled capacitor	18
C – Current controlled capacitor.....	19
D – Diode.....	20
D – Zener	23
D – Bidirectional zener	24
D – Bridge rectifier	25
D – Diode ring	26
D – Logic controlled thyristor.....	27
D – Voltage controlled thyristor	28
D – Current controlled thyristor.....	29
F – Function	30
F – Function with clock	33
F – Function-2.....	36
F – Function-2 with clock.....	40
F – Custom function.....	44
F – Integral.....	46
F – Integral with reset	46
F – s-function	47
F – z-function.....	51
I – Current source.....	52
I – Logic controlled current source	58
I – Voltage controlled current source.....	62
I – Current controlled current source	65
L – Inductor.....	68
L – Voltage controlled inductor.....	69
L – Current controlled inductor	70
L – Coupled inductors.....	71
L – Custom coupled inductors	72
O – Amplifier.....	74
O – Differential amplifier	77
O – Differential amplifier with reference	80
O – Fully differential amplifier.....	81
O – Differential amplifier with current output.....	82
O – Differential amplifier with differential current output.....	85
O – Current amplifier.....	88
O – Current amplifier with current output	91
O – Summing amplifier	95
O – Voltage controlled amplifier	98
O – Current controlled amplifier.....	99
R – Resistor.....	100

R – Potentiometer	101
R – Voltage controlled resistor	102
R – Current controlled resistor.....	103
S – Switch	104
S – Logic controlled switch	109
S – Voltage controlled switch.....	113
S – Current controlled switch	115
S – SPDT switch	117
S – SPDT logic controlled switch.....	117
S – SPDT voltage controlled switch	118
S – SPDT current controlled switch.....	118
T – NPN transistor	119
T – PNP transistor	121
T – N-FET	123
T – P-FET	127
V – Voltage source.....	131
V – Logic controlled voltage source.....	137
V – Voltage controlled voltage source.....	141
V – Current controlled voltage source	144
V – Voltmeter	147
W – Winding.....	148
W – Transformer	149
W – Differential transformer	150
W – Custom transformer.....	151
W – Wattmeter.....	152
X – Delay	153
X – Transmission line	154
X – Sample/hold	156
X – Directional coupler.....	157
X – Block-2...Block-8.....	158
X – Custom block	159
X – NL5 circuit	160
X – C-code	161
X – DLL.....	163
Y – Gates.....	165
Y – Logical function	167
Y – D flip-flop	169
Y – SR trigger	170
Y – JK trigger.....	171
Y – Schmitt trigger.....	172
Y – Logic generator	173
Y – Logic controlled logic generator.....	178
Y – Voltage controlled logic generator.....	181
Y – Current controlled logic generator	183
Y – Bus	185
Z – Impedance.....	186
Z – Z-meter	189
Operators	190

Functions.....	192
abs, mag	193
sign	193
re, im.....	193
phase.....	194
sqrt.....	194
sqr	194
sq	194
lim, limit	195
islow, ishigh	195
sum	195
mean, average.....	196
min	196
max	197
exp	197
pow	197
pwr	198
log(x,y)	198
ln, log.....	198
lg, log10	198
lb, log2.....	199
db	199
par.....	199
sin, cos, tan, tg.....	199
asin, acos, atan.....	200
atan2.....	200
random, rand.....	200
gauss.....	200
round	201
floor.....	201
ceil.....	201
bool	201
bool <i>C-keyword</i>	202
(bool) <i>type-casting operator</i>	202
int	202
int <i>C-keyword</i>	202
(int) <i>type-casting operator</i>	203
int64.....	203
int64 <i>C-keyword</i>	203
(int64) <i>type-casting operator</i>	203
double.....	204
double <i>C-keyword</i>	204
(double) <i>type-casting operator</i>	204
complex.....	204
complex <i>C-keyword</i>	205
(complex) <i>type-casting operator</i>	205
Script commands.....	206
ac	207

clear	207
close	207
cmd	207
cont	208
cursors	208
display	208
exit	208
export (<i>transient</i>)	209
export (AC)	209
import (<i>transient</i>)	210
logdata	211
open	211
pause	211
ready	211
return	212
rununtil	212
save	212
savedata	212
saveic	213
scope.cmd	213
scope.get	213
scope.getn	213
scope.image	213
scope.log	213
scope.off	213
scope.on	214
scope.read	214
scope.refresh	214
scope.run	214
scope.select	214
scope.single	214
scope.status	214
scope.stop	215
scope.update	215
show	215
silent	215
sleep	215
stop	216
store	216
storetext	216
traces	216
<i>tracename</i> (<i>transient</i>)	217
<i>tracename</i> (AC)	218
tran	219
Script examples	220
HTTP functions	223
NL5_GetInfo	225

NL5_New	225
NL5_Open	225
NL5_Save	225
NL5_Close	225
NL5_Show	225
NL5_GetActive	226
NL5_GetParam	226
NL5_SetParam	226
NL5_DisableCmp	227
NL5_EnableCmp	227
NL5_GetTracesSize	227
NL5_AddTrace	227
NL5_DeleteTrace	228
NL5_GetTrace	228
NL5_GetTraceValue	228
NL5_SetTrace	229
NL5_ShowTran	229
NL5_SetScreen	230
NL5_Cursors	230
NL5_Start	230
NL5_Pause	231
NL5_Continue	231
NL5_Stop	231
NL5_SaveIC	231
NL5_IsRunning	231
NL5_GetDataSize	232
NL5_GetDataAt	232
NL5_GetData	233
NL5_AddData	233
NL5_Export	234
NL5_GetStorageSize	234
NL5_Store	234
NL5_DeleteStorage	235
NL5_GetACTracesSize	235
NL5_AddACTrace	235
NL5_DeleteACTrace	235
NL5_GetACTrace	236
NL5_GetACTraceValue	236
NL5_SetACTrace	237
NL5_ShowAC	237
NL5_SetACScreen	238
NL5_ACCursors	238
NL5_StartAC	239
NL5_StopAC	239
NL5_GetACDataSize	239
NL5_GetACDataAt	240
NL5_GetACStorageSize	240
NL5_StoreAC	240

NL5_DeleteStorage	240
-------------------------	-----

Components


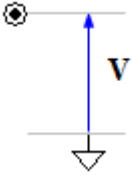
Logical levels and threshold for all components are defined in *Schematic settings/Components* window.

SubCir model

Model	Parameter	Units	Description
SubCir	File		File name of subcircuit schematic.
	Pin1		Name of subcircuit label connected to pin 1

	PinN		Name of subcircuit label connected to pin N
	Cmd		Subcircuit start-up command string
	IC		Subcircuit Initial conditions string
See <i>Working with Subcircuits</i> chapter of The NL5 Manual for details.			

Label

Symbol	Models	Signals
	Label V Step Single Pulse Clock Sin Sweep Function List File IC SubCir	

All models (except SubCir) can be used:

- As a voltage trace probe point.
- For connecting schematic points without wires, including points at different sheets.

Model	Parameter	Units	Description
Label	No parameters.		

Label can be used:

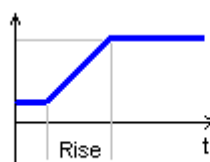
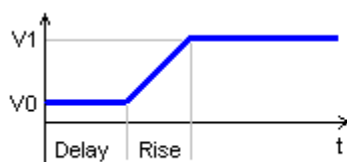
- As a voltage trace probe point.
- For connecting schematic points without wires, including points at different sheets.

Model	Parameter	Units	Description
V	V	V	Voltage.

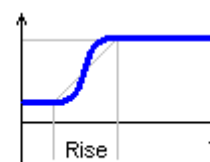
Constant voltage = **V**.

Model	Parameter	Units	Description
Step	V1	V	Step On voltage.
	V0	V	Step Off voltage.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Step rise length.
	Delay	s	Delay before step starts.

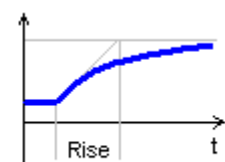
Step starts after **Delay** time. If **Rise** is non-zero, 3 **Slope** types are available.



Linear



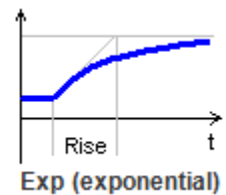
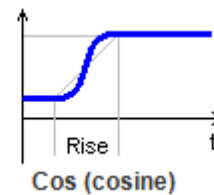
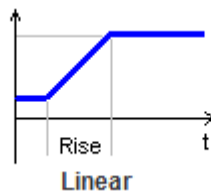
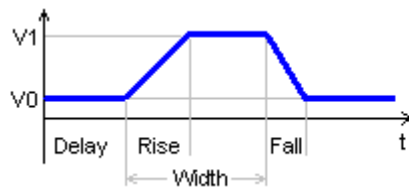
Cos (cosine)



Exp (exponential)

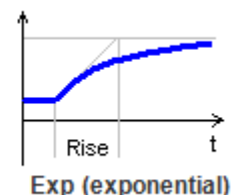
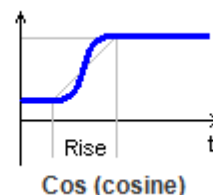
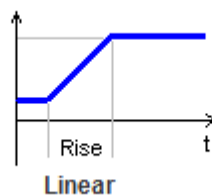
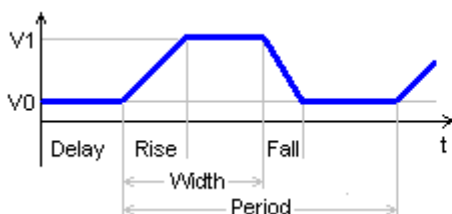
Model	Parameter	Units	Description
Single	V1	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before pulse starts.

Single pulse starts after **Delay** time. **Rise** time is included into **Width**, **Fall** time is **not** included into **Width**. **Slope** type applies both to pulse rise and fall (if non-zero).



Model	Parameter	Units	Description
Pulse	V1	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Period	s	Period.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before first pulse starts.

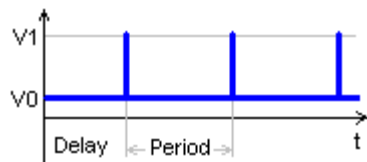
Pulses start after **Delay** time. **Rise** time is included into **Width**, **Fall** time is **not** included into **Width**. **Slope** type applies both to pulse rise and fall (if non-zero).



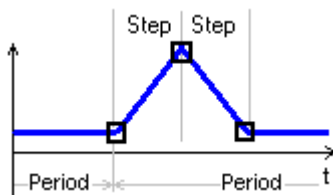
Model	Parameter	Units	Description
Clock	V1	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Period	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.

Periodic pulses with width of one simulation step. Pulses start after **Delay** time.

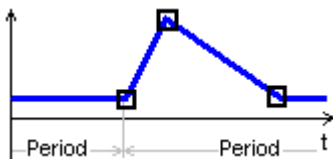
Clock model is recommended to produce a constant frequency clock signal for C-code, DLL, logical components, etc. Unlike **Pulse** model, it won't force unnecessary step reduction at the end of the pulse, which may help to accelerate simulation.



If **Step** parameter is not zero **and** is less than current schematic simulation step, the step is adjusted to provide rise and fall of clock pulse to be equal to **Step**:

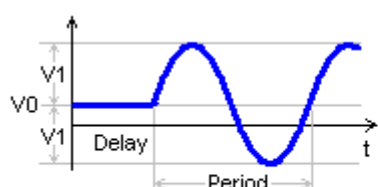


Otherwise, the clock pulse is created using current schematic simulation step, which depends on many factors, and cannot be easily predicted:

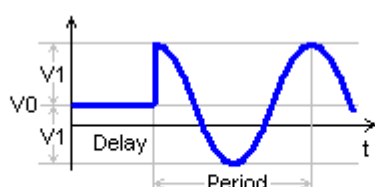


Model	Parameter	Units	Description
Sin	V1	V	Voltage amplitude.
	V0	V	Voltage baseline.
	Period	s	Period.
	Phase	deg	Phase.
	Decay	1/s	Decay constant
	Delay	s	Delay before sine signal starts.

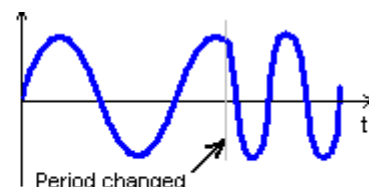
Sinusoidal signal starts after **Delay** time. **Phase** is sine phase in degrees at the moment when signal starts. If transient is paused, sine period changed, and then transient is continued, the phase of the signal remains continuous, providing smooth sine signal of variable frequency. If **Decay** is not zero, the sine signal is exponentially damped with time constant = $1/\text{Decay}$.



Phase = 0



Phase = 90



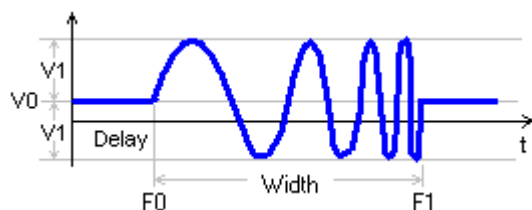
Model	Parameter	Units	Description
Sweep	V1	V	Voltage amplitude.
	V0	V	Voltage baseline.
	Width	s	Width of the signal.
	F0	Hz	Start frequency.
	F1	Hz	End frequency.
	Type		Signal type: Linear/Exp.
	Delay	s	Delay before signal starts.

Sinusoidal signal with variable frequency starts after **Delay** time. Signal frequency changes during **Width** interval from **F0** to **F1** linearly or exponentially, depending on specified **Type**.

If **F0** = **F1**, then one period of frequency $1/\text{Width}$ will be generated.

If lowest frequency is set to zero and **Type** = Exp, then lowest frequency $0.01/\text{Width}$ will be used.

If needed, the highest frequency will be increased to provide integer number of signal periods, so that signal phase at the beginning and at the end of **Width** interval is exactly zero.



Model	Parameter	Units	Description
Function	f	V	Function

Arbitrary function **f** defines voltage as a function of the following variables:

- t** - current time
- V(name)** - voltage on the component **name**
- I(name)** - current through the component **name**
- P(name)** – power on the component **name**
- S(name)** – state of the component **name**

where **name** is the name of the component in the schematic. If **f** is blank, voltage is zero.

Example:

```
f = sin(t) * (1+cos(t*.01))
f = V(R1) * I(R1)
```

Please note that **V**, **I**, **P**, and **S** variables are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
List	List		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Piecewise linear signal is defined by **List** parameter in the **csv** (comma-separated values) format, as follows:

```
t0,V0,t1,V1,...,tn,Vn
```

where all **t** and **V** can be numerical values or expressions.

If $t < t_0$, signal is V_0 .

If $t_0 < t < t_1$, signal value is linearly interpolated between V_0 and V_1 , etc.

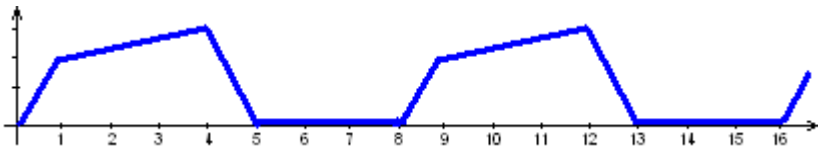
If $t > t_n$, and **Cycle** parameter is set to **No**, the signal value is V_n . Otherwise the signal defined in $t_0 \dots t_n$ interval is repeated continuously.

Signal start is delayed by **Delay** time.

Example:

```
List = 0,0,1,2,4,3,5,0,8,0
```

If **Cycle** = **Yes**, **Delay** = 0, the following voltage will be generated:



See *Working with List source* chapter for more details.

Model	Parameter	Units	Description
File	File		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Piecewise linear signal is defined in the text file. If **File** parameter does not have a full path, NL5 will search for the file in the directory where schematic file is located, then in the Library directories, and then in the Library directories relative to schematic file directory (see NL5 Manual, *Schematic Properties*).

Signal is defined in the **csv** (comma-separated values) format, as follows:

```
<if first line does not start with a number, it is ignored >
t0,V0
t1,V1
.....
tn,Vn
```

where all t and V can be numerical values or expressions.

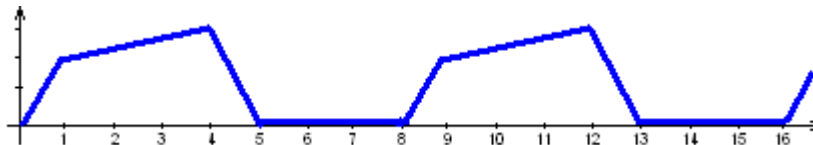
If $t < t_0$, signal is V0. If $t_0 < t < t_1$, signal value is linearly interpolated between V0 and V1, etc. If $t > t_n$, and **Cycle** parameter is set to **No**, the signal value is Vn. otherwise the signal defined in $t_0 \dots t_n$ interval is repeated continuously.

Signal start is delayed by **Delay** time.

Example. File content:

```
0,0
1,2
4,3
5,0
8,0
```


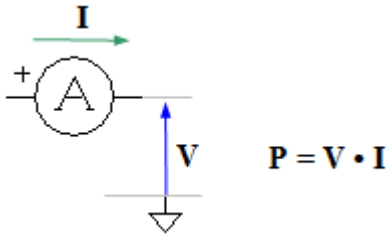
If **Cycle = Yes**, **Delay = 0**, the following voltage will be generated:



Model	Parameter	Units	Description
IC	V	V	Initial voltage.
	R	Ohm	Initial resistance.

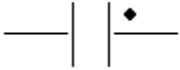
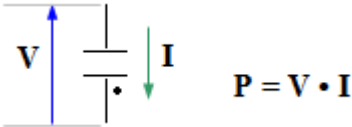
Initial condition. The model is used to apply initial voltage during DC operating point calculation. When calculating DC operating point, the temporary voltage source **V** is connected to the label through initial resistor **R**. When DC operating point is found, the voltage source is removed, and the **IC** model operates similar to **Label** model.

A – Amperemeter

Symbol	Models	Signals
	Amperemeter	

Model	Parameter	Units	Description
Amperemeter	No parameters.		
Short circuit. In addition to current, amperemeter can measure voltage relative to ground, and power delivered to grounded load.			

C – Capacitor

Symbol	Models	Signals
	C PWL SubCir	

Model	Parameter	Units	Description
C	C	F	Capacitance
	IC	V	Initial condition: voltage. Leave blank if IC not defined.
<p>Linear capacitor: $I = C * dV/dt$.</p> <p>When calculating DC operating point, if IC is not blank, capacitor is replaced with voltage source equal to IC. Otherwise, capacitor is temporarily removed (open circuit), DC operating point is calculated, and then the voltage found across the capacitor is assigned to the capacitor as its initial voltage.</p>			

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, C(V)
	IC	V	Initial condition: voltage. Leave blank if IC not defined.
<p>Piecewise constant capacitor: pwl string defines capacitance as a function of voltage across the capacitor C(V). Capacitor charge Q(V) is piecewise linear function, and it always goes through the origin (0,0). See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that C(V) is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p> <p>When calculating DC operating point, if IC is not blank, capacitor is replaced with voltage source equal to IC. Otherwise, capacitor is temporarily removed (open circuit), DC operating point is calculated, and then the voltage found across the capacitor is assigned to the capacitor as its initial voltage.</p>			

C – Voltage controlled capacitor

Symbol	Models	Signals
	PWL	
Views		

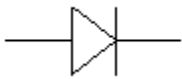
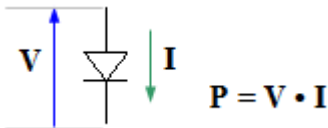
Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, C(Vin)
	IC	V	Initial condition: voltage. Leave blank if IC not defined.
<p>Piecewise constant voltage controlled capacitor. pwl string defines capacitance as a function of control voltage C(Vin). At any moment:</p> $I = C(Vin) * dV/dt.$ <p>See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that C(Vin) is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p> <p>When calculating DC operating point, if IC is not blank, capacitor is replaced with voltage source equal to IC. Otherwise, the capacitor is temporarily removed (open circuit), DC operating point calculated, and then the voltage found across the capacitor is assigned to the capacitor as its initial voltage.</p>			

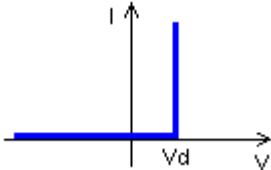
C – Current controlled capacitor

Symbol	Models	Signals
	PWL	
Views		

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, C(lin)
	IC	V	Initial condition: voltage. Leave blank if IC not defined.
<p>Piecewise constant current controlled capacitor. pwl string defines capacitance as a function of control current C(lin). At any moment:</p> $I = C(lin) * dV/dt.$ <p>See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that C(lin) is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p> <p>When calculating DC operating point, if IC is not blank, capacitor is replaced with voltage source equal to IC. Otherwise, the capacitor is temporarily removed (open circuit), DC operating point calculated, and then the voltage found across the capacitor is assigned to the capacitor as its initial voltage</p>			

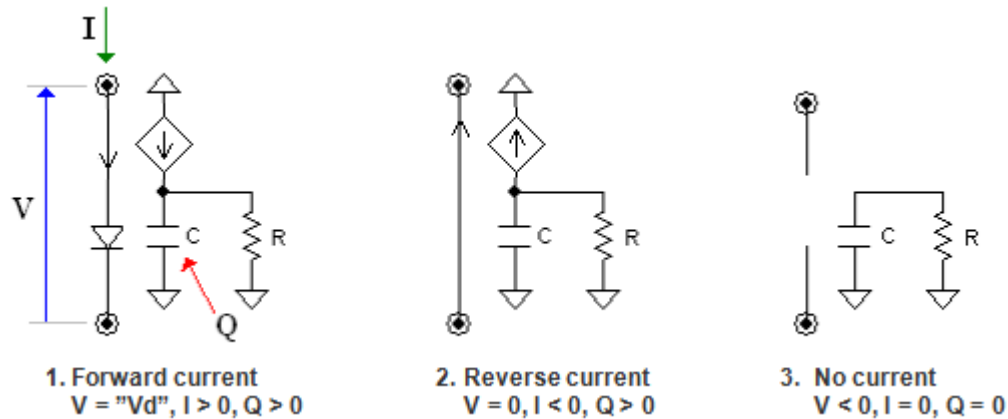
D – Diode

Symbol	Models	Signals
	Diode Storage Soft PWL SubCir	

Model	Parameter	Units	Description
Diode	Vd	V	Forward voltage drop.
	IC		Initial condition: On/Off.
<p>Ideal diode. If $V \geq V_d$, diode is On (short circuit). Otherwise diode is Off (open circuit, $I=0$).</p> <p>When calculating DC operating point diode is set to the state specified in IC.</p> 			

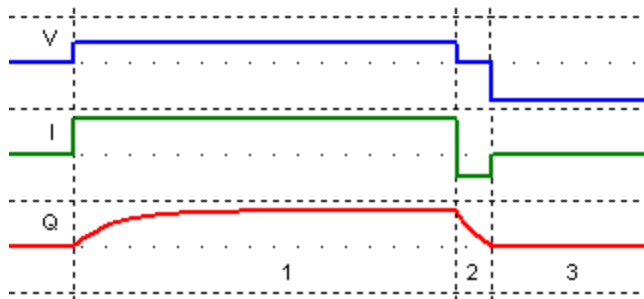
Model	Parameter	Units	Description
Storage	Vd	V	Forward voltage drop.
	t	s	Recombination time constant.
	IC		Initial condition: Off/On.
	ICQ	C (A*s)	Initial condition: charge.

Charge storage diode. Simplified equivalent schematic of the model is the following:



The diode has internal capacitor C and resistor R, with the time constant $RC = t$, Q is the charge on the capacitor.

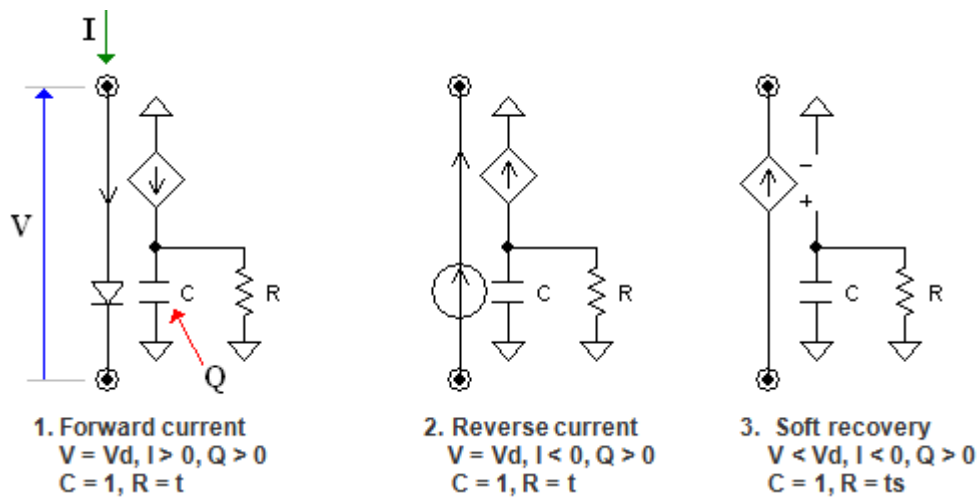
In **mode 1**, forward current flows through the diode and forward voltage drop is **Vd**. At the same time, the current equal to forward current is charging capacitor C. In **mode 2**, reverse current is applied to the diode, and capacitor C is being discharged by the current equal to reverse current. As long as charge Q on the capacitor is positive, the diode is a short circuit with zero voltage drop. Finally, when charge drops to zero, the diode switches to **mode 3**, with zero current and negative voltage drop (open circuit).



When calculating DC operating point the diode is set to the state specified in **IC**, and internal charge Q is set to specified **ICQ** value.

Model	Parameter	Units	Description
Soft	Vd	V	Forward voltage drop.
	t	s	Recombination time constant.
	ts	s	Soft recovery time constant.
	IC		Initial condition: Off/On.
	ICQ	C (A*s)	Initial condition: charge.

Soft recovery charge storage diode. Simplified equivalent schematic of the model is the following:



The diode has internal capacitor $C=1$ and resistor R . Time constant RC is equal either recombination time constant t , or soft recovery time constant $= ts$. Q is the charge on the capacitor. In **mode 1**, forward current flows through the diode and forward voltage drop is V_d . At the same time, the current equal to forward current is charging capacitor C . In **mode 2**, reverse current is applied to the diode, and capacitor C is being discharged by the current equal to reverse current. Voltage drop on the diode is still V_d . At the moment when reverse current is equal or less than charge divided by soft recovery time constant ts , a **mode 3** is turned on. In **mode 3**, capacitor C is being exponentially discharged by the current through resistor R with time constant ts (plus small constant current to ensure full discharge - not shown on the picture). Reverse diode current is proportional to the charge. As soon as charge drops to zero, the diode switches to **mode 4** (not shown), with zero current and negative voltage drop (open circuit).

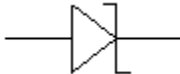
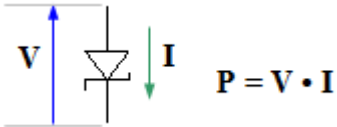
When calculating DC operating point the diode is set to the state specified in **IC**, and internal charge Q is set to specified **ICQ** value.

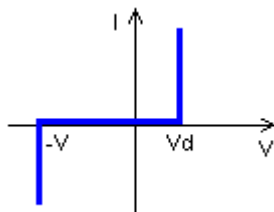
Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, R(V)

Piecewise linear diode. **pwl** string defines resistance as a function of voltage across the diode $R(V)$. Volt-ampere characteristic of the diode is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that $R(V)$ is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

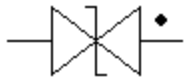
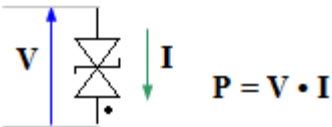
D – Zener

Symbol	Models	Signals
	Zener PWL SubCir	

Model	Parameter	Units	Description
Zener	V	V	Breakdown voltage drop.
	Vd	V	Forward voltage drop.
	IC		Initial condition: Minus/Off/Plus.
<p>Ideal zener. If V (across zener) $\leq -V$ or $V \geq Vd$, zener is On (short circuit). Otherwise zener is Off (open circuit, $I=0$).</p> <p>When calculating DC operating point zener is set to the state specified in IC.</p> 			

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, $R(V)$
<p>Piecewise linear zener. pwl string defines resistance as a function of voltage across zener $R(V)$. Volt-ampere characteristic of zener is piecewise linear function, and it always goes through the origin (0,0). See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that $R(V)$ is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p>			

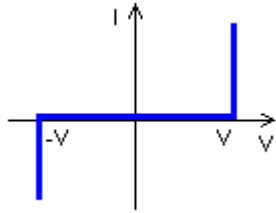
D – Bidirectional zener

Symbol	Models	Signals
	Zener PWL SubCir	

Model	Parameter	Units	Description
Zener	V	V	Breakdown voltage drop.
	IC		Initial condition: Minus/Off/Plus.

Ideal bidirectional zener. If V (across zener) $\leq -V$ or $V \geq V$, zener is On (short circuit). Otherwise zener is Off (open circuit, $I=0$).

When calculating DC operating point zener is set to the state specified in **IC**.

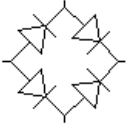


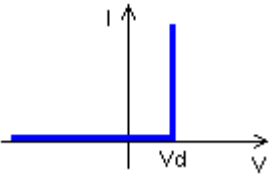
Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, R(V)

Piecewise linear zener. **pwl** string defines resistance as a function of voltage across zener R(V). Volt-ampere characteristic of zener is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.


Please note that R(V) is **piecewise constant** function, although he model and parameter are still called **pwl** for historical reasons.

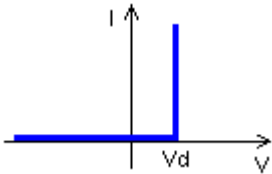
D – Bridge rectifier

Symbol	Models	Signals
	Diode	$P = V_1 \cdot I_1 + V_2 \cdot I_2 + V_3 \cdot I_3 + V_4 \cdot I_4$

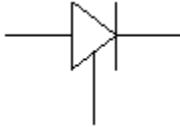
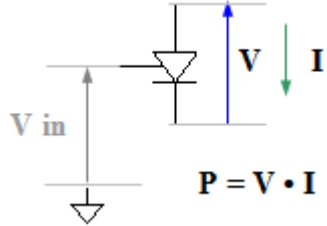
Model	Parameter	Units	Description
Diode	Vd	V	Forward voltage drop.
<p>Bridge rectifier with ideal diodes. For each diode, if $V \geq Vd$, diode is On (short circuit). Otherwise diode is Off (open circuit, $I=0$).</p> <p>When calculating DC operating point all diodes are Off.</p> 			

D – Diode ring

Symbol	Models	Signals
	Diode	$P = V_1 \cdot I_1 + V_2 \cdot I_2 + V_3 \cdot I_3 + V_4 \cdot I_4$

Model	Parameter	Units	Description
Diode	Vd	V	Forward voltage drop.
<p>Diode ring with ideal diodes. For each diode, if $V \geq Vd$, diode is On (short circuit). Otherwise diode is Off (open circuit, $I=0$).</p> <p>When calculating DC operating point all diodes are Off.</p> 			

D – Logic controlled thyristor

Symbol	Models	Signals
	Thyristor SubCir	

Model	Parameter	Units	Description
Thyristor	Vd	V	Forward voltage drop.
	Ihold	A	Holding current.
	IC		Initial condition: Off/On.

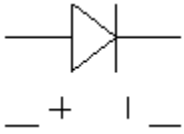
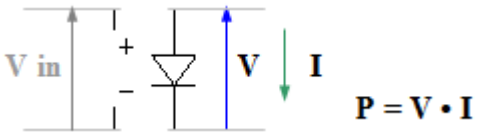
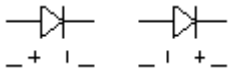
Thyristor has two states:

- Off state (non-conducting): open circuit.
- On state (conducting): ideal diode with **Vd** forward voltage drop.

If control voltage *V_{in}* is greater than logical threshold, thyristor is in On state (ideal diode). When control voltage drops below logical threshold, thyristor stays in On state as long as current *I* exceeds holding current **Ihold**, and voltage *V* is not negative.

When calculating DC operating point thyristor is set to the state specified in **IC**.

D – Voltage controlled thyristor

Symbol	Models	Signals
	Thyristor SubCir	
Views		

Model	Parameter	Units	Description
Thyristor	Vd	V	Forward voltage drop.
	Ihold	A	Holding current.
	Threshold	V	Voltage threshold.
	IC		Initial condition: Off/On.

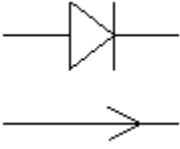
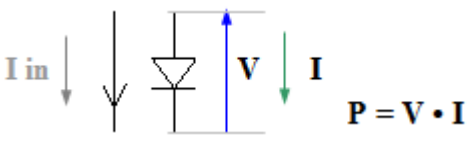
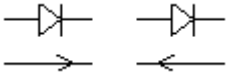
Thyristor has two states:

- Off state (non-conducting): open circuit.
- On state (conducting): ideal diode with **Vd** forward voltage drop.

If control voltage **Vin** is greater than **Threshold**, thyristor is in On state (ideal diode). When control voltage drops below **Threshold**, thyristor stays in On state as long as current **I** exceeds holding current **Ihold**, and voltage **V** is not negative.

When calculating DC operating point thyristor is set to the state specified in **IC**.

D – Current controlled thyristor

Symbol	Models	Signals
	Thyristor SubCir	
Views		

Model	Parameter	Units	Description
Thyristor	Vd	V	Forward voltage drop.
	Ihold	A	Holding current.
	Threshold	A	Current threshold.
	IC		Initial condition: Off/On.

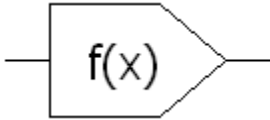
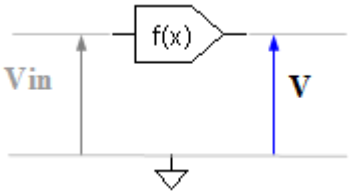
Thyristor has two states:

- Off state (non-conducting): open circuit.
- On state (conducting): ideal diode with **Vd** forward voltage drop.

If control current **I_{in}** is greater than **Threshold**, thyristor is in On state (ideal diode). When control current drops below **Threshold**, thyristor stays in On state as long as current **I** exceeds holding current **Ihold**, and voltage **V** is not negative.

When calculating DC operating point thyristor is set to the state specified in **IC**.

F – Function

Symbol	Models	Signals
	Function Pwr Abs Int	

The function is calculated and applied to the output on every calculation step.

For all models, when calculating DC operating point, and AC analysis, output is set to specified output voltage **IC**. Please note that output voltage is always delayed by one calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
Function	f	V	Output as function of the input.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

x – input voltage V_{in}

t - current time

V(name) - voltage on the component **name**

I(name) - current through the component **name**

P(name) – power on the component **name**

S(name) – state of the component **name**

where **name** is the name of the component in the schematic. If **f** is blank, output is zero.

Examples:

$$f = x * x$$

$$f = x * \sin(t)$$

$$f = P(r1) + P(r2)$$

Model	Parameter	Units	Description
Pwr	power		Power.
	K	V/V	Gain.
	IC	V	Initial condition: output voltage.

“Signed” power function: $V = K * \text{pwr}(Vin, \text{power})$.

The function is calculated as follows:

if **power** = 0:

- if $Vin < 0$. . . : $V = -K$
- if $Vin = 0$. . . : $V = 0$
- if $Vin > 0$. . . : $V = K$

if **power** $\neq 0$:

- if $Vin < 0$. . . : $V = -K * (-Vin)^{\text{power}}$
- if $Vin = 0$. . . : $V = 0$
- if $Vin > 0$. . . : $V = K * Vin^{\text{power}}$

Model	Parameter	Units	Description
Abs	K	V/V	Gain.
	IC	V	Initial condition: output voltage.

Absolute value: $V = K * \text{abs}(Vin)$.

Model	Parameter	Units	Description
Int	resolution	V	Resolution.
	K	V/V	Gain.
	IC	V	Initial condition: output voltage.

Rounding function: $V = K * \text{round}(Vin, \text{resolution})$.

Round to the nearest multiple of **resolution**. If **resolution** = 1, round to the nearest integer.

Model	Parameter	Units	Description
Lim	Max	V	Maximum.
	Min	V	Minimum.
	IC	V	Initial condition: output voltage.

Limiting function is calculated as follows:

- if $Vin < \text{Min}$. . . : $V = \text{Min}$
- if $Vin > \text{Max}$. . . : $V = \text{Max}$
- Otherwise . . . : $V = Vin$

Model	Parameter	Units	Description
Table	Table		Comma-separated string, <i>V_{in}</i> / <i>V_{out}</i> pairs.
	IC	V	Initial condition: output voltage.

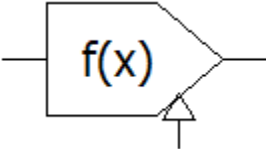
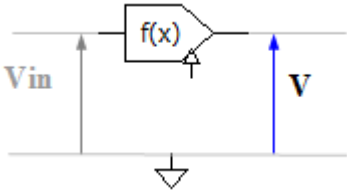
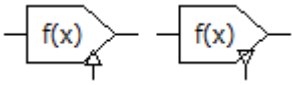
Look-up table. Function output is defined by **Table** parameter in the **csv** (comma separated values) format, as follows:

X1,Y1,X2,Y2,...,XN,YN

where X_i, Y_i pair defines input value (X) and output value (Y). Output value between specified points is linearly interpolated. Output value below X_1 is linearly extrapolated using $X_1 \dots X_2$ interval data, output value above X_N is linearly extrapolated using $X_{(N-1)} \dots X_N$ interval data. Values $X_1 \dots X_N$ should be given in an ascending order.

See *Working with Table model* chapter for more details.

F – Function with clock

Symbol	Models		Signals
	Function Pwr Abs Int	Lim Table SubCir	
<div></div> <p>Function component with clock operates in synchronized mode: the function is calculated and applied to the output only on rising (or falling) edge of logical clock signal. As a result, this mode may provide faster simulation.</p> <p>For all models, when calculating DC operating point, and AC analysis, output is set to specified output voltage IC. Please note that output voltage is always delayed by one calculation step. This may affect stability of the schematic with closed loop.</p>			

Model	Parameter	Units	Description
Function	f	V	Output as function of the input.
	IC	V	Initial condition: output voltage.
<p>Arbitrary function f defines output voltage as a function of the following variables:</p> <ul style="list-style-type: none">x – input voltage <i>Vin</i>t - current timeV(name) - voltage on the component <i>name</i>I(name) - current through the component <i>name</i>P(name) – power on the component <i>name</i>S(name) – state of the component <i>name</i> <p>where <i>name</i> is the name of the component in the schematic. If f is blank, output is zero.</p> <p>Examples:</p> <ul style="list-style-type: none">f = x*xf = x * sin(t)f = P(r1) + P(r2)			

Model	Parameter	Units	Description
Pwr	power		Power.
	K	V/V	Gain.
	IC	V	Initial condition: output voltage.

“Signed” power function: $V = K * \text{pwr}(Vin, \text{power})$.

The function is calculated as follows:

if **power** = 0:

- if $Vin < 0$. . . : $V = -K$
- if $Vin = 0$. . . : $V = 0$
- if $Vin > 0$. . . : $V = K$

if **power** $\neq 0$:

- if $Vin < 0$. . . : $V = -K * (-Vin)^{\text{power}}$
- if $Vin = 0$. . . : $V = 0$
- if $Vin > 0$. . . : $V = K * Vin^{\text{power}}$

Model	Parameter	Units	Description
Abs	K	V/V	Gain.
	IC	V	Initial condition: output voltage.

Absolute value: $V = K * \text{abs}(Vin)$.

Model	Parameter	Units	Description
Int	resolution	V	Resolution.
	K	V/V	Gain.
	IC	V	Initial condition: output voltage.

Rounding function: $V = K * \text{round}(Vin, \text{resolution})$.

Round to the nearest multiple of **resolution**. If **resolution** = 1, round to the nearest integer.

Model	Parameter	Units	Description
Lim	Max	V	Maximum.
	Min	V	Minimum.
	IC	V	Initial condition: output voltage.

Limiting function is calculated as follows:

- if $Vin < \text{Min}$. . . : $V = \text{Min}$
- if $Vin > \text{Max}$. . . : $V = \text{Max}$
- Otherwise . . . : $V = Vin$

Model	Parameter	Units	Description
Table	Table		Comma-separated string, <i>V_{in}</i> / <i>V_{out}</i> pairs.
	IC	V	Initial condition: output voltage.

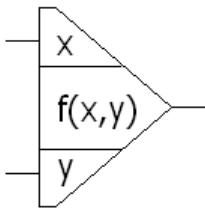
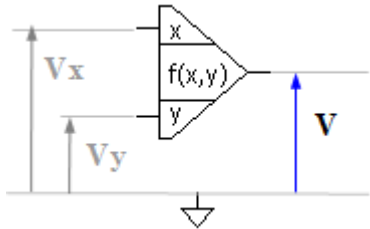
Look-up table. Function output is defined by **Table** parameter in the **csv** (comma separated values) format, as follows:

$X_1, Y_1, X_2, Y_2, \dots, X_N, Y_N$

where X_i, Y_i pair defines input value (X) and output value (Y). Output value between specified points is linearly interpolated. Output value below X_1 is linearly extrapolated using $X_1 \dots X_2$ interval data, output value above X_N is linearly extrapolated using $X_{(N-1)} \dots X_N$ interval data. Values $X_1 \dots X_N$ should be given in ascending order.

See *Working with Table model* chapter of NL5 Manual for more details.

F – Function-2

Symbol	Models		Signals
	Function Mul Div Sum Sub Max Min	Pwr Mag Phase GT LT Table SubCir	

The function is calculated and applied to the output on every calculation step.

For all models, when calculating DC operating point, and AC analysis, output is set to specified output voltage **IC**. Please note that output voltage is always delayed by one calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
Function	f	V	Output as function of the inputs.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

x – input voltage V_x

y – input voltage V_y

t - current time

V(name) - voltage on the component **name**

I(name) - current through the component **name**

P(name) – power on the component **name**

S(name) – state of the component **name**

where **name** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

$f = \sqrt{x^2 + y^2}$

$f = x * y * \sin(t)$

$f = P(r1) + P(r2)$

Model	Parameter	Units	Description
Mul	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
Multiplication: $V = K * V_x * V_y$.			

Model	Parameter	Units	Description
Div	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
Division.: $V = K * V_x / V_y$. If $V_y = 0$, $V = 0$.			

Model	Parameter	Units	Description
Sum	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
Addition: $V = K * (V_x + V_y)$.			

Model	Parameter	Units	Description
Sub	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
Subtraction: $V = K * (V_x - V_y)$.			

Model	Parameter	Units	Description
Max	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
Maximum: $V = K * \max(V_x, V_y)$. if $V_x \geq V_y \dots : V = K * V_x$ if $V_x < V_y \dots : V = K * V_y$			

Model	Parameter	Units	Description
Min	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
Maximum: $V = K * \max(V_x, V_y)$. if $V_x \geq V_y \dots : V = K * V_x$ if $V_x < V_y \dots : V = K * V_y$			

Model	Parameter	Units	Description
Pwr	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
<p>“Signed” power function: $V = K * \text{pwr}(V_x, V_y)$.</p> <p>The function is calculated as follows:</p> <div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>if $V_y = 0$:</p> <p>if $V_x < 0$. . . : $V = -K$</p> <p>if $V_x = 0$. . . : $V = 0$</p> <p>if $V_x > 0$. . . : $V = K$</p> </div> <div style="width: 45%;"> <p>if $V_y \neq 0$:</p> <p>if $V_x < 0$. . . : $V = -K * (-V_x)^{V_y}$</p> <p>if $V_x = 0$. . . : $V = 0$</p> <p>if $V_x > 0$. . . : $V = K * V_x^{V_y}$</p> </div> </div>			

Model	Parameter	Units	Description
Mag	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
<p>Magnitude. $V = K * \text{sqrt}(V_x^2 + V_y^2)$.</p>			

Model	Parameter	Units	Description
Phase	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
<p>Phase: $V = K * \text{phase}(V_x, V_y)$.</p> <p>V in Volts is equal to phase of a vector $V_x + jV_y$ in degrees.</p> <p>If $V_x = 0$ and $V_y = 0$: $V = 0$.</p>			

Model	Parameter	Units	Description
GT	IC	V	Initial condition: output voltage.
<p>Greater than: $V = V_x > V_y$? High : Low.</p> <p>High and Low are logical levels.</p>			

Model	Parameter	Units	Description
LT	IC	V	Initial condition: output voltage.
<p>Less than. $V = V_x < V_y$? High : Low.</p> <p>High and Low are logical levels.</p>			

Model	Parameter	Units	Description
Table	X		Comma-separated string, X (input values).
	Y		Comma-separated string, Y (input values).
	Table		Comma-separated string, Table of Z (output values).
	IC	V	Initial condition: output voltage.

2D look-up table. **Table** parameter defines output Z as a function of X and Y inputs of the component in the following format:

Z11,Z12,...,Z1N,Z21,Z22,...,Z2N,...,ZM1,ZM2,...,ZMN

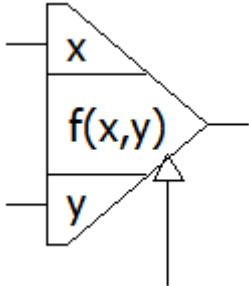
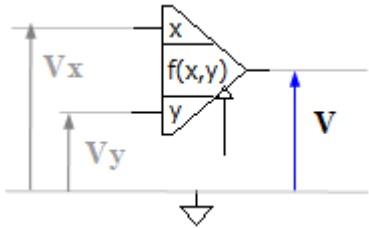
where:

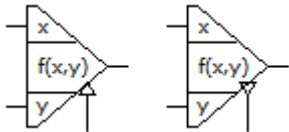
- Zij defines output of the function for input values Xi and Yj;
- N is total number of X input values, defined by **X** parameter;
- M is total number of Y input values, defined by **Y** parameter.

Output value between specified X and Y points is linearly interpolated on both coordinates. Output value below X1 is linearly extrapolated using X1...X2 interval data, output value above XN is linearly extrapolated using X(N-1)...XN interval data. The same rule is applied to Y coordinate

See *Working with 2D Table model* chapter for more details.

F – Function-2 with clock

Symbol	Models		Signals
	Function Mul Div Sum Sub Max Min	Pwr Mag Phase GT LT Table SubCir	



Function component with **clock** operates in **synchronized** mode: the function is calculated and applied to the output only on rising (or falling) edge of logical clock signal. As a result, this mode may provide faster simulation.

For all models, when calculating DC operating point, and AC analysis, output is set to specified output voltage **IC**. Please note that output voltage is always delayed by one calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
Function	f	V	Output as function of the inputs.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

- x** – input voltage V_x
- y** – input voltage V_y
- t** - current time
- V(name)** - voltage on the component **name**
- I(name)** - current through the component **name**
- P(name)** – power on the component **name**
- S(name)** – state of the component **name**

where **name** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

```
f = sqrt(x*x + y*y)
f = x * y * sin(t)
f = P(r1) + P(r2)
```

Model	Parameter	Units	Description
Mul	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
Multiplication: $V = K * V_x * V_y$.			

Model	Parameter	Units	Description
Div	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
Division.: $V = K * V_x / V_y$. If $V_y = 0$, $V = 0$.			

Model	Parameter	Units	Description
Sum	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
Addition: $V = K * (V_x + V_y)$.			

Model	Parameter	Units	Description
Sub	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
Subtraction: $V = K * (V_x - V_y)$.			

Model	Parameter	Units	Description
Max	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
Maximum: $V = K * \max(V_x, V_y)$. if $V_x \geq V_y \dots : V = K * V_x$ if $V_x < V_y \dots : V = K * V_y$			

Model	Parameter	Units	Description
Min	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
Maximum: $V = K * \max(V_x, V_y)$. if $V_x \geq V_y \dots : V = K * V_x$ if $V_x < V_y \dots : V = K * V_y$			

Model	Parameter	Units	Description
Pwr	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
<p>“Signed” power function: $V = K * \text{pwr}(V_x, V_y)$.</p> <p>The function is calculated as follows:</p> <div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>if $V_y = 0$:</p> <p>if $V_x < 0$. . . : $V = -K$</p> <p>if $V_x = 0$. . . : $V = 0$</p> <p>if $V_x > 0$. . . : $V = K$</p> </div> <div style="width: 45%;"> <p>if $V_y \neq 0$:</p> <p>if $V_x < 0$. . . : $V = -K * (-V_x)^{V_y}$</p> <p>if $V_x = 0$. . . : $V = 0$</p> <p>if $V_x > 0$. . . : $V = K * V_x^{V_y}$</p> </div> </div>			

Model	Parameter	Units	Description
Mag	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
<p>Magnitude. $V = K * \text{sqrt}(V_x^2 + V_y^2)$.</p>			

Model	Parameter	Units	Description
Phase	K	V/V	Gain.
	IC	V	Initial condition: output voltage.
<p>Phase: $V = K * \text{phase}(V_x, V_y)$.</p> <p>V in Volts is equal to phase of a vector $V_x + jV_y$ in degrees.</p> <p>If $V_x = 0$ and $V_y = 0$: $V = 0$.</p>			

Model	Parameter	Units	Description
GT	IC	V	Initial condition: output voltage.
<p>Greater than: $V = V_x > V_y$? High : Low.</p> <p>High and Low are logical levels.</p>			

Model	Parameter	Units	Description
LT	IC	V	Initial condition: output voltage.
<p>Less than. $V = V_x < V_y$? High : Low.</p> <p>High and Low are logical levels.</p>			

Model	Parameter	Units	Description
Table	X		Comma-separated string, X (input values).
	Y		Comma-separated string, Y (input values).
	Table		Comma-separated string, Table of Z (output values).
	IC	V	Initial condition: output voltage.

2D look-up table. **Table** parameter defines output Z as a function of X and Y inputs of the component in the following format:

Z11,Z12,...,Z1N,Z21,Z22,...,Z2N,...,ZM1,ZM2,...,ZMN

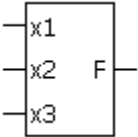
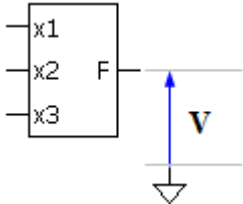
where:

- Zij defines output of the function for input values Xi and Yj;
- N is total number of X input values, defined by **X** parameter;
- M is total number of Y input values, defined by **Y** parameter.

Output value between specified X and Y points is linearly interpolated on both coordinates. Output value below X1 is linearly extrapolated using X1...X2 interval data, output value above XN is linearly extrapolated using X(N-1)...XN interval data. The same rule is applied to Y coordinate

See *Working with 2D Table model* chapter for more details.

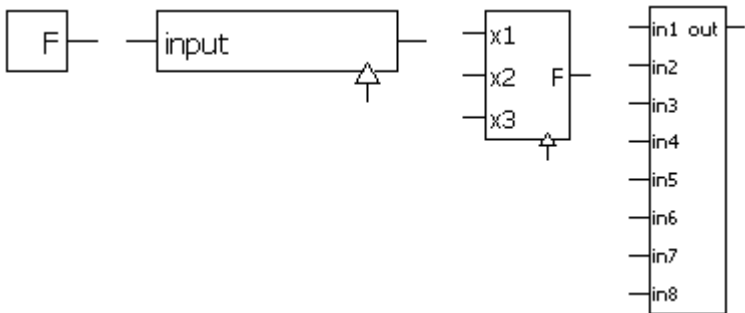
F – Custom function

Symbol	Models	Traces
	Function	

This is a customized component. it can be edited n the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

- This component may have:
- arbitrary size up to 32(width) X 8(height),
 - up to 8 inputs on the left side,
 - one output on the right side,
 - one or no clock pins on the bottom side,
 - custom input and output names.

Examples of Custom function component:



Model	Parameter	Units	Description
Function	f	V	Output as function of the inputs.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

pin_name – input voltage on the input pin **pin_name**.

t - current time

V(name) - voltage on the component **name**

I(name) - current through the component **name**

P(name) – power on the component **name**

S(name) – state of the component **name**

where **name** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

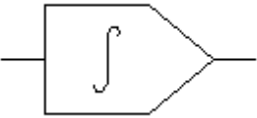
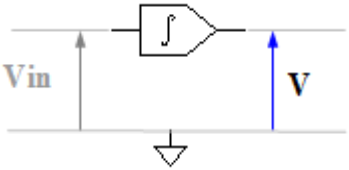
$f = \max(x1, x2, x3)$

$f = (in1+in2) * V(R1)$

If **clock** pin does not exist, the model operates in **continuous** mode: the function is calculated and applied to the output on every calculation step. If **clock** pin exists, the model operates in **synchronized** mode: the function is calculated and applied to the output only on rising (or falling) edge of logical clock signal. As a result, this mode may provide faster simulation than **continuous** mode.

When calculating DC operating point, and in AC analysis, output is set to specified output voltage **IC**. Please note that input voltages and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

F – Integral

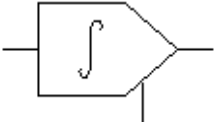
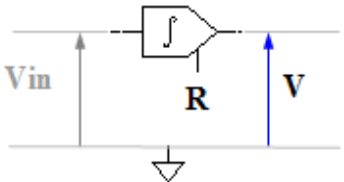
Symbol	Models	Signals
	Integral	

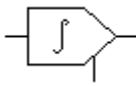
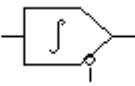
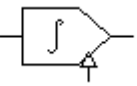

Model	Parameter	Units	Description
Integral	K	V/V	Gain.
	IC	V	Initial condition: output voltage.

Integral: $V = K * \int Vin \, dt.$

When calculating DC operating point, output is set to specified output voltage **IC**.

F – Integral with reset

Symbol	Models	Signals
	Integral	

Views				
-------	---	---	---	---

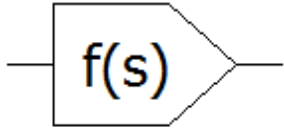
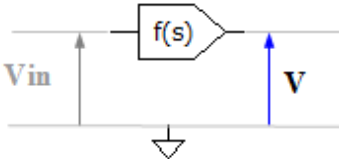
Model	Parameter	Units	Description
Integral	K	V/V	Gain.
	IC	V	Initial condition: output voltage.

Integral: $V = K * \int Vin \, dt.$

If reset signal **R** is active, output is always zero. If rising or falling edge reset signal applied, output is set to zero and integration continues.

When calculating DC operating point, output is set to specified output voltage **IC**.

F – s-function

Symbol	Models		Signals
	Function Poly1 Poly2 Poly3 Poly4	Poly5 Roots Table File	

Model	Parameter	Units	Description
Function	f	V/V	Transfer function.
<p>Transfer function f defines transfer function in s domain. The following variables can represent frequency in the function:</p> <p>f – current AC frequency, Hz w – angular AC frequency, w = 2πf s or p – Laplace parameter, s = p = j*2πf</p> <p>Example:</p> <p>f = 1/(1+s) f = exp(-R1*C1**s)</p> <p>Only operators and functions that support complex numbers can be used in this function. If f is blank, it is assumed to be 1.</p> <p>At transient and DC operation point calculation for AC (if enabled), the component behaves as a buffer with infinite bandwidth, and gain equal to f(0).</p>			

Model	Parameter	Units	Description
Poly1 Poly2 Poly3 Poly4 Poly5	b0		Numerator polynomial coefficients 0.

	a0		Denominator polynomial coefficients 0.

	IC		Initial condition.

AC transfer function is a ratio of polynomials of Laplace parameter s :

$$f(s) = (b0 + b1*s + b2*s^2 + \dots) / (a0 + a1*s + a2*s^2 + \dots)$$

These models support transient as well.

Initial condition **IC** is a **csv** (comma separated values) string, where the first value is initial output voltage, and other values are internal model values (derivatives of input and output signal).

IC can be modified manually:

- Clear **IC** string and leave it blank to indicate that initial conditions are not specified.
- Enter just one value – initial output voltage.
- Modify the first value – initial output voltage.

If **Save IC** command was performed, then modifying **IC** parameter manually is not recommended.

At DC operation point calculation, **f(0)** is used.

Model	Parameter	Units	Description
Roots	K		Gain.
	Roots		Roots (zeroes and poles).
	IC	V	Initial condition.

AC transfer function is defines by zeroes and poles:

$$f(s) = K * (s-z1)*(s-z2).../(s-p1)/(s-p2)...$$

where **K** is gain, z1...zn are zeroes, p1...pN are poles.

Roots are defined by **Roots** parameter in the **csv** (comma-separated values) format, as follows:

Nz,Rez1,Imz1,...,Np,Rep1,Imp1,...

where:

Nz - number of zeroes

Rezi – real part of zi

Imzi – imaginary part of zi

Np - number of poles,

Repi – real part of pi

Impi – imaginary part of pi

There could be any number of zeroes and poles, however the resulting numerator and denominator polynomials order should not exceed 5. See *Working with Roots model* chapter for details on entering/editing roots.

The model supports transient as well.

Initial condition **IC** is a **csv** (comma separated values) string, where the first value is initial output voltage, and other values are internal model values (derivatives of input and output signal).

IC can be modified manually:

- Clear **IC** string and leave it blank to indicate that initial conditions are not specified.
- Enter just one value – initial output voltage.
- Modify the first value – initial output voltage.

If **Save IC** command was performed, then modifying **IC** parameter manually is not recommended.

At DC operation point calculation, **f(0)** is used.

Model	Parameter	Units	Description
Table	Table		Comma-separated string.
	K		Gain used for transient simulation.

Look-up table. Transfer function is defined by **Table** parameter in the **csv** (comma separated values) format, as follows:

f1,mag1,phase1,f2,mag2,phase2,...,fN,magN,phaseN

where mag and phase are transfer function magnitude and phase at specified frequency. Output value between specified frequencies is interpolated using linear or logarithmic interpolation, depending on simulation scale. Below first and above last specified frequency, simulation result is not provided. Frequencies should be given in ascending order.

At transient simulation, constant gain **K** is used.

See *Working with Table model* chapter of NL5 Manual for more details.

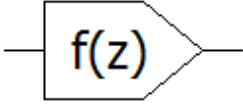
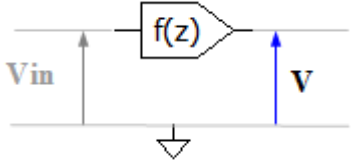
Model	Parameter	Units	Description
File	File		File name.
	K		Gain used for transient simulation.

Transfer function is defined in the text file in the following format:

f1,mag1,phase1
f2,mag2,phase2

At transient simulation, constant gain **K** is used.

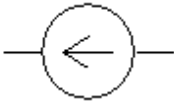
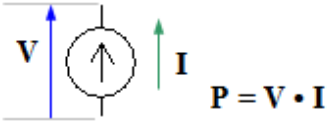
F – z-function

Symbol	Models	Signals
	Function Poly1 Poly2	Poly3 Poly4 Poly5
		

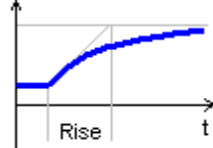
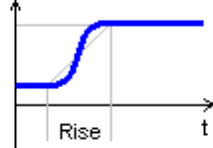
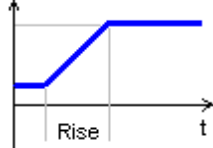
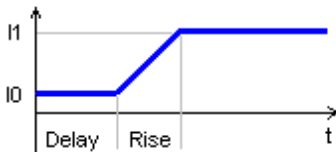
Model	Parameter	Units	Description
Function	f	V/V	Transfer function.
<p>Transfer function f defines transfer function in z domain. The following variables can represent frequency in the function:</p> <p>f – current AC frequency, Hz w – angular AC frequency, $w = 2\pi f$. s or p – Laplace parameter, $s = p = j*2\pi f$. z – z-parameter.</p> <p>Definition of z-parameter is located in <i>AC settings/Advanced settings/AC</i> window. Please note that if T parameter is used in z-parameter formula - for example, $\exp(s*T)$ - it should be defined as a schematic variable.</p> <p>If f is blank, it is assumed to be 1.</p> <p>At transient and DC operation point calculation for AC (if enabled), the component behaves as a buffer with infinite bandwidth, and gain equal to the transfer function value at zero frequency.</p>			

Model	Parameter	Units	Description
Poly1	b0		Numerator polynomial coefficients 0.
Poly2
Poly3	a0		Denominator polynomial coefficients 0.
Poly4
Poly5			
<p>AC transfer function is a ratio of polynomials of z-parameter:</p> $f(z) = (b0 + b1*z^{-1} + b2*z^{-2} + \dots) / (a0 + a1*z^{-1} + a2*z^{-2} + \dots)$ <p>At transient, and DC operation point calculation for AC (if enabled), the component behaves as a buffer with infinite bandwidth, and gain equal to the transfer function value at zero frequency.</p>			

I – Current source

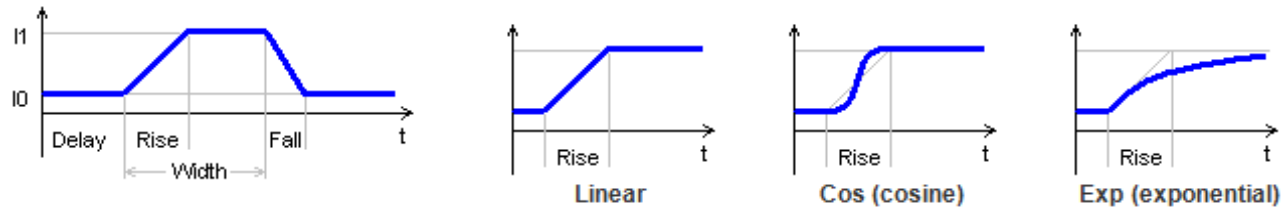
Symbol	Models		Signals
	I Step Single Pulse Clock Sin	Sweep Function List File SubCir	

Model	Parameter	Units	Description
I	I	A	Current.
Constant current = I.			

Model	Parameter	Units	Description
Step	I1	A	Step On current.
	I0	A	Step Off current.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Step rise length.
	Delay	s	Delay before step starts.
Step starts after Delay time. If Rise is non-zero, 3 Slope types are available.			
<div><div>LinearCos (cosine)Exp (exponential)</div></div>			

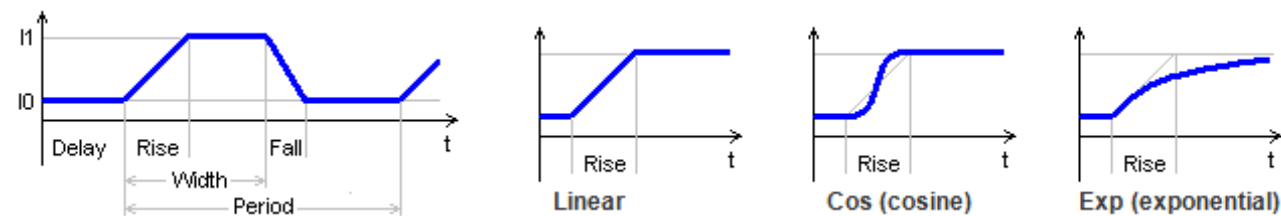
Model	Parameter	Units	Description
Single	I1	A	Pulse On current.
	I0	A	Pulse Off current.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before pulse starts.

Single pulse starts after **Delay** time. **Rise** time is included into **Width**, **Fall** time is **not** included into **Width**. **Slope** type applies both to pulse rise and fall (if non-zero).



Model	Parameter	Units	Description
Pulse	I1	A	Pulse On current.
	I0	A	Pulse Off current.
	Period	s	Period.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before first pulse starts.

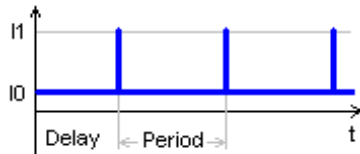
Pulses start after **Delay** time. **Rise** time is included into **Width**, **Fall** time is **not** included into **Width**. **Slope** type applies both to pulse rise and fall (if non-zero).



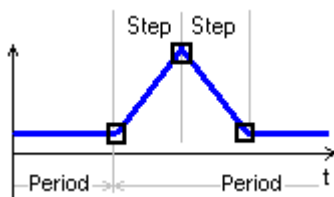
Model	Parameter	Units	Description
Clock	I1	A	Pulse On current.
	I0	A	Pulse Off current.
	Period	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.

Periodic pulses with width of one simulation step. Pulses start after **Delay** time.

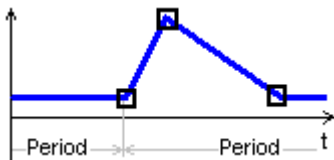
Clock model is recommended to produce a constant frequency clock signal. Unlike **Pulse** model, it won't force unnecessary step reduction at the end of the pulse, which may help to accelerate simulation.



If **Step** parameter is not zero **and** is less than current schematic simulation step, the step is adjusted to provide rise and fall of clock pulse to be equal to **Step**:

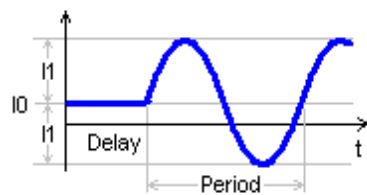


Otherwise, the clock pulse is created using current schematic simulation step, which depends on many factors, and cannot be easily predicted:

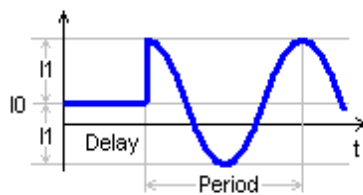


Model	Parameter	Units	Description
Sin	I1	A	Current amplitude.
	I0	A	Current baseline.
	Period	s	Period.
	Phase	deg	Phase.
	Decay	1/s	Decay constant
	Delay	s	Delay before sine signal starts.

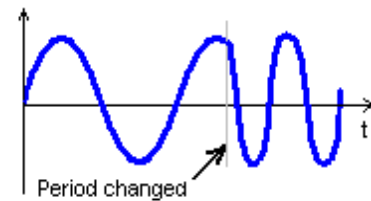
Sine signal starts after **Delay** time. **Phase** is sine phase in degrees at the moment when signal starts. If transient is paused, sine period changed, and then transient is continued, the phase of the signal remains continuous, providing smooth sine signal of variable frequency. If **Decay** is not zero, the sine signal is exponentially dumped with time constant = $1/\text{Decay}$.



Phase = 0



Phase = 90



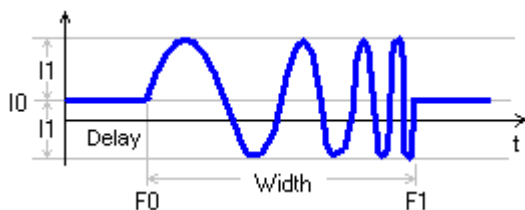
Model	Parameter	Units	Description
Sweep	I1	V	Current amplitude.
	I0	V	Current baseline.
	Width	s	Width of the signal.
	F0	Hz	Start frequency.
	F1	Hz	End frequency.
	Type		Signal type: Linear/Exp.
	Delay	s	Delay before signal starts.

Sinusoidal signal with variable frequency starts after **Delay** time. Signal frequency changes during **Width** interval from **F0** to **F1** linearly or exponentially, depending on specified **Type**.

If **F0 = F1**, then one period of frequency $1/\text{Width}$ will be generated.

If lowest frequency is set to zero and **Type** = Exp, then lowest frequency $0.01/\text{Width}$ will be used.

If needed, the highest frequency will be increased to provide integer number of signal periods, so that signal phase at the beginning and at the end of **Width** interval is exactly zero.



Model	Parameter	Units	Description
Function	f	A	Function

Arbitrary function **f** defines current as a function of the following variables:

- t** - current time
- V(name)** - voltage on the component **name**
- I(name)** - current through the component **name**
- P(name)** – power on the component **name**
- S(name)** – state of the component **name**

where **name** is the name of any component in the schematic. If **f** is blank, current is zero.

Example:

```
f = sin(t) * (1+cos(t*.01))
f = V(R1) * I(R1)
```

Please note that **V**, **I**, **P**, and **S** variables are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
List	List		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Piecewise linear signal is defined by **List** parameter in the **csv** (comma-separated values) format, as follows:

$$t_0, I_0, t_1, I_1, \dots, t_n, I_n$$

where all **t** and **I** can be numerical values or expressions.

If $t < t_0$, signal is I_0 .

If $t_0 < t < t_1$, signal value is linearly interpolated between I_0 and I_1 , etc.

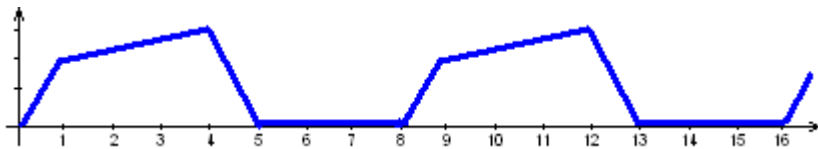
If $t > t_n$, and **Cycle** parameter is set to **No**, the signal value is I_n . Otherwise the signal defined in $t_0 \dots t_n$ interval is repeated continuously.

Signal start is delayed by **Delay** time.

Example:

```
List = 0,0,1,2,4,3,5,0,8,0
```

If **Cycle = Yes**, **Delay = 0**, the following current will be generated:



See *Working with List source* chapter for more details.

Model	Parameter	Units	Description
File	File		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Piecewise linear signal is defined in the text file.If **File** parameter does not have a full path, NL5 will search for the file in the directory where schematic file is located, then in the Library directories, and then in the Library directories relative to schematic file directory (see NL5 Manual, *Schematic Properties*).

Signal is defined in the **csv** (comma-separated values) format, as follows:

```
<if first line does not start with a number, it is ignored>
t0,I0
t1,I1
.....
tn,In
```

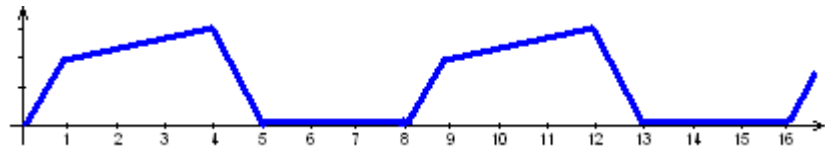
where all t and I can be numerical values or expressions.
If $t < t_0$, signal is v_0 .
If $t_0 < t < t_1$, signal value is linearly interpolated between I_0 and I_1 , etc.
If $t > t_n$, and **Cycle** parameter is set to **No**, the signal value is I_n . Otherwise the signal defined in $t_0 \dots t_n$ interval is repeated continuously.

Signal start is delayed by **Delay** time.

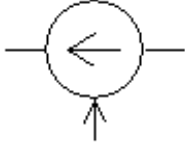
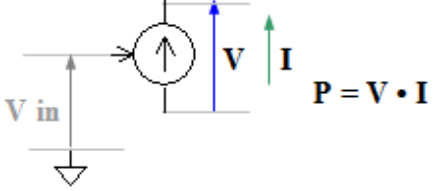
Example:

```
0,0
1,2
4,3
5,0
8,0
```

If **Cycle = Yes**, **Delay = 0**, the following current will be generated:



I – Logic controlled current source

Symbol	Models	Signals
	<div>I</div> <div>One-shot</div> <div>Step</div> <div>Single</div> <div>Pulse</div> <div>Clock</div> <div>Sin</div> <div>Sweep</div> <div>Function</div> <div>List</div> <div>File</div> <div>SubCir</div>	

Model	Parameter	Units	Description
I	I	A	Current.
Constant current = I.			

Model	Parameter	Units	Description
One-shot	I1	A	Pulse On current.
	I0	A	Pulse Off current.
	Width	s	Pulse width.
<p>One-shot pulse generator. When increasing input voltage V_{in} crosses logical threshold, current pulse of Width duration is generated. I0 is pulse Off level, I1 is pulse On level.</p> <p>If increasing V_{in} crosses logical threshold value while pulse is being generated, the pulse is restarted.</p>			

Model	Parameter	Units	Description
Step	I1	A	Step On current.
	I0	A	Step Off current.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Step rise length.
	Delay	s	Delay before step starts.
<p>When control signal V_{in} is below logical threshold, output is in Off state. When increasing control signal V_{in} crosses logical threshold, a signal similar to Step model of Current source component is generated. When decreasing control signal V_{in} drops below logical threshold, output goes to Off state immediately.</p>			

Model	Parameter	Units	Description
Single	I1	A	Pulse On current.
	I0	A	Pulse Off current.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before pulse starts.

When control signal *Vin* is below logical threshold, output is in Off state. When increasing control signal *Vin* crosses logical threshold, a signal similar to **Single** model of **Current source** component is generated. When decreasing control signal *Vin* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Pulse	I1	A	Pulse On current.
	I0	A	Pulse Off current.
	Period	s	Period.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before first pulse starts.

When control signal *Vin* is below logical threshold, output is in Off state. When increasing control signal *Vin* crosses logical threshold, a signal similar to **Pulse** model of **Current source** component is generated. When decreasing control signal *Vin* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Clock	I1	A	Pulse On current.
	I0	A	Pulse Off current.
	Period	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.

When control signal *Vin* is below logical threshold, output is in Off state. When increasing control signal *Vin* crosses logical threshold, a signal similar to **Clock** model of **Current source** component is generated. When decreasing control signal *Vin* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Sin	I1	A	Current amplitude.
	I0	A	Current baseline.
	Period	s	Period.
	Phase	deg	Phase.
	Decay	1/s	Decay constant
	Delay	s	Delay before sine signal starts.

When control signal V_{in} is below logical threshold, output current is **I0**. When increasing control signal V_{in} crosses logical threshold, a signal similar to **Sin** model of **Current source** component is generated. When decreasing control signal V_{in} drops below logical threshold, output goes to **I0** immediately.

Model	Parameter	Units	Description
Sweep	I1	V	Current amplitude.
	I0	V	Current baseline.
	Width	s	Width of the signal.
	F0	Hz	Start frequency.
	F1	Hz	End frequency.
	Type		Signal type: Linear/Exp.
	Delay	s	Delay before signal starts.

When control signal V_{in} is below logical threshold, output current is **I0**. When increasing control signal V_{in} crosses logical threshold, a signal similar to **Sweep** model of **Current source** component is generated. When decreasing control signal V_{in} drops below logical threshold, output goes to **I0** immediately.

Model	Parameter	Units	Description
Function	f	A	Function

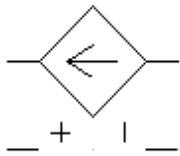
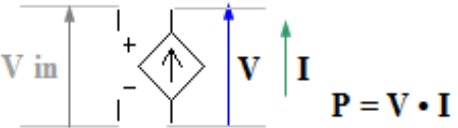
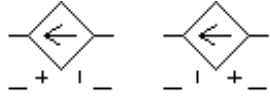
When control signal V_{in} is below logical threshold, output is zero. When increasing control signal V_{in} crosses logical threshold, a signal similar to **Function** model of **Current source** component is generated. If the function is using current time variable t , this moment will be considered as $t=0$. When decreasing control signal V_{in} drops below logical threshold, output goes to zero immediately.

Model	Parameter	Units	Description
List	List		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

When control signal V_{in} is below logical threshold, output is equal to **I0** value of **List** signal. When increasing control signal V_{in} crosses logical threshold, a signal similar to **List** model of **Current source** component is generated. This moment is also considered as $t=0$ for the **List** signal. When decreasing control signal V_{in} drops below logical threshold, output goes to **I0** immediately.

Model	Parameter	Units	Description
File	File		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.
<p>When control signal V_{in} is below logical threshold, output is equal to I0 value specified in the File. When increasing control signal V_{in} crosses logical threshold, a signal similar to File model of Current source component is generated. This moment is also considered as $t=0$ for the File signal. When decreasing control signal V_{in} drops below logical threshold, output goes to I0 immediately.</p>			

I – Voltage controlled current source

Symbol	Models	Signals
	Linear I Function PWL VCO One-shot PWM SubCir	
Views		

Model	Parameter	Units	Description
Linear	K	A/V	Gain
Linear voltage controlled current source: $I = K * V_{in}$.			

Model	Parameter	Units	Description
I	I	A	Current.
Constant current = I.			

Model	Parameter	Units	Description
Function	f	A	Output as function of the input.
	IC	A	Initial condition: output current.

Arbitrary function **f** defines output current as a function of the following variables:

- x** – input voltage *V_{in}*
- t** - current time
- V(name)** - voltage on the component **name**
- I(name)** - current through the component **name**
- P(name)** – power on the component **name**
- S(name)** – state of the component **name**

where **name** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

```
f = x*x
f = x * sin(t)
f = P(r1) + P(r2)
```

When calculating DC operating point, and in AC analysis, output is set to specified output current **IC**. Please note that variable **x** (input voltage *V_{in}*) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, K(<i>V_{in}</i>)

Piecewise linear voltage controlled current source. **pwl** string defines gain as a function of input voltage K(*V_{in}*). The transfer function of the source I(*V_{in}*) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that K(*V_{in}*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

Model	Parameter	Units	Description
VCO	I1	A	Current amplitude.
	I0	A	Current baseline or Off level.
	dFdV	Hz/V	Gain.
	Type		Signal type: Sin/Square/Triangle/Sawtooth.
	Phase	deg	Phase.

Voltage controlled oscillator. Output current is a signal with frequency equal to:

$$f(\text{Hz}) = \text{dFdV} * V_{in}.$$

For **Sine** signal, **I0** is baseline, and **I1** is amplitude. For **Square**, **Triangle**, and **Sawtooth** signals, **I0** is Off level, **I1** is On level. **Phase** is additional phase of the signal, in degrees.

Model	Parameter	Units	Description
One-shot	I1	A	Pulse On current.
	I0	A	Pulse Off current.
	Width	s	Pulse width.
	Threshold	V	Voltage threshold.

One-shot pulse generator. When increasing input voltage V_{in} crosses **Threshold** value, current pulse of **Width** duration is generated. **I0** is pulse Off level, **I1** is pulse On level.

If increasing V_{in} crosses **Threshold** value while pulse is being generated, the pulse is restarted.

Model	Parameter	Units	Description
PWM	I1	A	Pulse On current.
	I0	A	Pulse Off current.
	F	Hz	Frequency.
	Vmax	V	Input voltage corresponding to 100% duty.
	Phase	deg	Phase.

Voltage controlled Pulse-Width Modulator. Output current is a pulse signal of frequency **F** shifted by **Phase**. Input voltage V_{in} is sampled at the beginning of each cycle of the signal, and width of the output pulse during this cycle is calculated according to the equation:

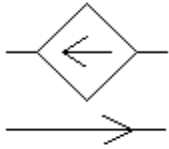
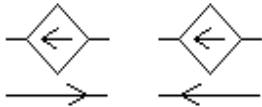
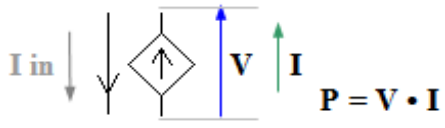
$$\text{width} = 1/F * (V_{in} / V_{\text{max}})$$

or

$$\text{duty} = 100\% * (V_{in} / V_{\text{max}});$$

If the width is equal or less than zero, a short On pulse with the width equal to the minimum calculation step at that moment will be generated. If the width is equal or greater than period of frequency **F**, a short Off pulse at the end of the period will be generated. As a result, the frequency of the output signal is always **F**.

I – Current controlled current source

Symbol	Models	Signals
	Linear I Function PWL	CCO One-shot PWM SubCir
Views 		

Model	Parameter	Units	Description
Linear	K	A/A	Gain
Linear current controlled current source: $I = K * I_{in}$.			

Model	Parameter	Units	Description
I	I	A	Current.
Constant current = I.			

Model	Parameter	Units	Description
Function	f	A	Output as function of the input.
	IC	A	Initial condition: output current.
<p>Arbitrary function f defines output current as a function of the following variables:</p> <ul style="list-style-type: none"> x – input current I_{in} t - current time V(name) - voltage on the component name I(name) - current through the component name P(name) – power on the component name S(name) – state of the component name <p>where name is the name of the component in the schematic. If f is blank, output is zero.</p> <p>Example:</p> <pre>f = x*x f = x * sin(t) f = P(r1) + P(r2)</pre> <p>When calculating DC operating point, and in AC analysis, output is set to specified output current IC. Please note that variable x (input current I_{in}) and variables V, I, P, and S are taken at previous calculation step. This may affect stability of the schematic with closed loop.</p>			

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, K(<i>I_{in}</i>)

Piecewise linear current controlled current source. **pwl** string defines gain as a function of input current K(*I_{in}*). The transfer function of the source *I*(*I_{in}*) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that K(*I_{in}*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

Model	Parameter	Units	Description
CCO	I1	A	Current amplitude.
	I0	A	Current baseline or Off level.
	dFdl	Hz/A	Gain.
	Type		Signal type: Sin/Square/Triangle/Sawtooth.
	Phase	deg	Phase.

Current controlled oscillator. Output current is a signal with frequency equal to:

$$f(\text{Hz}) = \text{dFdl} * I_{in}.$$

For **Sine** signal, **I0** is baseline, and **I1** is amplitude. For **Square**, **Triangle**, and **Sawtooth** signals, **I0** is Off level, **I1** is On level. **Phase** is additional phase of the signal, in degrees.


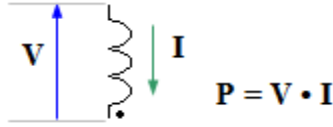
Model	Parameter	Units	Description
One-shot	I1	A	Pulse On current.
	I0	A	Pulse Off current.
	Width	s	Pulse width.
	Threshold	A	Current threshold.

One-shot pulse generator. When increasing input current *I_{in}* crosses **Threshold** value, current pulse of **Width** duration is generated. **I0** is pulse Off level, **I1** is pulse On level.

If increasing *I_{in}* crosses **Threshold** value while pulse is being generated, the pulse is restarted.

Model	Parameter	Units	Description
PWM	I1	A	Pulse On current.
	I0	A	Pulse Off current.
	F	Hz	Frequency.
	I_{max}	A	Input current corresponding to 100% duty.
	Phase	deg	Phase.
<p>Current controlled Pulse-Width Modulator. Output current is a pulse signal of frequency F shifted by Phase. Input current <i>I_{in}</i> is sampled at the beginning of each cycle of the signal, and width of the output pulse during this cycle is calculated according to the equation:</p> $\text{width} = 1/F * (I_{in} / I_{max})$ <p>or</p> $\text{duty} = 100\% * (I_{in} / I_{max});$ <p>If the width is equal or less than zero, a short On pulse with the width equal to the minimum calculation step at that moment will be generated. If the width is equal or greater than period of frequency F, a short Off pulse at the end of the period will be generated. As a result, the frequency of the output signal is always F.</p>			

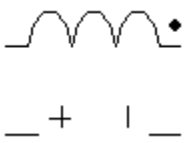
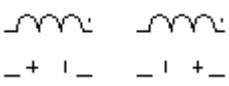
L – Inductor

Symbol	Models	Signals
	L PWL SubCir	

Model	Parameter	Units	Description
L	L	H	Inductance
	IC	A	Initial condition: current. Leave blank if IC not defined.
<p>Linear inductor, $V = L \cdot di/dt$.</p> <p>When calculating DC operating point, if IC is defined, inductor is replaced with current source equal to IC. Otherwise, inductor is temporarily replaced by short circuit, DC operating point is calculated, and then the current through the short circuit is assigned to the inductor as its initial current.</p>			

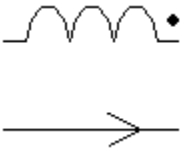
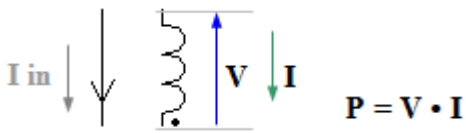
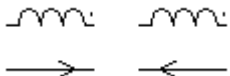
Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, L(I)
	IC	A	Initial condition: current. Leave blank if IC not defined.
<p>Piecewise constant inductor: pwl string defines inductance as a function of current through the inductor L(I). H(I) is piecewise linear function, and it always goes through the origin (0,0). See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that L(I) is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p> <p>When calculating DC operating point, if IC is defined, inductor is replaced with current source equal to IC. Otherwise, inductor is temporarily replaced by short circuit, DC operating point is calculated, and then the current through the short circuit is assigned to the inductor as its initial current.</p>			

L – Voltage controlled inductor

Symbol	Models	Signals
	PWL	
Views		

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, L(Vin)
	IC	A	Initial condition: current. Leave blank if IC not defined.
<p>Piecewise constant voltage controlled inductor. pwl string defines inductance as a function of control voltage L(Vin). At any moment:</p> $V = L(Vin) * di/dt.$ <p>See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that L(Vin) is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p> <p>When calculating DC operating point, if IC is defined, inductor is replaced with current source equal to IC. Otherwise, inductor is temporarily replaced by short circuit, DC operating point is calculated, and then the current through the short circuit is assigned to the inductor as its initial current.</p>			

L – Current controlled inductor

Symbol	Models	Signals
	PWL	
Views 		

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, L(lin)
	IC	A	Initial condition: current. Leave blank if IC not defined.

Piecewise constant current controlled inductor. **pwl** string defines inductance as a function of control current C(lin). At any moment:

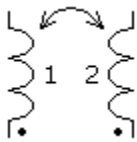
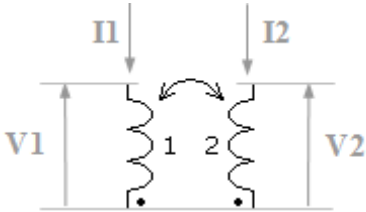
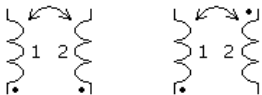
$$V = L(lin) * dl/dt.$$

See *Working with PWL model* chapter for details.

Please note that L(lin) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.


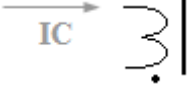
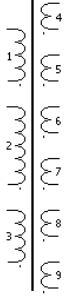
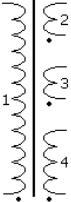


When calculating DC operating point, if **IC** is defined, inductor is replaced with current source equal to **IC**. Otherwise, inductor is temporarily replaced by short circuit, DC operating point is calculated, and then the current through the short circuit is assigned to the inductor as its initial current.

L – Coupled inductors

Symbol	Models	Signals
	L	
Views		

Model	Parameter	Units	Description
L	L1	H	L1 inductance
	L2	H	L2 inductance
	K		Coupling coefficient (-1...1)
	IC1	A	L1 initial condition: current. Leave blank if IC1 not defined.
	IC2	A	L2 initial condition: current. Leave blank if IC2 not defined.
<p>Coupled linear inductors.</p> $V1 = L1 * dI1/dt + M * dI2/dt$ $V2 = M * dI1/dt + L2 * dI2/dt$ <p>Where $M = K * \text{sqrt}(L1 * L2)$ is mutual inductance.</p> <p>When calculating DC operating point, initial conditions IC1 and IC2 are independently applied to corresponding inductors L1 and L2, similar to how it is done for the component L (inductor).</p>			

L – Custom coupled inductors

Symbol	Models	Signals
	L SubCir	
<p>This is a customized component. A component can be edited in the Edit Component dialog box. See <i>Editing customized component</i> chapter for instructions on editing a component.</p> <p>This component may have:</p> <ul style="list-style-type: none">- height from 2 to 32,- up to 32 windings (total),- arbitrary length of a winding. <p>Examples of Custom coupled inductors component:</p> <div></div>		

Model	Parameter	Units	Description
L	L1	H	L1 inductance
	...	H	...
	LN	H	LN inductance
	K12		L1-L2 coupling coefficient (-1...1)

	K(N-1)N		L(N-1)-LN coupling coefficient (-1...1)
	IC1	A	L1 initial condition: current. Leave blank if IC1 not defined.
	...	A	...
	ICN	A	LN initial condition: current. Leave blank if ICN not defined.

Custom coupled inductors.

$$V1 = L1 * dI1/dt + M12 * dI2/dt + \dots + M1N * dIN/dt$$

$$V2 = M12 * dI1/dt + L2 * dI2/dt + \dots + M2N * dIN/dt$$

...

$$VN = M1N * dI1/dt + M2N * dI2/dt + \dots + LN * dIN/dt$$

Where $M_{ij} = K_{ij} * \sqrt{L_i * L_j}$ is mutual inductance, $M_{ij} = M_{ji}$.

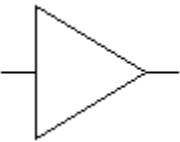
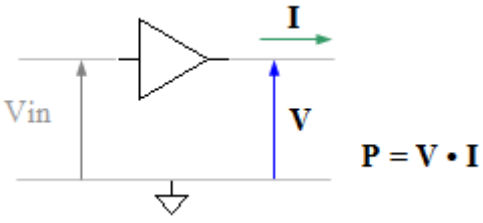
When calculating DC operating point, initial conditions **ICN** are independently applied to corresponding inductors **LN**, similar to how it is done for the component **L** (inductor).

If only one winding is defined, a component behaves exactly as a linear inductor **L**.

Please be aware that coupling coefficients **Kij** should be properly specified within allowable range (-1...1) in order to represent a "physically-realizable" system. If all coupling coefficients are equal to 1 (or -1), using **Winding** components **W** with one magnetizing inductor may give better performance and more stable solution.

If number of windings is more than 9, coupling coefficient parameters **Kij** are shown with underscore '_' between numbers **i** and **j**: for example, **K6_24**. For number of windings 9 and less, the old notation is used for backward compatibility.

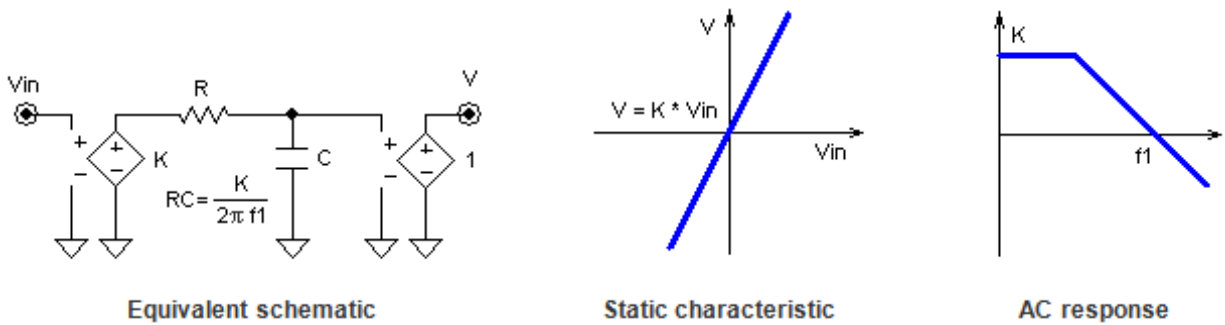
O – Amplifier

Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	

Model	Parameter	Units	Description
Linear	K	V/V	Gain
	f1	Hz	Unit gain frequency.
	IC	V	Initial condition: output voltage.

Linear amplifier. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**).

When calculating DC operating point, if **f1** is not infinity, and **IC** is defined, amplifier output is set to specified output voltage **IC**. If **IC** is blank, static characteristic is used.

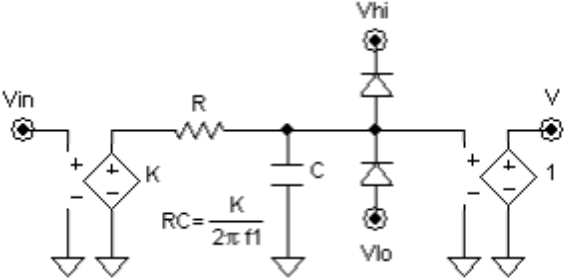


Model	Parameter	Units	Description
OpAmp	K	V/V	Gain
	f1	Hz	Unit gain frequency.
	Vhi	V	Max output voltage.
	Vlo	V	Min output voltage.
	IC	V	Initial condition: output voltage.

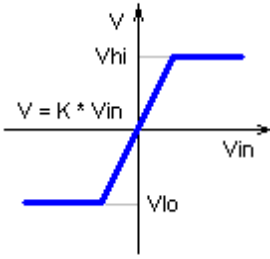
Linear amplifier with output limiter. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**). Output voltage is limiting between **Vlo** and **Vhi**.

When calculating DC operating point, amplifier output is set to specified output voltage **IC**.

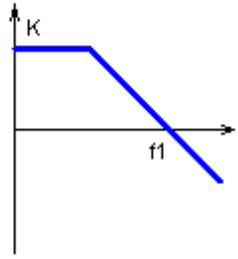
If both **K** and **f1** are set to infinity, the model may experience convergence problem: use **Comparator** model instead.



Equivalent schematic



Static characteristic



AC response

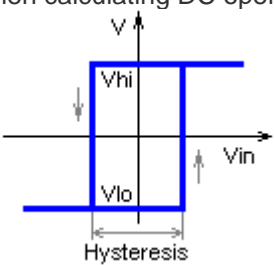
Model	Parameter	Units	Description
Comparator	Hysteresis	V	Hysteresis
	Vhi	V	Max output voltage.
	Vlo	V	Min output voltage.
	Delay	s	Output delay.
	IC		Initial condition: Low/High.

Comparator with hysteresis. Comparator output is set to **Vhi** or **Vlo** using following rules:

$V_{in} > \text{Hysteresis}/2 \dots V = V_{hi}$
 $V_{in} < -\text{Hysteresis}/2 \dots V = V_{lo}$
Otherwise: $V = \text{previous state}$

The output is delayed by **Delay** time. Input pulses shorter than **Delay** will not pass through and will not affect output.

When calculating DC operating point comparator, output is set to **Vlo** or to **Vhi**, according to selected **IC**.



Model	Parameter	Units	Description
Function	f	V	Output as function of the input.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

x – input voltage V_{in}

t - current time

V(name) - voltage on the component **name**

I(name) - current through the component **name**

P(name) – power on the component **name**

S(name) – state of the component **name**

where **name** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

$$f = x * x$$

$$f = x * \sin(t)$$

$$f = P(r1) + P(r2)$$

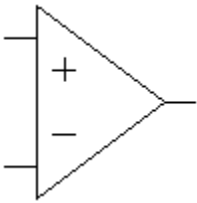
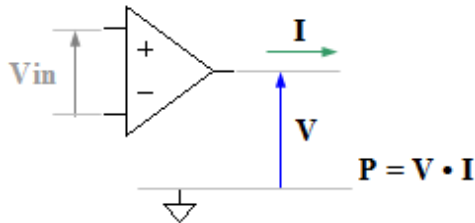
When calculating DC operating point, and in AC analysis, output is set to specified output voltage **IC**. Please note that variable **x** (input voltage V_{in}) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, $K(V_{in})$

Piecewise linear amplifier. **pwl** string defines gain as a function of input voltage $K(V_{in})$. Amplifier transfer function is $V(V_{in})$ is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that $K(V_{in})$ is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

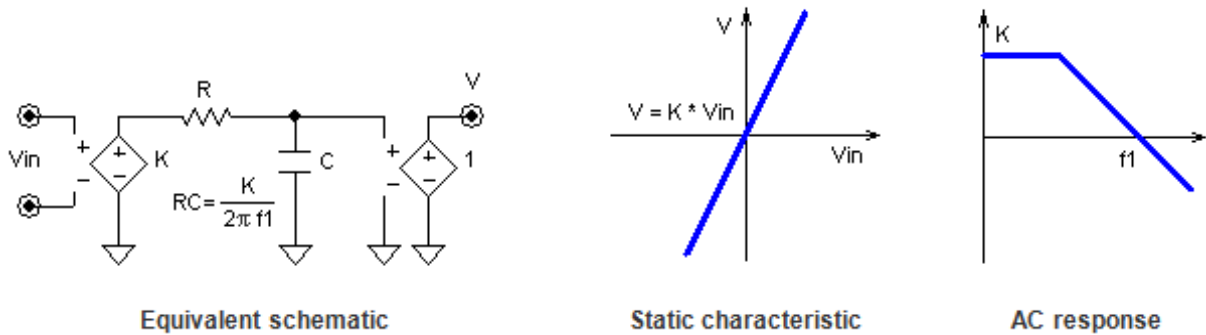
O – Differential amplifier

Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	

Model	Parameter	Units	Description
Linear	K	V/V	Gain
	f1	Hz	Unit gain frequency.
	IC	V	Initial condition: output voltage.

Linear differential amplifier. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**).

When calculating DC operating point, if **f1** is not infinity, and **IC** is defined, amplifier output is set to specified output voltage **IC**. If **IC** is blank, static characteristic is used.

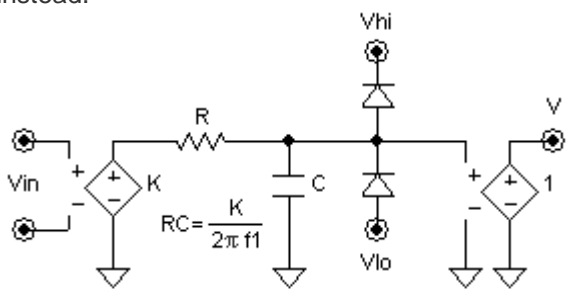


Model	Parameter	Units	Description
OpAmp	K	V/V	Gain
	f1	Hz	Unit gain frequency.
	Vhi	V	Max output voltage.
	Vlo	V	Min output voltage.
	IC	V	Initial condition: output voltage.

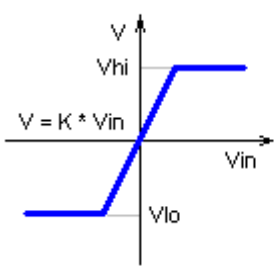
Linear amplifier with output limiter. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**). Output voltage is limiting between **Vlo** and **Vhi**.

When calculating DC operating point, amplifier output is set to specified output voltage **IC**.

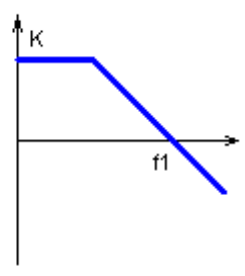
If both **K** and **f1** are set to infinity, the model may experience convergence problem: use **Comparator** model instead.



Equivalent schematic



Static characteristic



AC response

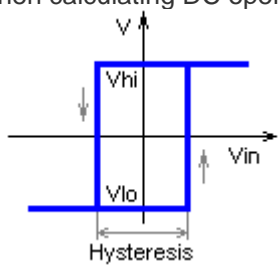
Model	Parameter	Units	Description
Comparator	Hysteresis	V	Hysteresis
	Vhi	V	Max output voltage.
	Vlo	V	Min output voltage.
	Delay	s	Output delay.
	IC		Initial condition: Low/High.

Comparator with hysteresis. Comparator output is set to **Vhi** or **Vlo** using following rules:

$V_{in} > \text{Hysteresis}/2 \dots V = V_{hi}$
 $V_{in} < -\text{Hysteresis}/2 \dots V = V_{lo}$
Otherwise $V = \text{previous state}$

The output is delayed by **Delay** time. Input pulses shorter than **Delay** will not pass through and will not affect output.

When calculating DC operating point comparator, output is set to **Vlo** or to **Vhi**, according to selected **IC**.



Model	Parameter	Units	Description
Function	f	V	Output as function of the input.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

x – input voltage V_{in}

t - current time

V(name) - voltage on the component **name**

I(name) - current through the component **name**

P(name) – power on the component **name**

S(name) – state of the component **name**

where **name** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

$$f = x * x$$

$$f = x * \sin(t)$$

$$f = P(r1) + P(r2)$$

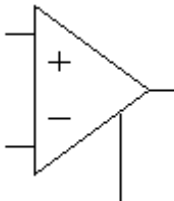
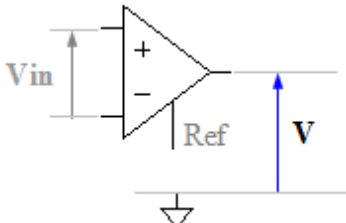
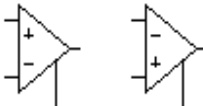
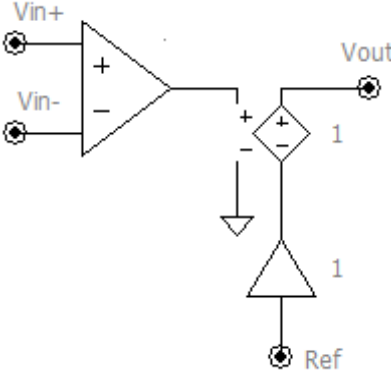
When calculating DC operating point, and in AC analysis, output is set to specified output voltage **IC**. Please note that variable **x** (input voltage V_{in}) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, $K(V_{in})$

Piecewise linear amplifier. **pwl** string defines gain as a function of input voltage $K(V_{in})$. Amplifier transfer function is $V(V_{in})$ is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that $K(V_{in})$ is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

O – Differential amplifier with reference

Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	
Views 		
<p>Equivalentent schematic :</p> 		
<p>All models are similar to the models of Differential amplifier component. Model parameters apply to the differential amplifier shown on the equivalent schematic. For example, Vhi and Vlo parameters of the OpAmp model will limit output of the differential amplifier, rather than voltage at the component output <i>Vout</i>.</p>		

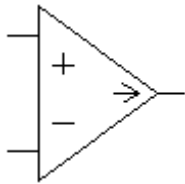
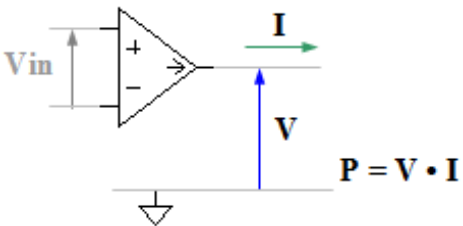
O – Fully differential amplifier

Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	
Views		

Equivalent schematic:

All models are similar to the models of **Differential amplifier** component. Model parameters apply to the differential amplifier shown on the equivalent schematic. For example, **Vhi** and **Vlo** parameters of the **OpAmp** model will limit output of the differential amplifier, rather than voltages at the component outputs *Vout+* and *Vout-*.

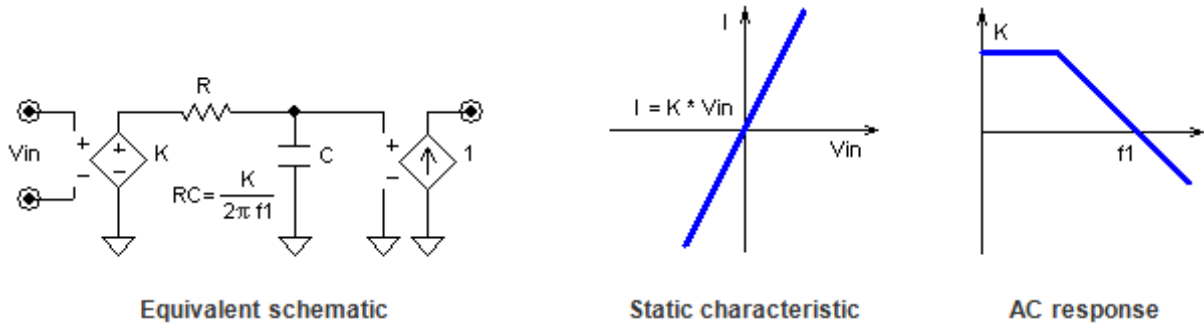
O – Differential amplifier with current output

Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	

Model	Parameter	Units	Description
Linear	K	A/V	Gain
	f1	Hz	Unit gain frequency.
	IC	A	Initial condition: output current.

Linear differential amplifier. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**).

When calculating DC operating point, if **f1** is not infinity, and **IC** is defined, amplifier output is set to specified output current **IC**. If **IC** is blank, static characteristic is used.

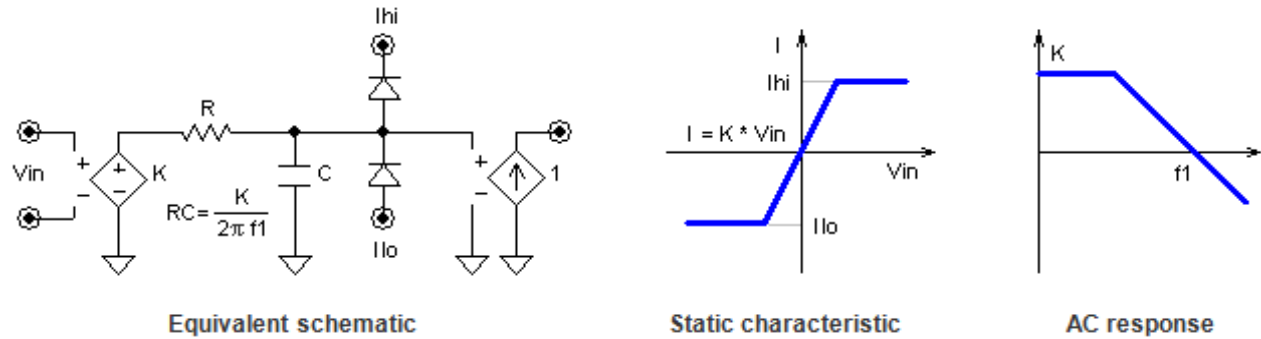


Model	Parameter	Units	Description
OpAmp	K	V/A	Gain
	f1	Hz	Unit gain frequency.
	Ihi	A	Max output current.
	Ilo	A	Min output current.
	IC	A	Initial condition: output current.

Linear amplifier with output limiter. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**). Output current is limiting between **Ilo** and **Ihi**.

When calculating DC operating point, amplifier output is set to specified output current **IC**.

If both **K** and **f1** are set to infinity, the model may experience convergence problem: use **Comparator** model instead.



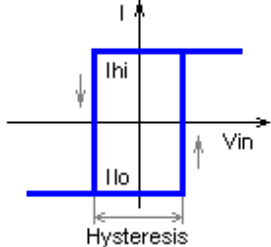
Model	Parameter	Units	Description
Comparator	Hysteresis	V	Hysteresis
	Ihi	A	Max output current.
	Ilo	A	Min output current.
	Delay	s	Output delay.
	IC		Initial condition: Low/High.

Comparator with hysteresis. Comparator output is set to **Ihi** or **Ilo** using following rules:

$V_{in} > \text{Hysteresis}/2 \dots I = I_{hi}$
 $V_{in} < -\text{Hysteresis}/2 \dots I = I_{lo}$
Otherwise $I = \text{previous state}$

The output is delayed by **Delay** time. Input pulses shorter than **Delay** will not pass through and will not affect output.

When calculating DC operating point comparator, output is set to **Ilo** or to **Ihi**, according to selected **IC**.



Model	Parameter	Units	Description
Function	f	A	Output as function of the input.
	IC	A	Initial condition: output current.

Arbitrary function **f** defines output current as a function of the following variables:

x – input voltage V_{in}

t - current time

V(name) - voltage on the component **name**

I(name) - current through the component **name**

P(name) – power on the component **name**

S(name) – state of the component **name**

where **name** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

$$f = x * x$$

$$f = x * \sin(t)$$

$$f = P(r1) + P(r2)$$

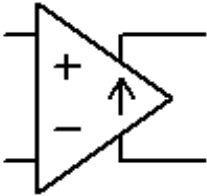
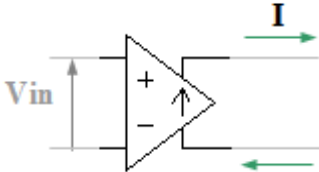
When calculating DC operating point, and in AC analysis, output is set to specified output current **IC**. Please note that variable **x** (input voltage V_{in}) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, $K(V_{in})$

Piecewise linear amplifier. **pwl** string defines gain as a function of input voltage $K(V_{in})$. Amplifier transfer function is $I(V_{in})$ is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that $K(V_{in})$ is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

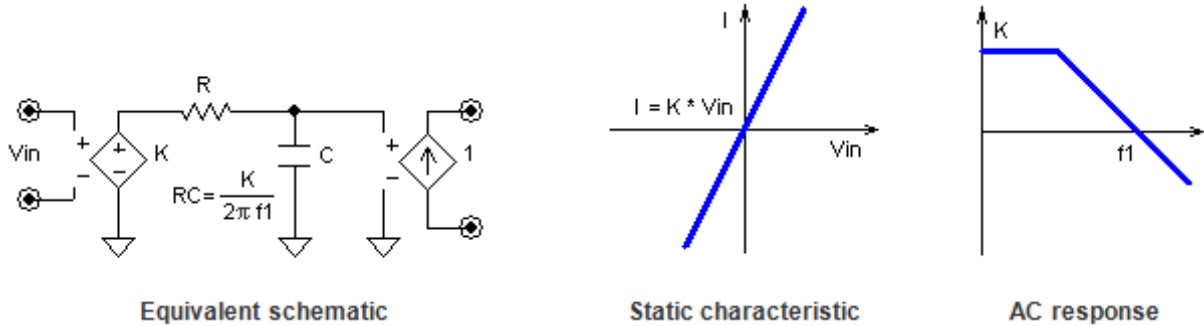
O – Differential amplifier with differential current output

Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	

Model	Parameter	Units	Description
Linear	K	A/V	Gain
	f1	Hz	Unit gain frequency.
	IC	A	Initial condition: output current.

Linear differential amplifier. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**).

When calculating DC operating point, if **f1** is not infinity, and **IC** is defined, amplifier output is set to specified output current **IC**. If **IC** is blank, static characteristic is used.

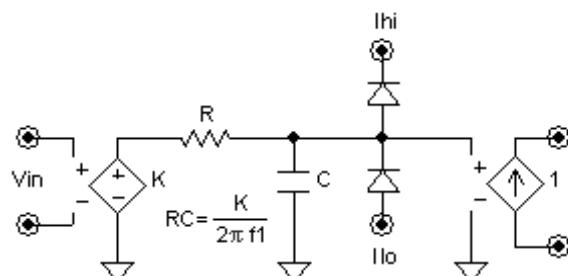


Model	Parameter	Units	Description
OpAmp	K	V/A	Gain
	f1	Hz	Unit gain frequency.
	Ihi	A	Max output current.
	Ilo	A	Min output current.
	IC	A	Initial condition: output current.

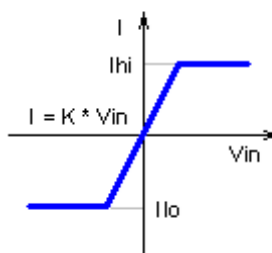
Linear amplifier with output limiter. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**). Output current is limiting between **Ilo** and **Ihi**.

When calculating DC operating point, amplifier output is set to specified output current **IC**.

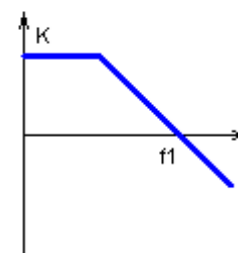
If both **K** and **f1** are set to infinity, the model may experience convergence problem: use **Comparator** model instead.



Equivalent schematic



Static characteristic



AC response

Model	Parameter	Units	Description
Comparator	Hysteresis	V	Hysteresis
	I _{hi}	A	Max output current.
	I _{lo}	A	Min output current.
	Delay	s	Output delay.
	IC		Initial condition: Low/High.

Comparator with hysteresis. Comparator output is set to **lhi** or **llo** using following rules:

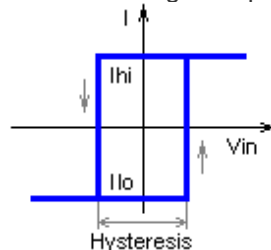
```

Vin > Hysteresis/2 . . . : I = Ihi
Vin < - Hysteresis/2 . . . : I = Ilo
Otherwise . . . . . : I = previous state

```

The output is delayed by **Delay** time. Input pulses shorter than **Delay** will not pass through and will not affect output.

When calculating DC operating point comparator, output is set to **llo** or to **lhi**, according to selected **IC**.



Model	Parameter	Units	Description
Function	f	A	Output as function of the input.
	IC	A	Initial condition: output current.

Arbitrary function **f** defines output current as a function of the following variables:

x – input voltage V_{in}

t - current time

V(name) - voltage on the component **name**

I(name) - current through the component **name**

P(name) – power on the component **name**

S(name) – state of the component **name**

where **name** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

$$f = x * x$$

$$f = x * \sin(t)$$

$$f = P(r1) + P(r2)$$

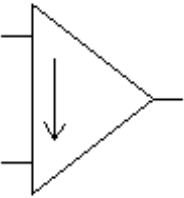
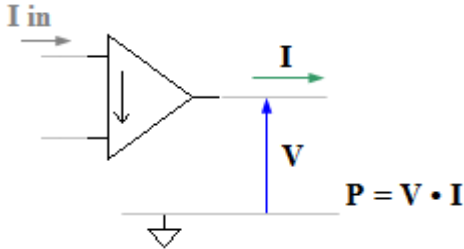
When calculating DC operating point, and in AC analysis, output is set to specified output current **IC**. Please note that variable **x** (input voltage V_{in}) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, $K(V_{in})$

Piecewise linear amplifier. **pwl** string defines gain as a function of input voltage $K(V_{in})$. Amplifier transfer function is $I(V_{in})$ is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that $K(V_{in})$ is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

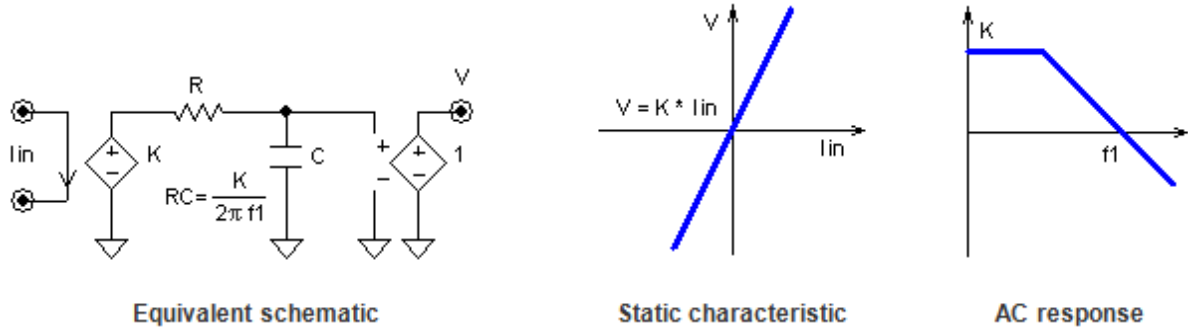
O – Current amplifier

Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	

Model	Parameter	Units	Description
Linear	K	V/A	Gain
	f1	Hz	Unit gain frequency.
	IC	V	Initial condition: output voltage.

Linear differential amplifier. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**).

When calculating DC operating point, if **f1** is not infinity, and **IC** is defined, amplifier output is set to specified output voltage **IC**. If **IC** is blank, static characteristic is used.

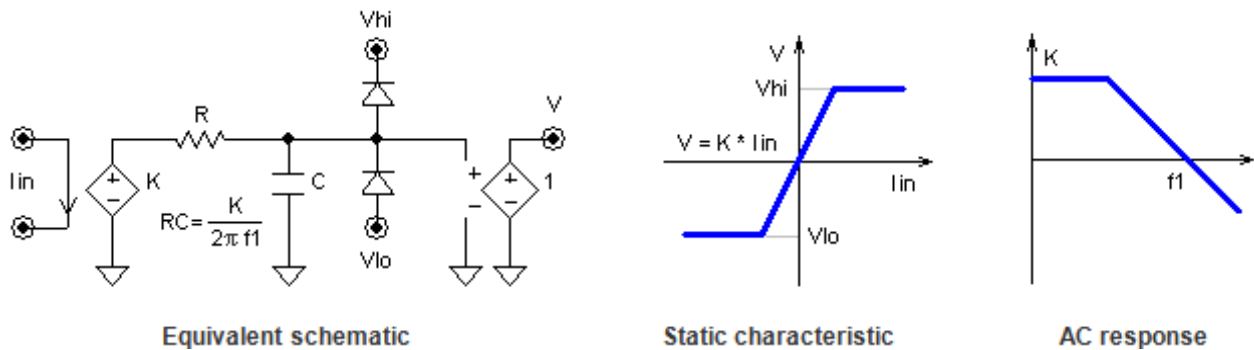


Model	Parameter	Units	Description
OpAmp	K	V/A	Gain
	f1	Hz	Unit gain frequency.
	Vhi	V	Max output voltage.
	Vlo	V	Min output voltage.
	IC	V	Initial condition: output voltage.

Linear amplifier with output limiter. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**). Output voltage is limiting between **Vlo** and **Vhi**.

When calculating DC operating point, amplifier output is set to specified output voltage **IC**.

If both **K** and **f1** are set to infinity, the model may experience convergence problem: use **Comparator** model instead.



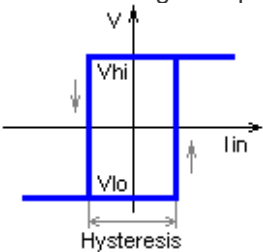
Model	Parameter	Units	Description
Comparator	Hysteresis	A	Hysteresis
	Vhi	V	Max output voltage.
	Vlo	V	Min output voltage.
	Delay	s	Output delay.
	IC		Initial condition: Low/High.

Comparator with hysteresis. Comparator output is set to **Vhi** or **Vlo** using following rules:

$I_{in} > \text{Hysteresis}/2 \dots V = V_{hi}$
 $I_{in} < -\text{Hysteresis}/2 \dots V = V_{lo}$
Otherwise: $V = \text{previous state}$

The output is delayed by **Delay** time. Input pulses shorter than **Delay** will not pass through and will not affect output.

When calculating DC operating point comparator, output is set to **Vlo** or to **Vhi**, according to selected **IC**.



Model	Parameter	Units	Description
Function	f	V	Output as function of the input.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

- x** – input current *I_{in}*
- t** - current time
- V(name)** - voltage on the component **name**
- I(name)** - current through the component **name**
- P(name)** – power on the component **name**
- S(name)** – state of the component **name**

where **name** is the name of the component in the schematic. If **f** is blank, output is zero.

Example:

- $f = x * x$
- $f = x * \sin(t)$
- $f = P(r1) + P(r2)$

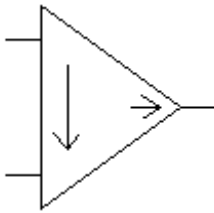
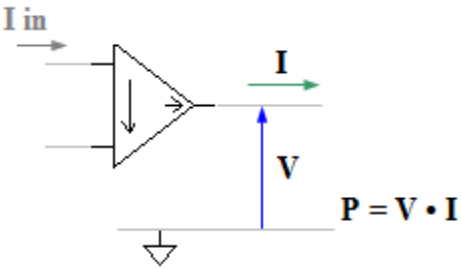
When calculating DC operating point, and in AC analysis, output is set to specified output voltage **IC**. Please note that variable **x** (input current *I_{in}*) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, K(<i>V_{in}</i>)

Piecewise linear amplifier. **pwl** string defines gain as a function of input current K(*I_{in}*). Amplifier transfer function is V(*I_{in}*) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that K(*I_{in}*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

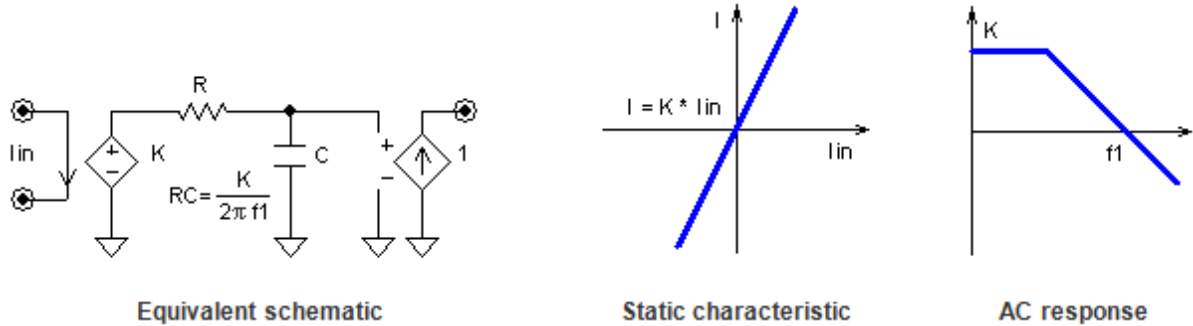
O – Current amplifier with current output

Symbol	Models	Signals
	Linear OpAmp Comparator Function PWL SubCir	

Model	Parameter	Units	Description
Linear	K	A/A	Gain
	f1	Hz	Unit gain frequency.
	IC	A	Initial condition: output current.

Linear differential amplifier. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**).

When calculating DC operating point, if **f1** is not infinity, and **IC** is defined, amplifier output is set to specified output current **IC**. If **IC** is blank, static characteristic is used.

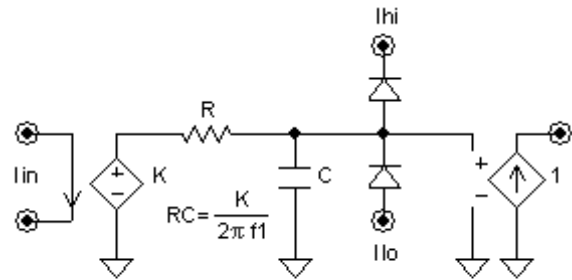


Model	Parameter	Units	Description
OpAmp	K	A/A	Gain
	f1	Hz	Unit gain frequency.
	Ihi	A	Max output current.
	Ilo	A	Min output current.
	IC	A	Initial condition: output current.

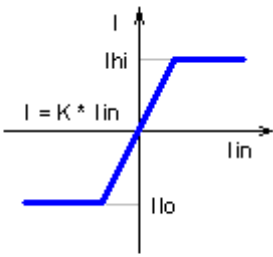
Linear amplifier with output limiter. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**). Output current is limiting between **Ilo** and **Ihi**.

When calculating DC operating point, amplifier output is set to specified output current **IC**.

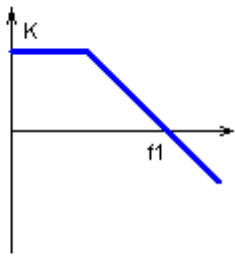
If both **K** and **f1** are set to infinity, the model may experience convergence problem: use **Comparator** model instead.



Equivalent schematic



Static characteristic



AC response

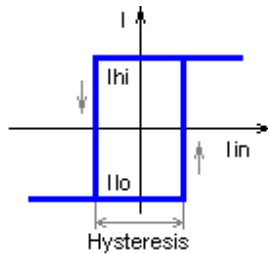
Model	Parameter	Units	Description
Comparator	Hysteresis	A	Hysteresis
	Ihi	A	Max output current.
	Ilo	A	Min output current.
	Delay	s	Output delay.
	IC		Initial condition: Low/High.

Comparator with hysteresis. Comparator output is set to **Ihi** or **Ilo** using following rules:

$I_{in} > \text{Hysteresis}/2 \dots : I = I_{hi}$
 $I_{in} < -\text{Hysteresis}/2 \dots : I = I_{lo}$
 Otherwise : $I = \text{previous state}$

The output is delayed by **Delay** time. Input pulses shorter than **Delay** will not pass through and will not affect output.

When calculating DC operating point comparator, output is set to **Ilo** or to **Ihi**, according to selected **IC**.



Model	Parameter	Units	Description
Function	f	A	Output as function of the input.
	IC	A	Initial condition: output current.

Arbitrary function **f** defines output current as a function of the following variables:

x – input current I_{in}
t - current time
V(name) - voltage on the component **name**
I(name) - current through the component **name**
P(name) – power on the component **name**
S(name) – state of the component **name**

where **name** is the name of the component in the schematic. If **f** is blank, output is zero.

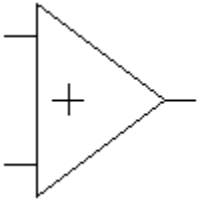
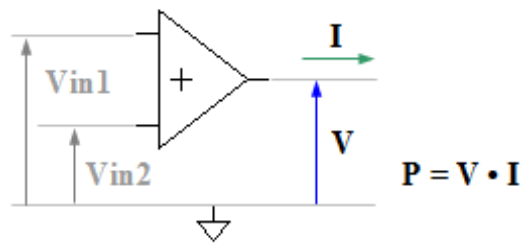
Example:

$f = x * x$
 $f = x * \sin(t)$
 $f = P(r1) + P(r2)$

When calculating DC operating point, and in AC analysis, output is set to specified output current **IC**. Please note that variable **x** (input current I_{in}) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, $K(I_{in})$
<p>Piecewise linear amplifier. pwl string defines gain as a function of input current $K(I_{in})$. Amplifier transfer function is $I(I_{in})$ is piecewise linear function, and it always goes through the origin (0,0). See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that $K(I_{in})$ is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p>			

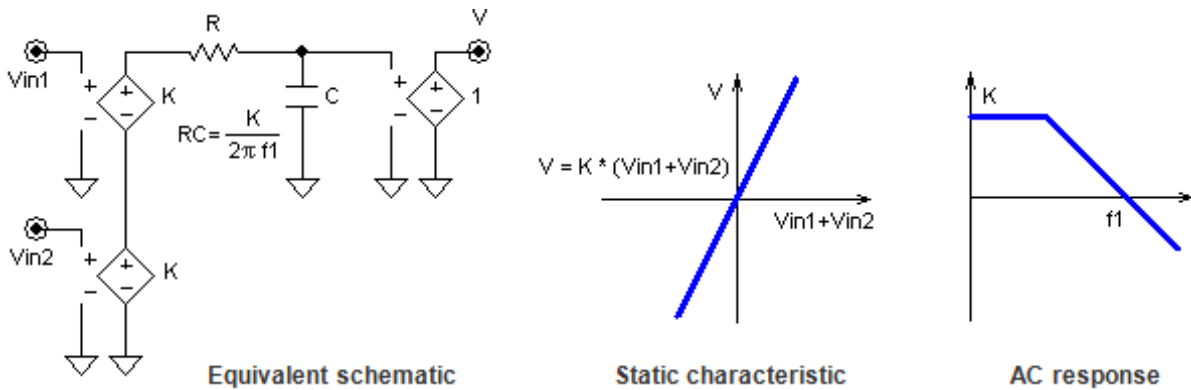
O – Summing amplifier

Symbol	Models	Signals
	Linear OpAmp Function PWL SubCir	

Model	Parameter	Units	Description
Linear	K	V/V	Gain
	f1	Hz	Unit gain frequency.
	IC	V	Initial condition: output voltage.

Linear summing amplifier. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**).

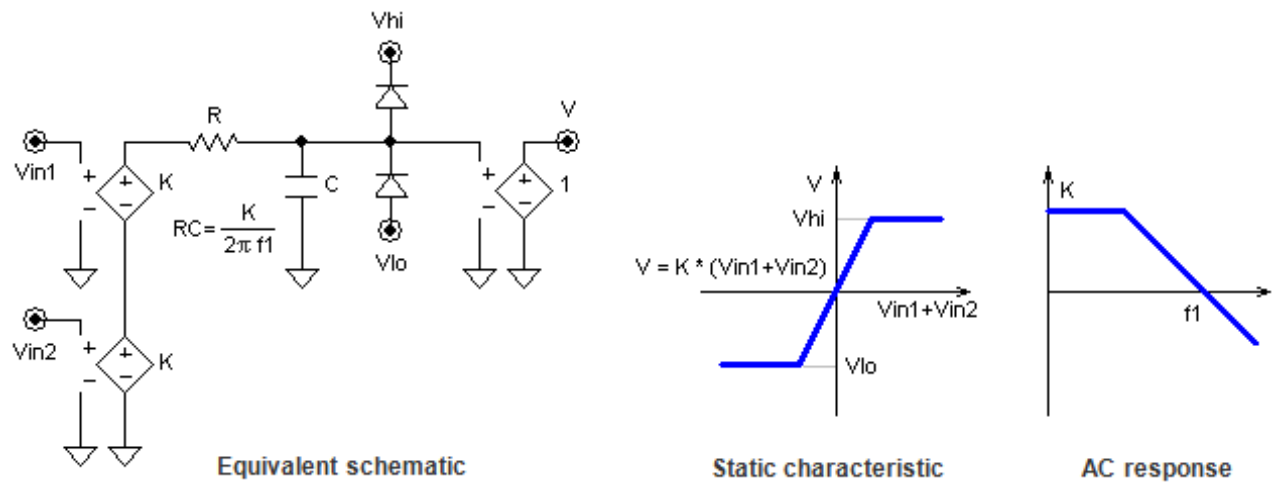
When calculating DC operating point, if **f1** is not infinity, and **IC** is defined, amplifier output is set to specified output voltage **IC**. If **IC** is blank, static characteristic is used.



Model	Parameter	Units	Description
OpAmp	K	V/V	Gain
	f1	Hz	Unit gain frequency.
	Vhi	V	Max output voltage.
	Vlo	V	Min output voltage.
	IC	V	Initial condition: output voltage.

Linear amplifier with output limiter. **K** is open loop gain. Frequency response consists of one pole, **f1** is unit gain frequency. **K** and **f1** can be set to infinity (**inf**). Output voltage is limiting between **Vlo** and **Vhi**.

When calculating DC operating point, amplifier output is set to specified output voltage **IC**.



Model	Parameter	Units	Description
Function	f	V	Output as function of the input.
	IC	V	Initial condition: output voltage.

Arbitrary function **f** defines output voltage as a function of the following variables:

- x** – voltage $V_{in1}+V_{in2}$
- t** - current time
- V(name)** - voltage on the component **name**
- I(name)** - current through the component **name**
- P(name)** – power on the component **name**
- S(name)** – state of the component **name**

where **name** is the name of the component in the schematic. If **f** is blank, output is zero.

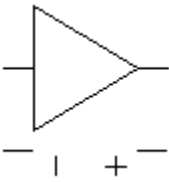
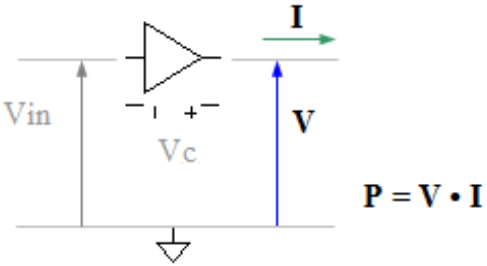
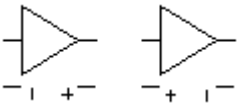
Example:

- $f = x*x$
- $f = x * \sin(t)$
- $f = P(r1) + P(r2)$

When calculating DC operating point, and in AC analysis, output is set to specified output voltage **IC**. Please note that variable **x** (input voltage $V_{in1}+V_{in2}$) and variables **V**, **I**, **P**, and **S** are taken at previous calculation step. This may affect stability of the schematic with closed loop.

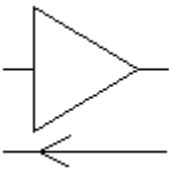
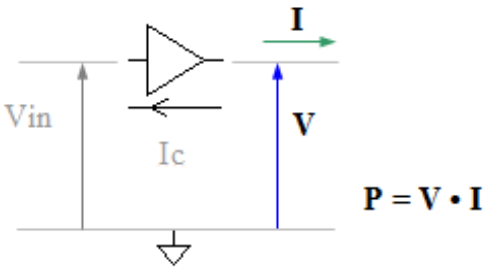
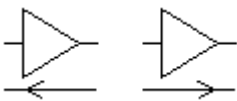
Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, $K(V_{in1}+V_{in2})$
<p>Piecewise linear amplifier. pwl string defines gain as a function of sum of input voltages $K(V_{in1}+V_{in2})$. Amplifier transfer function is $V(V_{in1}+V_{in2})$ is piecewise linear function, and it always goes through the origin (0,0). See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that $K(V_{in1}+V_{in2})$ is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p>			

O – Voltage controlled amplifier

Symbol	Models	Signals
	PWL	
Views		


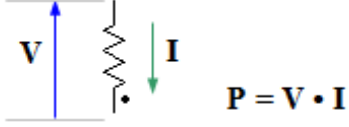
Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, K(Vc)
<p>Piecewise constant voltage controlled amplifier. pwl string defines gain as a function of control voltage K(Vc). At any moment:</p> $V = K(Vc) * Vin.$ <p>See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that K(Vc) is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p>			

O – Current controlled amplifier

Symbol	Models	Signals
	PWL	
Views		

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, K(Ic)
<p>Piecewise constant current controlled amplifier. pwl string defines gain as a function of control current K(Ic). At any moment:</p> $V = K(Ic) * Vin.$ <p>See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that K(Ic) is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p>			

R – Resistor


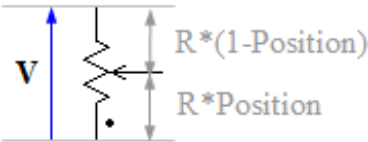
Symbol	Models	Signals
	R PWL PWL-I SubCir	

Model	Parameter	Units	Description
R	R	Ohm	Resistance
Linear resistor: $V = R * I$.			

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, R(V)
<p>Piecewise constant resistor. pwl string defines resistance as a function of voltage across the resistor R(V). Current through the resistor I(V) is piecewise linear function, and it always goes through the origin (0,0). See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that R(V) is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p>			

Model	Parameter	Units	Description
PWL-I	pwl		Comma-separated string, R(I)
<p>Piecewise constant resistor. pwl string defines resistance as a function of current through the resistor R(I). Voltage across the resistor V(I) is piecewise linear function, and it always goes through the origin (0,0). See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that R(I) is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p>			

R – Potentiometer

Symbol	Models	Signals
	Potentiometer	

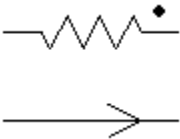
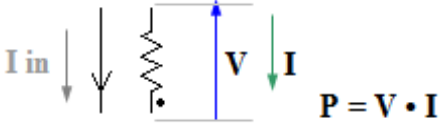
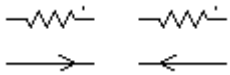
Model	Parameter	Units	Description
Potentiometer	R	Ohm	Resistance
	Position		Position of the wiper (0...1)
Position of the wiper is referenced to the terminal with dot: 0 – wiper is connected to the terminal with dot 1 – wiper is connected to another terminal.			

R – Voltage controlled resistor

Symbol	Models	Signals
	PWL	
Views		

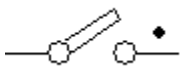
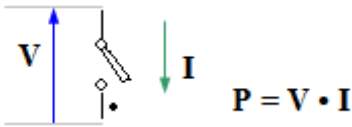
Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, R(Vin)
<p>Piecewise constant voltage controlled resistor. pwl string defines resistance as a function of control voltage R(Vin). At any moment:</p> $V = R(Vin) * I.$ <p>See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that R(Vin) is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p>			

R – Current controlled resistor

Symbol	Models	Signals
	PWL	
Views		

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, R(lin)
<p>Piecewise constant current controlled resistor. pwl string defines resistance as a function of control current R(lin). At any moment:</p> $V = R(lin) * I.$ <p>See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that R(lin) is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p>			

S – Switch

Symbol	Models		Signals
	Off On Step Single Pulse	Clock List File SubCir	

For **Off/On** models and models with **Active** parameter, switch symbol indicates switch position in non-active state:

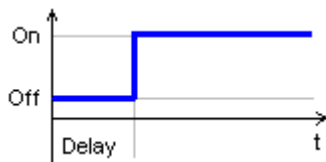


Model	Parameter	Units	Description
Off	No parameters.		
Switch is always in Off state (open circuit).			

Model	Parameter	Units	Description
On	No parameters.		
Switch is always in On state (short circuit).			

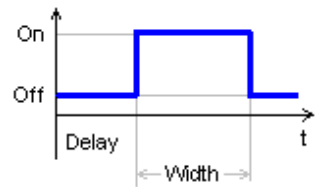
Model	Parameter	Units	Description
Step	Delay	s	Delay before active state.
	Active		Active switch state: Off/On.

Switch goes to **Active** state after **Delay** time. Switching diagram for **Active** = On:



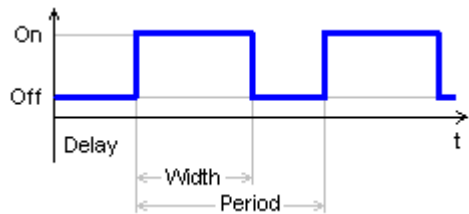
Model	Parameter	Units	Description
Single	Width	s	Pulse width.
	Delay	s	Delay before first pulse starts.
	Active		Active switch state: Off/On.

Single pulse starts after **Delay** time. Switching diagram is shown for **Active** = On:



Model	Parameter	Units	Description
Pulse	Period	s	Period.
	Width	s	Pulse width.
	Delay	s	Delay before first pulse starts.
	Active		Active switch state: Off/On.

Pulses start after **Delay** time. Switching diagram is shown for **Active** = On:

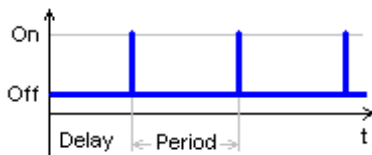


Model	Parameter	Units	Description
Clock	Period	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.
	Active		Active switch state: Off/On.

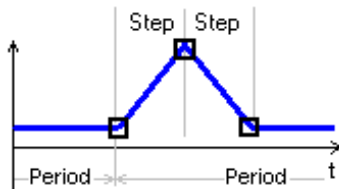
Periodic pulses with width of one simulation step. Pulses start after **Delay** time. Switch goes to **Active** state for one simulation step only.

Unlike **Pulse** model, **Clock** model won't force unnecessary step reduction at the end of the pulse, which may help to accelerate simulation.

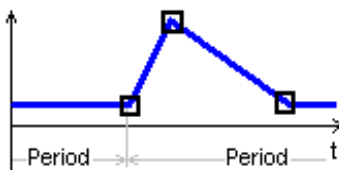
Switching diagram is shown for **Active** = On:



If **Step** parameter is not zero **and** is less than current schematic simulation step, the step is adjusted to provide rise and fall of clock pulse to be equal to **Step**:



Otherwise, the clock pulse is created using current schematic simulation step, which depends on many factors, and cannot be easily predicted:



Model	Parameter	Units	Description
List	List		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Switching sequence is defined in the **List** parameter in the **csv** (comma-separated values) format, as follows:

$t_0, s_0, t_1, s_1, \dots, t_n, s_n$

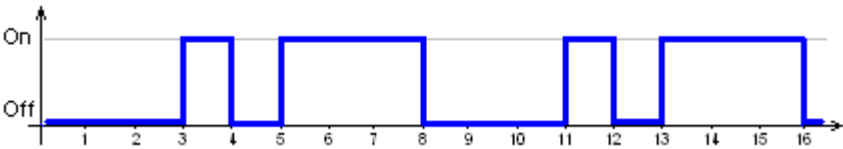
$s_0 \dots s_n$ defines switch state: positive number corresponds to On state, zero or negative number - Off state.
If $t < t_0$, switch is in s_0 state.
At t_0 switch is set to s_0 state, at t_1 switch is set to s_1 state, and so on.
If $t > t_n$, and **Cycle** parameter is set to **No**, switch remains in state s_n . Otherwise the sequence is repeated continuously.

Switching start is delayed by **Delay** time.

Example:

List = 0,0,3,1,4,0,5,1,8,0

If **Cycle = Yes**, **Delay = 0**, the following sequence will be generated:



See *Working with List model* chapter for more details.

Model	Parameter	Units	Description
File	File		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Switching sequence defined in the text file. If **File** parameter does not have a full path, NL5 will search for the file in the directory where schematic file is located, then in the Library directories, and then in the Library directories relative to schematic file directory (see NL5 Manual, *Schematic Properties*).

Switching sequence is defined in the **csv** (comma-separated values) format, as follows:

```
<if first line does not start with a number, it is ignored>
t0,s0
t1,s1
.....
tn,sn
```

s0...sn defines switch state: positive number corresponds to On state, zero or negative number - Off state.
If $t < t_0$, switch is in s0 state.

At t_0 switch is set to s0 state, at t_1 switch is set to s1 state, and so on.

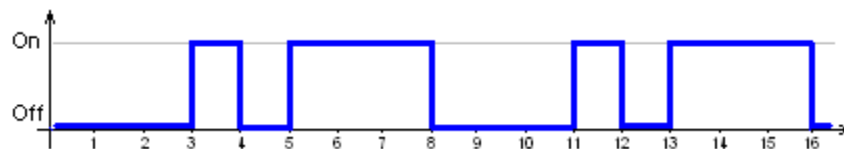
If $t > t_n$, and **Cycle** parameter is set to **No**, switch remains in state sn. Otherwise the sequence is repeated continuously.

Switching start delayed by **Delay** time.

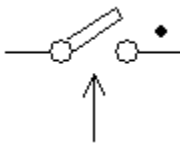
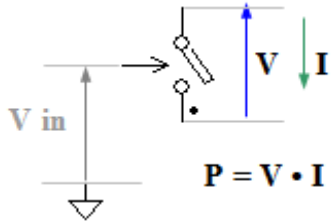
Example:

```
0,0
3,1
4,0
5,1
8,0
```

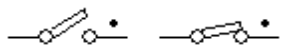
If **Cycle** = **Yes**, **Delay** = 0, the following sequence will be generated:



S – Logic controlled switch

Symbol	Models		Signals
	Switch Off On One-shot Step Single	Pulse Clock List File Steps SubCir	

For **Off/On** models and models with **Active** parameter, switch symbol indicates switch position in non-active state:



Model	Parameter	Units	Description
Switch	Active		Active state: Off/On.
	IC		Initial condition: Off/On.

Logic controlled switch. Switch is set to active or non-active state using following rules:

$V_{in} > \text{logical threshold} \dots : \text{active}$
 $V_{in} < \text{logical threshold} \dots : \text{non-active}$

When calculating DC operating point, switch is set to the state defined in **IC**.

Model	Parameter	Units	Description
Off	No parameters.		
Switch is always in Off state (open circuit).			

Model	Parameter	Units	Description
On	No parameters.		
Switch is always in On state (short circuit).			

Model	Parameter	Units	Description
One-shot	Width	s	Pulse width.
	Active		Active state: Off/On.

One-shot switch. When increasing control signal V_{in} crosses logical threshold, switch is set to **Active** state for **Width** time interval.

If increasing V_{in} crosses logical threshold value while switch is in active state, the **Active** interval is restarted.

Model	Parameter	Units	Description
Step	Delay	s	Delay before active state.
	Active		Active switch state: Off/On.

When control signal V_{in} is below logical threshold, switch is in non-active state. When increasing control signal V_{in} crosses logical threshold, a switching sequence similar to **Step** model of **Switch** component is generated. When decreasing control signal V_{in} drops below logical threshold, switch goes to non-active state immediately.

Model	Parameter	Units	Description
Single	Width	s	Pulse width.
	Delay	s	Delay before first pulse starts.
	Active		Active switch state: Off/On.

When control signal V_{in} is below logical threshold, switch is in non-active state. When increasing control signal V_{in} crosses logical threshold, a switching sequence similar to **Single** model of **Switch** component is generated. When decreasing control signal V_{in} drops below logical threshold, switch goes to non-active state immediately.

Model	Parameter	Units	Description
Pulse	Period	s	Period.
	Width	s	Pulse width.
	Delay	s	Delay before first pulse starts.
	Active		Active switch state: Off/On.

When control signal V_{in} is below logical threshold, switch is in non-active state. When increasing control signal V_{in} crosses logical threshold, a switching sequence similar to **Pulse** model of **Switch** component is generated. When decreasing control signal V_{in} drops below logical threshold, switch goes to non-active state immediately.

Model	Parameter	Units	Description
Clock	Period	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.
	Active		Active switch state: Off/On.
<p>When control signal <i>V_{in}</i> is below logical threshold, switch is in non-active state. When increasing control signal <i>V_{in}</i> crosses logical threshold, a switching sequence similar to Pulse model of Switch component is generated. When decreasing control signal <i>V_{in}</i> drops below logical threshold, switch goes to non-active state immediately.</p>			

Model	Parameter	Units	Description
List	List		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.
<p>When control signal <i>V_{in}</i> is below logical threshold, switch is in s0 state defined in the List parameter. When increasing control signal <i>V_{in}</i> crosses logical threshold, a switching sequence similar to List model of Switch component is generated. When decreasing control signal <i>V_{in}</i> drops below logical threshold, switch goes to s0 state immediately.</p>			

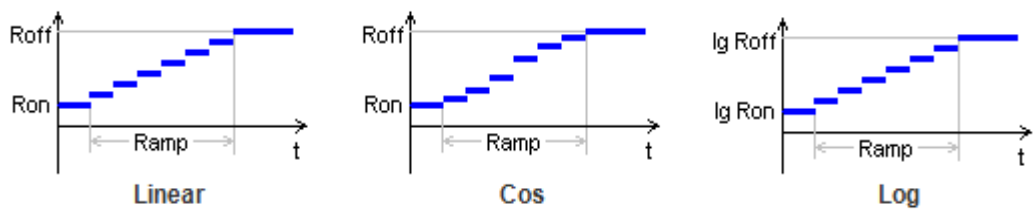
Model	Parameter	Units	Description
File	File		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.
<p>When control signal <i>V_{in}</i> is below logical threshold, switch is in s0 state specified in the File. When increasing control signal <i>V_{in}</i> crosses logical threshold, a switching sequence similar to File model of Switch component is generated. When decreasing control signal <i>V_{in}</i> drops below logical threshold, switch goes to non-active state immediately.</p>			

Model	Parameter	Units	Description
Steps	Roff	Ohm	Off state resistance.
	Ron	Ohm	On state resistance.
	Slope		Type of resistance change: Linear/Cos/Log.
	Ramp	s	Resistance ramp time.
	Steps		Number of resistance steps in the ramp.
	IC		Initial condition: Off/On.

Switch with resistance ramping. When increasing input voltage V_{in} crosses logical threshold, switch resistance starts ramping from **Roff** to **Ron**. When decreasing input voltage V_{in} crosses logical threshold, switch resistance starts ramping from **Ron** to **Roff**.

Resistance is changing during **Ramp** time interval, with number of steps specified by **Steps** parameter. If **Steps** = 0, resistance is changed instantly.

Slope parameter specifies how resistance is changing during the ramp:



When calculating DC operating point switch is set to the state specified in **IC**.

S – Voltage controlled switch

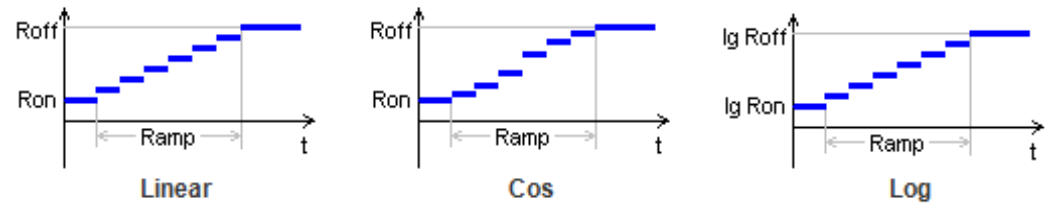
Symbol	Models	Signals
	Switch Off On One-shot Steps SubCir	
Views 		
For Off/On models and models with Active parameter, switch symbol indicates switch position in non-active state: 		

Model	Parameter	Units	Description
Switch	Threshold	V	Voltage threshold.
	Hysteresis	V	Hysteresis.
	Active		Active state: Off/On.
	IC		Initial condition: Off/On.
Voltage controlled switch. Switch is set to active or non-active state using following rules: $V_{in} > \text{Threshold} + \text{Hysteresis}/2 \dots$: active $V_{in} < \text{Threshold} - \text{Hysteresis}/2 \dots$: non-active Otherwise : previous state When calculating DC operating point switch is set to the state defined in IC . Switching diagram for Active = On: 			

Model	Parameter	Units	Description
Off	No parameters.		
Switch is always in Off state (open circuit).			

Model	Parameter	Units	Description
On	No parameters.		
Switch is always in On state (short circuit).			

Model	Parameter	Units	Description
One-shot	Width	s	Pulse width.
	Threshold	V	Voltage threshold.
	Active		Active state: Off/On.
One-shot pulse generator. When increasing input voltage V_{in} crosses Threshold value, switch is set to Active state for Width time interval.			
If increasing V_{in} crosses logical threshold value while switch is in active state, the Active interval is restarted.			

Model	Parameter	Units	Description
Steps	Threshold	V	Voltage threshold.
	Hysteresis	V	Hysteresis.
	Roff	Ohm	Off state resistance.
	Ron	Ohm	On state resistance.
	Slope		Type of resistance change: Linear/Cos/Log.
	Ramp	s	Resistance ramp time.
	Steps		Number of resistance steps in the ramp.
	IC		Initial condition: Off/On.
Switch with resistance ramping. When increasing input voltage V_{in} crosses Threshold + Hysteresis/2 value, switch resistance starts ramping from Roff to Ron . When decreasing input voltage V_{in} crosses Threshold - Hysteresis/2 value, switch resistance starts ramping from Ron to Roff .			
Resistance is changing during Ramp time interval, with number of steps specified by Steps parameter. If Steps = 0, resistance is changed instantly.			
Slope parameter specifies how resistance is changing during the ramp:			
 <div style="display: flex; justify-content: space-around; margin-top: 10px;"> Linear Cos Log </div>			
When calculating DC operating point switch is set to the state specified in IC .			

S – Current controlled switch

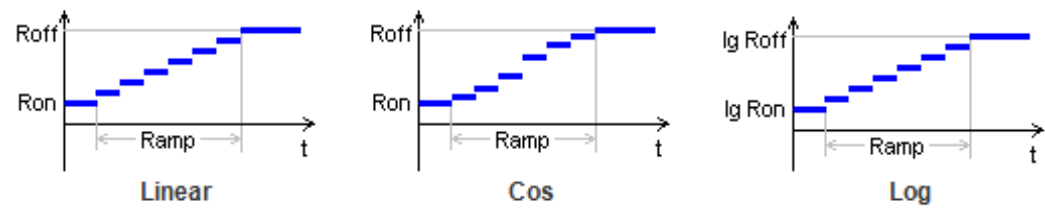
Symbol	Models	Signals
	Switch Off On	One-shot Steps SubCir
Views 		
For Off/On models and models with Active parameter, switch symbol indicates switch position in non-active state: 		

Model	Parameter	Units	Description
Switch	Threshold	A	Current threshold.
	Hysteresis	A	Hysteresis.
	Active		Active state: Off/On.
	IC		Initial condition: Off/On.
<p>Current controlled switch. Switch is set to active or non-active state using following rules:</p> <p>$I_{in} > \text{Threshold} + \text{Hysteresis}/2 \dots$: active $I_{in} < \text{Threshold} - \text{Hysteresis}/2 \dots$: non-active Otherwise : previous state</p> <p>When calculating DC operating point switch is set to the state defined in IC.</p> <p>Switching diagram for Active = On:</p>			

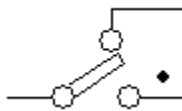
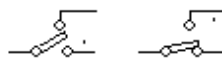

Model	Parameter	Units	Description
Off	No parameters.		
Switch is always in Off state (open circuit).			

Model	Parameter	Units	Description
On	No parameters.		
Switch is always in On state (short circuit).			

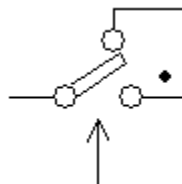
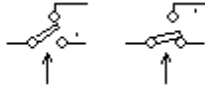
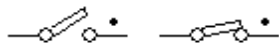
Model	Parameter	Units	Description
One-shot	Width	s	Pulse width.
	Threshold	A	Current threshold.
	Active		Active state: Off/On.
One-shot pulse generator. When increasing input current I_{in} crosses Threshold value, switch is set to Active state for Width time interval.			
If increasing I_{in} crosses logical threshold value while switch is in active state, the Active interval is restarted.			

Model	Parameter	Units	Description
Steps	Threshold	A	Current threshold.
	Hysteresis	A	Hysteresis.
	Roff	Ohm	Off state resistance.
	Ron	Ohm	On state resistance.
	Slope		Type of resistance change: Linear/Cos/Log.
	Ramp	s	Resistance ramp time.
	Steps		Number of resistance steps in the ramp.
	IC		Initial condition: Off/On.
Switch with resistance ramping. When increasing input current I_{in} crosses Threshold + Hysteresis/2 value, switch resistance starts ramping from Roff to Ron . When decreasing input current I_{in} crosses Threshold - Hysteresis/2 value, switch resistance starts ramping from Ron to Roff .			
Resistance is changing during Ramp time interval, with number of steps specified by Steps parameter. If Steps = 0, resistance is changed instantly.			
Slope parameter specifies how resistance is changing during the ramp:			
			
When calculating DC operating point switch is set to the state specified in IC .			

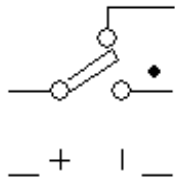
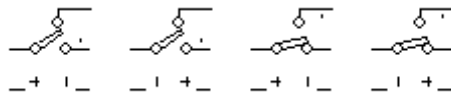
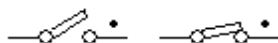
S – SPDT switch

Symbol		Models		Signals
		Off On Step Single Pulse	Clock List File SubCir	
Views				
<p>All models and operation of the SPDT (single pole, double throw) switch are similar to single pole Switch, with Off and On states defined as follows:</p> <p>Off state: “common to pin with dot” - open, “common to another pin” - short. On state: “common to pin with dot” - short, “common to another pin” - open.</p> <p>For Off/On models and models with Active parameter, switch symbol indicates switch position in non-active state:</p> 				

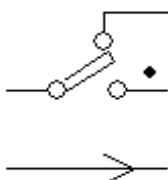
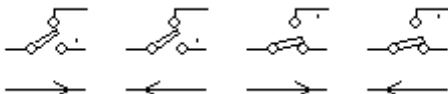
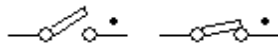
S – SPDT logic controlled switch

Symbol	Models		Signals
	Switch Off On One-shot Step Single	Pulse Clock List File Steps SubCir	
Views			
<p>All models and operation of the SPDT (single pole, double throw) logic controlled switch are similar to single pole Logic controlled switch, with Off and On states defined as follows:</p> <p>Off state: “common to pin with dot” - open, “common to another pin” - short. On state: “common to pin with dot” - short, “common to another pin” - open.</p> <p>For Off/On models and models with Active parameter, switch symbol indicates switch position in non-active state:</p> 			


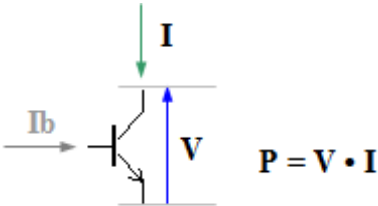
S – SPDT voltage controlled switch

Symbol		Models		Signals
		Switch Off On	One-shot Steps SubCir	
Views				
	<p>All models and operation of the SPDT (single pole, double throw) voltage controlled switch are similar to single pole Voltage controlled switch, with Off and On states defined as follows:</p> <p>Off state: “common to pin with dot” - open, “common to another pin” - short. On state: “common to pin with dot” - short, “common to another pin” - open.</p> <p>For Off/On models and models with Active parameter, switch symbol indicates switch position in non-active state:</p> 			

S – SPDT current controlled switch

Symbol		Models		Signals
		Switch Off On	One-shot Steps SubCir	
Views				
	<p>All models and operation of SPDT (single pole, double throw) current controlled switch are similar to single pole Current controlled switch component, with Off and On states defined as follows:</p> <p>Off state: “common to pin with dot” - open, “common to another pin” - short. On state: “common to pin with dot” - short, “common to another pin” - open.</p> <p>For Off/On models and models with Active parameter, switch symbol indicates switch position in non-active state:</p> 			

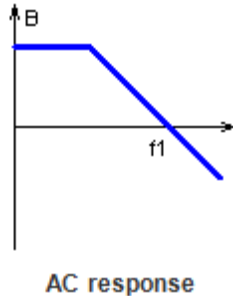
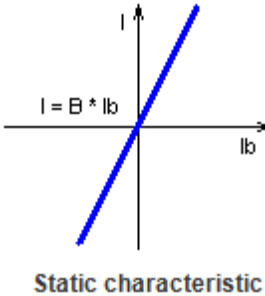
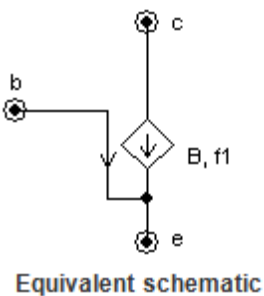
T – NPN transistor

Symbol	Models	Signals
	Linear Switch Transistor SubCir	

Model	Parameter	Units	Description
Linear	B	A/A	Gain (beta)
	f1	Hz	Unit gain frequency.
	IC	A	Initial condition: collector current.

Linear BJT transistor. Current controlled current source with specified bandwidth. **B** is open loop gain (beta). Frequency response consists of one pole, **f1** is unit gain frequency. **B** and **f1** can be set to infinity (**inf**).

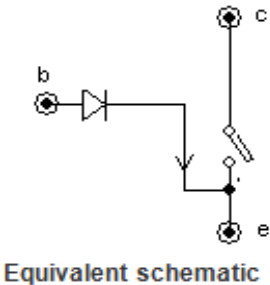
When calculating DC operating point, if **f1** is not infinity and **IC** is defined, collector current is set to specified output current **IC**. If **IC** is blank, static characteristic is used.



Model	Parameter	Units	Description
Switch	Vbe	V	Forward voltage drop of base-emitter diode.
	IC		Initial condition of base-emitter diode: Off/On.

BJT transistor switch. Current controlled switch with a base-emitter diode. Switch is closed if diode current is non-zero.

When calculating DC operating point, the diode is set to the state specified in **IC**.



Model	Parameter	Units	Description
Transistor	B	A/A	Gain (beta)
	f1	Hz	Unit gain frequency.
	Vbe	V	Forward voltage drop of base-emitter diode.
	Vsat	V	Collector-emitter saturation voltage drop.
	IC	A	Initial condition: collector current.
	ICbe		Initial condition of base-emitter diode: Off/On.
	ICbc		Initial condition of base-collector diode: Off/On.

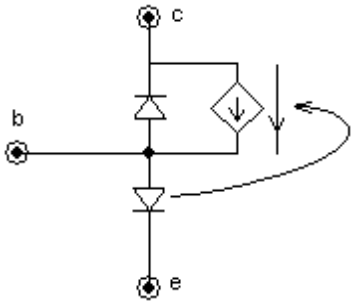
BJT transistor. Simplified Ebers-Moll BJT transistor model with saturation. It consists of two diodes (base-emitter and base-collector), and current source controlled by current through base-emitter diode with gain “alpha”:

$$\alpha = \frac{\beta}{1 + \beta}$$

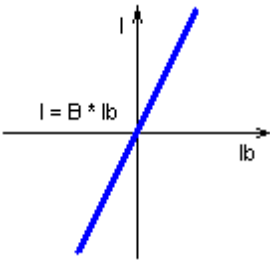
If collector-emitter voltage is higher than **Vsat**, base-collector diode is open, transistor is not saturated, and behaves as **Linear** model (current controlled current source with specified bandwidth). **B** is open loop gain (beta). Low signal frequency response consists of one pole, **f1** is unit gain frequency. **B** and **f1** can be set to infinity (**inf**).

If collector voltage drops below **Vsat**, base-collector diode is closed, and transistor is saturated: collector-emitter voltage is equal to **Vsat**.

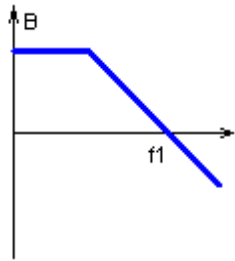
When calculating DC operating point, if **f1** is not infinity and **IC** is defined, collector current is set to specified output current **IC**. If **IC** is blank, static characteristic is used. Base-emitter diode is set to the state specified in **ICbe**, Base-collector diode is set to the state specified in **ICbc**.



Equivalent schematic

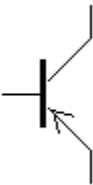
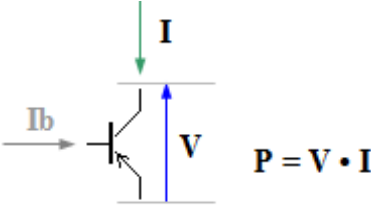


Non-saturated static characteristic



Low signal AC response

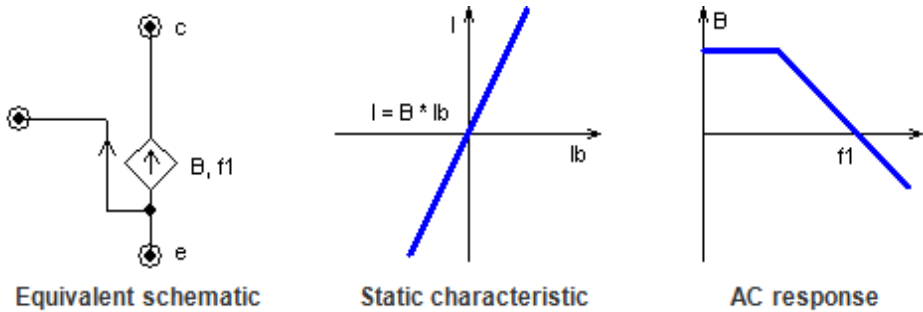
T – PNP transistor

Symbol	Models	Signals
	Linear Switch Transistor SubCir	

Model	Parameter	Units	Description
Linear	B	A/A	Gain (beta)
	f1	Hz	Unit gain frequency.
	IC	A	Initial condition: collector current.

Linear BJT transistor. Current controlled current source with specified bandwidth. **B** is open loop gain (beta). Frequency response consists of one pole, **f1** is unit gain frequency. **B** and **f1** can be set to infinity (**inf**).

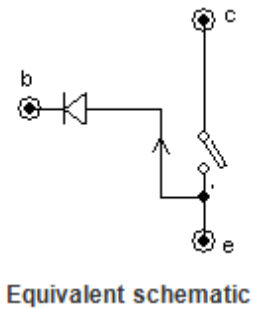
When calculating DC operating point, if **f1** is not infinity and **IC** is defined, collector current is set to specified output current **IC**. If **IC** is blank, static characteristic is used.



Model	Parameter	Units	Description
Switch	Vbe	V	Forward voltage drop of base-emitter diode.
	IC		Initial condition of base-emitter diode: Off/On.

BJT transistor switch. Current controlled switch with a base-emitter diode. Switch is closed if diode current is non-zero.

When calculating DC operating point, the diode is set to the state specified in **IC**.



Model	Parameter	Units	Description
Transistor	B	A/A	Gain (beta)
	f1	Hz	Unit gain frequency.
	Vbe	V	Forward voltage drop of base-emitter diode.
	Vsat	V	Collector-emitter saturation voltage drop.
	IC	A	Initial condition: collector current.
	ICbe		Initial condition of base-emitter diode: Off/On.
	ICbc		Initial condition of base-collector diode: Off/On.

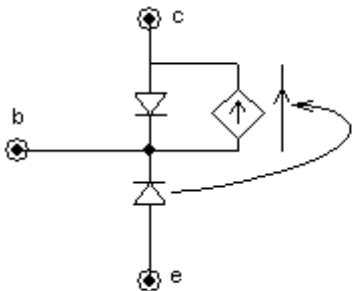
BJT transistor. Simplified Ebers-Moll BJT transistor model with saturation. It consists of two diodes (base-emitter and base-collector), and current source controlled by current through base-emitter diode with gain “alpha”:

$$\alpha = \frac{\beta}{1 + \beta}$$

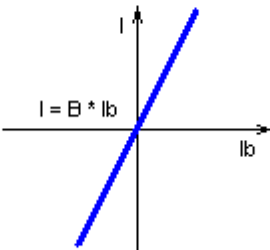
If collector-emitter voltage is negative, and less than **-Vsat**, base-collector diode is open, transistor is not saturated, and behaves as **Linear** model (current controlled current source with specified bandwidth). **B** is open loop gain (beta). Low signal frequency response consists of one pole, **f1** is unit gain frequency. **B** and **f1** can be set to infinity (**inf**).

If collector voltage is higher than **-Vsat**, base-collector diode is closed, and transistor is saturated: collector-emitter voltage is equal to **-Vsat**.

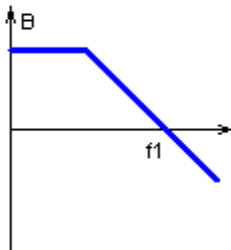
When calculating DC operating point, if **f1** is not infinity and **IC** is defined, collector current is set to specified output current **IC**. If **IC** is blank, static characteristic is used. Base-emitter diode is set to the state specified in **ICbe**, Base-collector diode is set to the state specified in **ICbc**.



Equivalent schematic

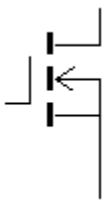
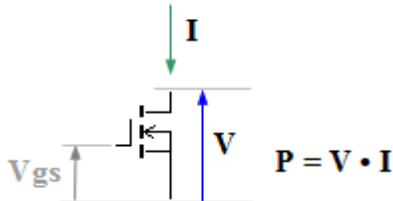


Non-saturated static characteristic



Low signal AC response

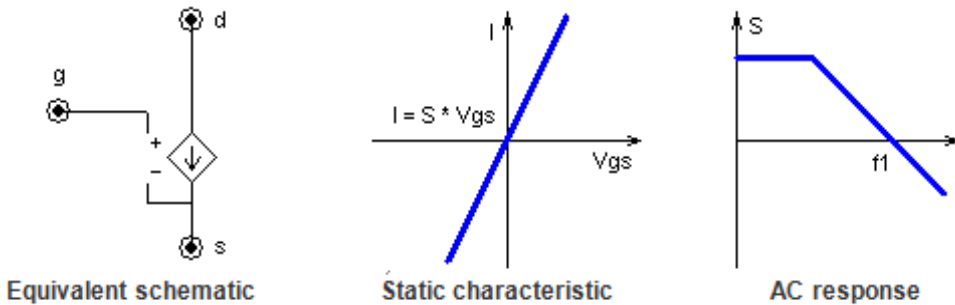
T – N-FET

Symbol	Models	Signals
	Linear Switch Switch-D FET FET-D SubCir	

Model	Parameter	Units	Description
Linear	S	A/V	Slope
	f1	Hz	Unit gain frequency.
	IC	A	Initial condition: drain current.

Linear FET transistor. Voltage controlled current source with specified bandwidth. **S** is open loop slope. Frequency response consists of one pole, **f1** is unit gain frequency. **S** and **f1** can be set to infinity (**inf**).

When calculating DC operating point, if **f1** is not infinity and **IC** is defined, drain current is set to specified output current **IC**. If **IC** is blank, static characteristic is used.



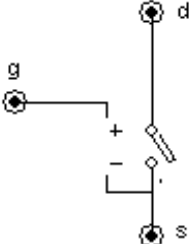
This model of FET transistor does not have a **body diode**. If you need a body diode, you should add it as an external component.

Model	Parameter	Units	Description
Switch	Vth	V	Threshold.
	IC		Initial condition of the switch: Off/On.

FET switch. Voltage controlled switch. Switch is closed if gate-source voltage exceeds threshold **Vth**.


When calculating DC operating point switch is set to the state specified in **IC**.

This model of FET transistor does not have a **body diode**. If you need a body diode, you should add it as an external component, or use **Switch-D** model..



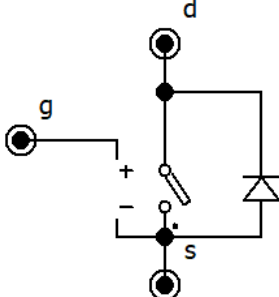
The schematic shows a voltage-controlled switch. A gate terminal (g) is connected to the control input of a switch symbol. The switch is connected between a drain terminal (d) and a source terminal (s). The switch is represented by two parallel lines that cross in the middle, with a diagonal line through the center. The source terminal (s) is connected to ground.

Equivalent schematic

Model	Parameter	Units	Description
	Vth	V	Threshold.
	Vd	V	Body diode forward voltage drop
	IC		Initial condition: Off/On/Diode

FET switch with body diode. The model is the same as Switch model with one addition: a body diode connected between drain and source.

When calculating DC operating point switch is set to the state specified in **IC**. “Diode” state means diode is closed.



The schematic shows a voltage-controlled switch with an internal body diode. A gate terminal (g) is connected to the control input of a switch symbol. The switch is connected between a drain terminal (d) and a source terminal (s). A diode is connected in parallel with the switch, with its anode at the source (s) and its cathode at the drain (d). The source terminal (s) is connected to ground.

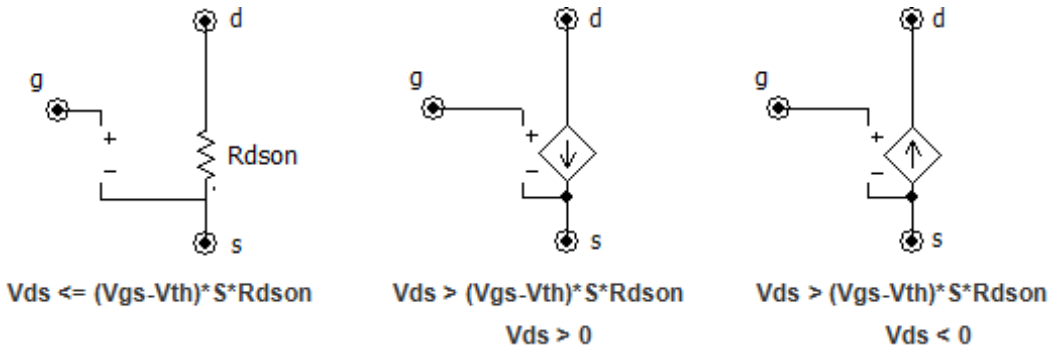
Equivalent schematic

Model	Parameter	Units	Description
FET	S	A/V	Slope.
	Vth	V	Threshold.
	Rdson	Ohm	Rdson resistance.
	IC		Initial condition: Off/R/Plus/Minus

FET transistor. The model has 3 modes of operation.

- 1. $V_{gs} \leq V_{th}$: $I = 0$ (open)
- 2. $V_{gs} > V_{th}$, $V_{ds} \leq (V_{gs} - V_{th}) * S * R_{dson}$..: $V = I * R_{dson}$ (resistor)
- 3. $V_{gs} > V_{th}$, $V_{ds} > (V_{gs} - V_{th}) * S * R_{dson}$..: $I = (V_{gs} - V_{th}) * S$ (current source)

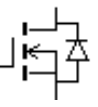
FET works similar for positive and negative drain-source voltage, current direction changes accordingly.
Equivalent schematics ($V_{gs} > V_{th}$):



When calculating DC operating point, transistor is set to an initial state specified by IC parameter as follows:

- Off ...: $I = 0$ (open)
- R ...: $V = I * R_{dson}$ (resistor)
- Plus ...: $V_{ds} > 0$, $I = (V_{gs} - V_{th}) * S$ ("positive" current source)
- Minus.: $V_{ds} < 0$, $I = (V_{gs} - V_{th}) * S$ ("negative" current source)

This model of FET transistor does not have a **body diode**. If you need a body diode, you should add it as an external component.

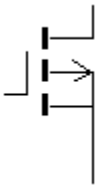
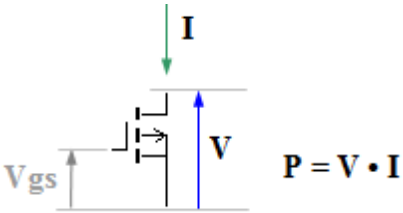
Model	Parameter	Units	Description
FET-D 	S	A/V	Slope.
	Vth	V	Threshold.
	Rdson	Ohm	Rdson resistance.
	Vd	V	Body diode forward voltage drop
	IC		Initial condition: Off/R/Plus/Minus/Diode

FET transistor with body diode. The model is the same as **FET** model with one addition: a body diode connected between drain and source. In each mode of operation, the diode may be open or close, depending on external conditions.

When calculating DC operating point, transistor is set to an initial state specified by **IC** as follows:

Off . . . : $I = 0$ (open)
R : $V = I * \mathbf{Rdson}$ (resistor)
Plus . . : $V_{ds} > 0$, $I = (V_{gs} - \mathbf{Vth}) * \mathbf{S}$ ("positive" current source)
Minus . : $V_{ds} < 0$, $I = (V_{gs} - \mathbf{Vth}) * \mathbf{S}$ ("negative" current source)
Diode . : The body diode is conducting.

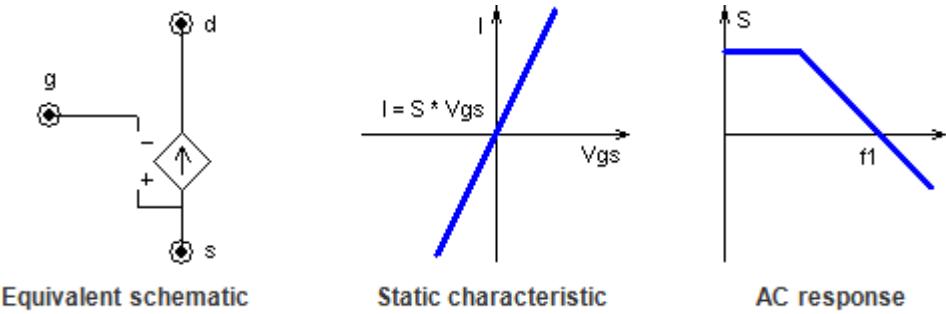
T – P-FET

Symbol	Models	Signals
	Linear Switch Switch-D FET FET-D SubCir	

Model	Parameter	Units	Description
Linear	S	A/V	Slope
	f1	Hz	Unit gain frequency.
	IC	A	Initial condition: drain current.

Linear FET transistor. Voltage controlled current source with specified bandwidth. **S** is open loop slope. Frequency response consists of one pole, **f1** is unit gain frequency. **S** and **f1** can be set to infinity (**inf**).

When calculating DC operating point, if **f1** is not infinity and **IC** is defined, drain current is set to specified output current **IC**. If **IC** is blank, static characteristic is used.



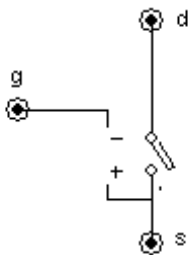
This model of FET transistor does not have a **body diode**. If you need a body diode, you should add it as an external component.

Model	Parameter	Units	Description
Switch	Vth	V	Threshold.
	IC		Initial condition of the switch: Off/On.


FET switch. Voltage controlled switch. Switch is closed if gate-source voltage is less than threshold **Vth**.

When calculating DC operating point switch is set to the state specified in **IC**.

This model of FET transistor does not have a **body diode**. If you need a body diode, you should add it as an external component.

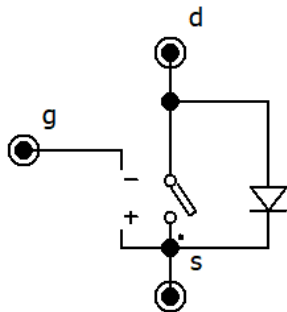


Equivalent schematic

Model	Parameter	Units	Description
	Vth	V	Threshold.
	Vd	V	Body diode forward voltage drop
	IC		Initial condition: Off/On/Diode

FET switch with body diode. The model is the same as Switch model with one addition: a body diode connected between drain and source.

When calculating DC operating point switch is set to the state specified in **IC**. “Diode” state means diode is closed.



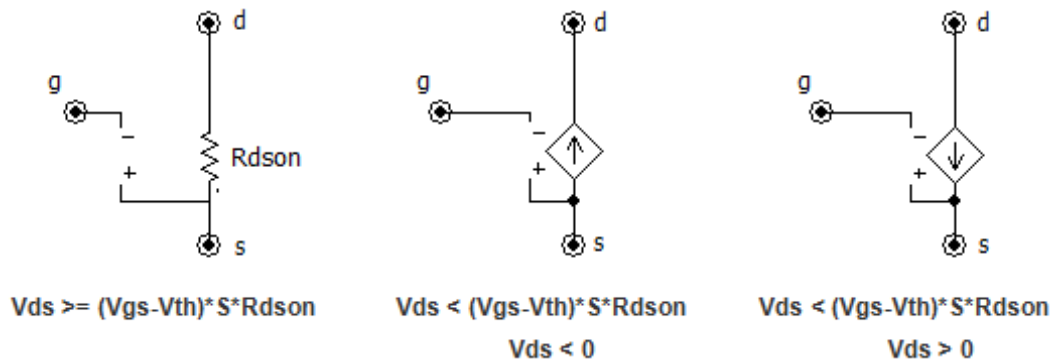
Equivalent schematic

Model	Parameter	Units	Description
FET	S	A/V	Slope.
	Vth	V	Threshold.
	Rdson	Ohm	Rdson resistance.
	IC		Initial condition: Off/R/Plus/Minus

FET transistor. The model has 3 modes of operation.

1. $V_{gs} \geq V_{th}$: $I = 0$ (open)
2. $V_{gs} < V_{th}$, $V_{ds} \geq (V_{gs} - V_{th}) * S * R_{dson}$...: $V = I * R_{dson}$ (resistor)
3. $V_{gs} < V_{th}$, $V_{ds} < (V_{gs} - V_{th}) * S * R_{dson}$...: $I = (V_{gs} - V_{th}) * S$ (current source)


FET works similar for positive and negative drain-source voltage, current direction changes accordingly.
Equivalent schematics ($V_{gs} < V_{th}$):



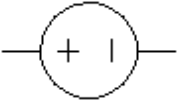
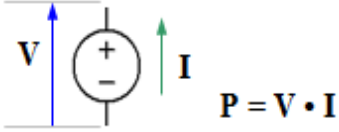
When calculating DC operating point, transistor is set to an initial state specified by **IC** parameter as follows:

- Off** ...: $I = 0$ (open)
- R** ...: $V = I * R_{dson}$ (resistor)
- Plus** ...: $V_{ds} < 0$, $I = (V_{gs} - V_{th}) * S$ ("positive" current source)
- Minus** ...: $V_{ds} > 0$, $I = (V_{gs} - V_{th}) * S$ ("negative" current source)

This model of FET transistor does not have a **body diode**. If you need a body diode, you should add it as an external component.

Model	Parameter	Units	Description
FET-D 	S	A/V	Slope.
	Vth	V	Threshold.
	Rdson	Ohm	Rdson resistance.
	Vd	V	Body diode forward voltage drop
	IC		Initial condition: Off/R/Plus/Minus/Diode
<p>FET transistor with body diode. The model is the same as FET model with one addition: a body diode connected between drain and source. In each mode of operation, the diode may be open or close, depending on external conditions.</p> <p>When calculating DC operating point, transistor is set to an initial state specified by IC as follows:</p> <p>Off . . . : $I = 0$ (open)</p> <p>R : $V = I * R_{dson}$ (resistor)</p> <p>Plus . . : $V_{ds} < 0$, $I = (V_{gs} - V_{th}) * S$ ("positive" current source)</p> <p>Minus . : $V_{ds} > 0$, $I = (V_{gs} - V_{th}) * S$ ("negative" current source)</p> <p>Diode . : The body diode is conducting.</p>			

V – Voltage source

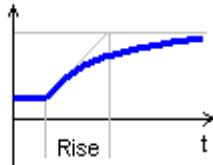

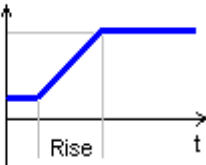
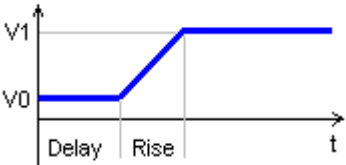
Symbol	Models	Signals
	V Step Single Pulse Clock Sin Sweep Function List File SubCir	 $P = V \cdot I$

Model	Parameter	Units	Description
V	V	V	Voltage.

Constant voltage = V.

Model	Parameter	Units	Description
Step	V1	V	Step On voltage.
	V0	V	Step Off voltage.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Step rise length.
	Delay	s	Delay before step starts.

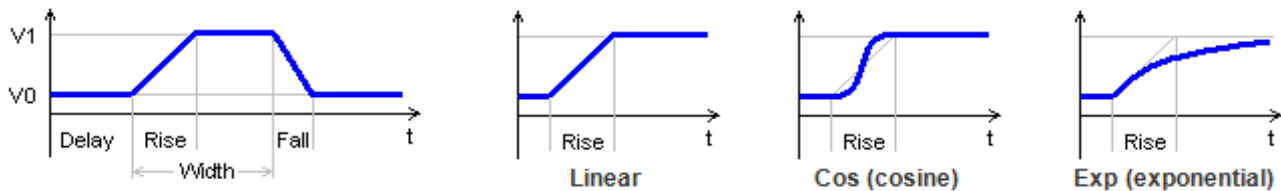
Step starts after **Delay** time. If **Rise** is non-zero, 3 **Slope** types are available.



LinearCos (cosine)Exp (exponential)

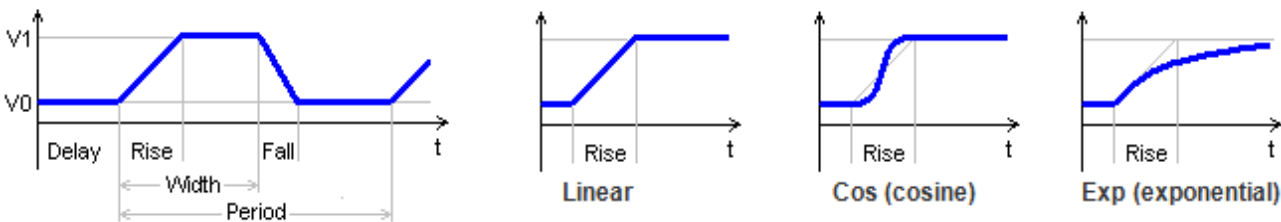
Model	Parameter	Units	Description
Single	V1	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before pulse starts.

Single pulse starts after **Delay** time. **Rise** time is included into **Width**, **Fall** time is **not** included into **Width**. **Slope** type applies both to pulse rise and fall (if non-zero).



Model	Parameter	Units	Description
Pulse	V1	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Period	s	Period.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before first pulse starts.

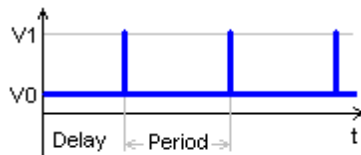
Periodic pulse source. Pulses start after **Delay** time. **Rise** time is included into **Width**, **Fall** time is **not** included into **Width**. **Slope** type applies both to pulse rise and fall (if non-zero).



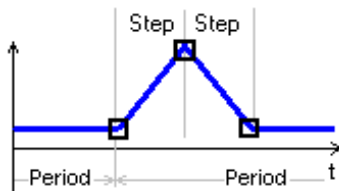
Model	Parameter	Units	Description
Clock	V1	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Period	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.

Periodic pulses with width of one simulation step. Pulses start after **Delay** time.

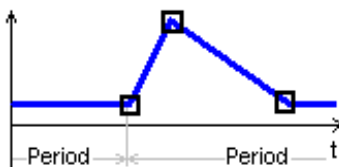
Clock model is recommended to produce a constant frequency clock signal for C-code, DLL, logical components, etc. Unlike **Pulse** model, it won't force unnecessary step reduction at the end of the pulse, which may help to accelerate simulation.



If **Step** parameter is not zero **and** is less than current schematic simulation step, the step is adjusted to provide rise and fall of clock pulse to be equal to **Step**:

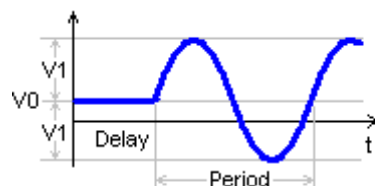


Otherwise, the clock pulse is created using current schematic simulation step, which depends on many factors, and cannot be easily predicted:

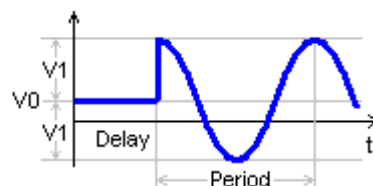


Model	Parameter	Units	Description
Sin	V1	V	Voltage amplitude.
	V0	V	Voltage baseline.
	Period	s	Period.
	Phase	deg	Phase.
	Decay	1/s	Decay constant
	Delay	s	Delay before sine signal starts.

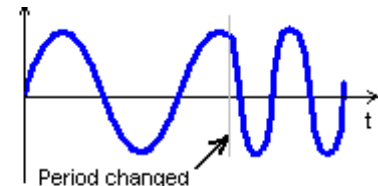
Sinusoidal signal starts after **Delay** time. **Phase** is sine phase in degrees at the moment when signal starts. If transient is paused, sine period changed, and then transient is continued, the phase of the signal remains continuous, providing smooth sine signal of variable frequency. If **Decay** is not zero, the sine signal is exponentially damped with time constant = $1/\text{Decay}$.



Phase = 0



Phase = 90



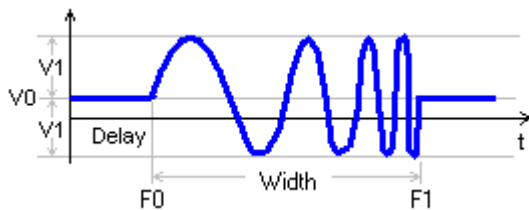
Model	Parameter	Units	Description
Sweep	V1	V	Voltage amplitude.
	V0	V	Voltage baseline.
	Width	s	Width of the signal.
	F0	Hz	Start frequency.
	F1	Hz	End frequency.
	Type		Signal type: Linear/Exp.
	Delay	s	Delay before signal starts.

Sinusoidal signal with variable frequency starts after **Delay** time. Signal frequency changes during **Width** interval from **F0** to **F1** linearly or exponentially, depending on specified **Type**.

If **F0 = F1**, then one period of frequency $1/\text{Width}$ will be generated.

If lowest frequency is set to zero and **Type** = Exp, then lowest frequency $0.01/\text{Width}$ will be used.

If needed, the highest frequency will be increased to provide integer number of signal periods, so that signal phase at the beginning and at the end of **Width** interval is exactly zero.



Model	Parameter	Units	Description
Function	f	V	Function

Arbitrary function. **f** defines voltage as a function of the following variables:

- t** - current time
- V(name)** - voltage on the component **name**
- I(name)** - current through the component **name**
- P(name)** – power on the component **name**
- S(name)** – state of the component **name**

where **name** is the name of the component in the schematic. If **f** is blank, voltage is zero.

Example:

```
f = sin(t) * (1+cos(t*.01))
f = V(R1) * I(R1)
```

Please note that **V**, **I**, **P**, and **S** variables are taken at previous calculation step. This may affect stability of the schematic with closed loop.

Model	Parameter	Units	Description
List	List		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Piecewise linear signal is defined by **List** parameter in the **csv** (comma-separated values) format, as follows:

```
t0,V0,t1,V1,...,tn,Vn
```

where all **t** and **V** can be numerical values or expressions.

If $t < t_0$, signal is **V0**.

If $t_0 < t < t_1$, signal value is linearly interpolated between **V0** and **V1**, etc.

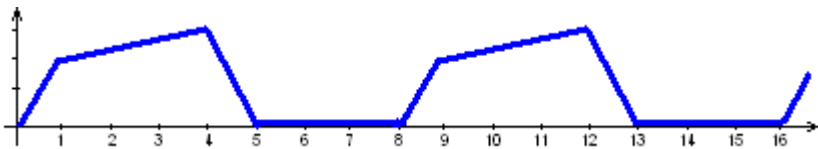
If $t > t_n$, and **Cycle** parameter is set to **No**, the signal value is **Vn**. Otherwise the signal defined in $t_0 \dots t_n$ interval is repeated continuously.

Signal start is delayed by **Delay** time.

Example:

```
List = 0,0,1,2,4,3,5,0,8,0
```

If **Cycle = Yes**, **Delay = 0**, the following voltage will be generated:



See *Working with List source* chapter for more details.

Model	Parameter	Units	Description
File	File		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Piecewise linear signal is defined in the text file. If **File** parameter does not have a full path, NL5 will search for the file in the directory where schematic file is located, then in the Library directories, and then in the Library directories relative to schematic file directory (see NL5 Manual, *Schematic Properties*).

Signal is defined in the **csv** (comma-separated values) format, as follows:

```
<if first line does not start with a number, it is ignored >
t0,V0
t1,V1
.....
tn,Vn
```

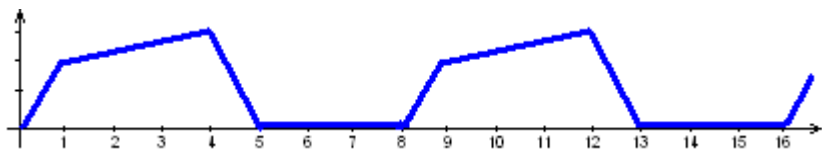
where all t and V can be numerical values or expressions.
If $t < t_0$, signal is V_0 .
If $t_0 < t < t_1$, signal value is linearly interpolated between V_0 and V_1 , etc.
If $t > t_n$, and **Cycle** parameter is set to **No**, the signal value is V_n . otherwise the signal defined in $t_0 \dots t_n$ interval is repeated continuously.

Signal start is delayed by **Delay** time.

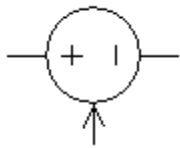
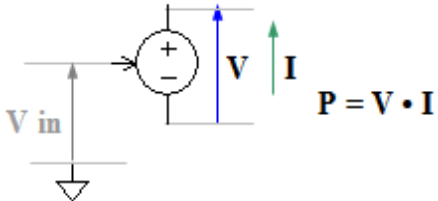
Example. File content:

```
0,0
1,2
4,3
5,0
8,0
```

If **Cycle = Yes**, **Delay = 0**, the following voltage will be generated:



V – Logic controlled voltage source

Symbol	Models	Signals
	<div>V</div> <div>One-shot</div> <div>Step</div> <div>Single</div> <div>Pulse</div> <div>Clock</div> <div>Sin</div> <div>Sweep</div> <div>Function</div> <div>List</div> <div>File</div> <div>SubCir</div>	

Model	Parameter	Units	Description
V	V	V	Voltage.
Constant voltage = V .			

Model	Parameter	Units	Description
One-shot	V1	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Width	s	Pulse width.
<p>One-shot pulse generator. When increasing input voltage <i>Vin</i> crosses logical threshold, voltage pulse of Width duration is generated. V0 is pulse Off level, V1 is pulse On level.</p> <p>If increasing <i>Vin</i> crosses logical threshold value while pulse is being generated, the pulse is restarted.</p>			

Model	Parameter	Units	Description
Step	V1	V	Step On voltage.
	V0	V	Step Off voltage.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Step rise length.
	Delay	s	Delay before step starts.
<p>When control signal <i>Vin</i> is below logical threshold, output is in Off state. When increasing control signal <i>Vin</i> crosses logical threshold, a signal similar to Step model of Voltage source component is generated. When decreasing control signal <i>Vin</i> drops below logical threshold, output goes to Off state immediately.</p>			

Model	Parameter	Units	Description
Single	V1	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before pulse starts.

When control signal *V_{in}* is below logical threshold, output is in Off state. When increasing control signal *V_{in}* crosses logical threshold, a signal similar to **Single** model of **Voltage source** component is generated. When decreasing control signal *V_{in}* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Pulse	V1	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Period	s	Period.
	Width	s	Pulse width.
	Slope		Slope type: Linear/Cos/Exp
	Rise	s	Pulse rise length.
	Fall	s	Pulse fall length.
	Delay	s	Delay before first pulse starts.

When control signal *V_{in}* is below logical threshold, output is in Off state. When increasing control signal *V_{in}* crosses logical threshold, a signal similar to **Pulse** model of **Voltage source** component is generated. When decreasing control signal *V_{in}* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Clock	V1	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Period	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.

When control signal *V_{in}* is below logical threshold, output is in Off state. When increasing control signal *V_{in}* crosses logical threshold, a signal similar to **Clock** model of **Voltage source** component is generated. When decreasing control signal *V_{in}* drops below logical threshold, output goes to Off state immediately.

Model	Parameter	Units	Description
Sin	V1	V	Voltage amplitude.
	V0	V	Voltage baseline.
	Period	s	Period.
	Phase	deg	Phase.
	Decay	1/s	Decay constant
	Delay	s	Delay before sine signal starts.
<p>When control signal <i>Vin</i> is below logical threshold, output voltage is V0. When increasing control signal <i>Vin</i> crosses logical threshold, a signal similar to Sin model of Voltage source component is generated. When decreasing control signal <i>Vin</i> drops below logical threshold, output goes to V0 immediately.</p>			

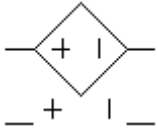
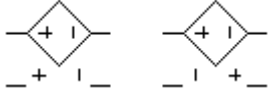
Model	Parameter	Units	Description
Sweep	V1	V	Voltage amplitude.
	V0	V	Voltage baseline.
	Width	s	Width of the signal.
	F0	Hz	Start frequency.
	F1	Hz	End frequency.
	Type		Signal type: Linear/Exp.
	Delay	s	Delay before first signal starts.
<p>When control signal <i>Vin</i> is below logical threshold, output voltage is V0. When increasing control signal <i>Vin</i> crosses logical threshold, a signal similar to Sweep model of Voltage source component is generated. When decreasing control signal <i>Vin</i> drops below logical threshold, output goes to V0 immediately.</p>			

Model	Parameter	Units	Description
Function	f	A	Function
<p>When control signal <i>Vin</i> is below logical threshold, output is zero. When increasing control signal <i>Vin</i> crosses logical threshold, a signal similar to Function model of Voltage source component is generated. If the function is using current time variable <i>t</i>, this moment will be considered as <i>t</i>=0. When decreasing control signal <i>Vin</i> drops below logical threshold, output goes to zero immediately</p>			

Model	Parameter	Units	Description
List	List		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.
<p>When control signal <i>Vin</i> is below logical threshold, output is equal to V0 value of List signal. When increasing control signal <i>Vin</i> crosses logical threshold, a signal similar to List model of Voltage source component is generated. This moment is also considered as <i>t</i>=0 for the List signal. When decreasing control signal <i>Vin</i> drops below logical threshold, output goes to V0 immediately.</p>			

Model	Parameter	Units	Description
File	File		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.
<p>When control signal V_{in} is below logical threshold, output is equal to V0 value specified in the File. When increasing control signal V_{in} crosses logical threshold, a signal similar to File model of Voltage source component is generated. This moment is also considered as $t=0$ for the File signal. When decreasing control signal V_{in} drops below logical threshold, output goes to V0 immediately.</p>			

V – Voltage controlled voltage source

Symbol	Models	Signals
	Linear V Function PWL	VCO One-shot PWM SubCir
Views		

Model	Parameter	Units	Description
Linear	K	V/V	Gain
Linear voltage controlled voltage source: $V = K * V_{in}$.			

Model	Parameter	Units	Description
V	V	V	Voltage.
Constant voltage = V .			

Model	Parameter	Units	Description
Function	f	V	Output as function of the input.
	IC	V	Initial condition: output voltage.
<p>Arbitrary function f defines output voltage as a function of the following variables:</p> <ul style="list-style-type: none"> x – input voltage V_{in} t - current time V(name) - voltage on the component name I(name) - current through the component name P(name) – power on the component name S(name) – state of the component name <p>where name is the name of the component in the schematic. If f is blank, output is zero.</p> <p>Example:</p> <pre>f = x*x f = x * sin(t) f = P(r1) + P(r2)</pre> <p>When calculating DC operating point, and in AC analysis, output is set to specified output voltage IC. Please note that variable x (input voltage V_{in}) and variables V, I, P, and S are taken at previous calculation step. This may affect stability of the schematic with closed loop.</p>			

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, K(Vin)
<p>Piecewise linear voltage controlled voltage source. pwl string defines gain as a function of input voltage K(Vin). The transfer function of the source V(Vin) is piecewise linear function, and it always goes through the origin (0,0). See <i>Working with PWL model</i> chapter for details.</p> <p>Please note that K(Vin) is piecewise constant function, although the model and parameter are still called pwl for historical reasons.</p>			

Model	Parameter	Units	Description
VCO	V1	V	Voltage amplitude.
	V0	V	Voltage baseline or Off level.
	dFdV	Hz/V	Gain.
	Type		Signal type: Sin/Square/Triangle/Sawtooth.
	Phase	deg	Phase.
<p>Voltage controlled oscillator. Output voltage is a signal with frequency equal to:</p> $f(\text{Hz}) = \text{dFdV} * V_{in}.$ <p>For Sine signal, V0 is baseline, and V1 is amplitude. For Square, Triangle, and Sawtooth signals, V0 is Off level, V1 is On level. Phase is additional phase of the signal, in degrees.</p>			

Model	Parameter	Units	Description
One-shot	V1	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Width	s	Pulse width.
	Threshold	V	Voltage threshold.
<p>One-shot pulse generator. When increasing input voltage <i>Vin</i> crosses Threshold value, voltage pulse of Width duration is generated. V0 is pulse Off level, V1 is pulse On level.</p> <p>If increasing <i>Vin</i> crosses Threshold value while pulse is being generated, the pulse is restarted.</p>			

Model	Parameter	Units	Description
PWM	V1	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	F	Hz	Frequency.
	Vmax	V	Input voltage corresponding to 100% duty.
	Phase	deg	Phase.

Voltage controlled Pulse-Width Modulator. Output voltage is a pulse signal of frequency **F** shifted by **Phase**. Input voltage *Vin* is sampled at the beginning of each cycle of the signal, and width of the output pulse during this cycle is calculated according to the equation:

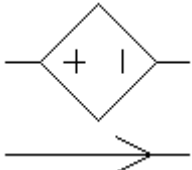
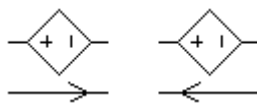
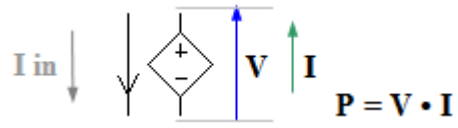
$$\text{width} = 1/F * (V_{in} / V_{max})$$

or

$$\text{duty} = 100\% * (V_{in} / V_{max});$$

If the width is equal or less than zero, a short On pulse with the width equal to the minimum calculation step at that moment will be generated. If the width is equal or greater than period of frequency **F**, a short Off pulse at the end of the period will be generated. As a result, the frequency of the output signal is always **F**.

V – Current controlled voltage source

Symbol	Models	Signals
	Linear V Function PWL	CCO One-shot PWM SubCir
Views 		

Model	Parameter	Units	Description
Linear	K	V/V	Gain
Linear current controlled voltage source. $V = K * I_{in}$.			

Model	Parameter	Units	Description
V	V	V	Voltage.
Constant voltage = V .			

Model	Parameter	Units	Description
Function	f	V	Output as function of the input.
	IC	V	Initial condition: output current.
<p>Arbitrary function f defines output current as a function of the following variables:</p> <ul style="list-style-type: none"> x – input current I_{in} t - current time V(name) - voltage on the component name I(name) - current through the component name P(name) – power on the component name S(name) – state of the component name <p>where name is the name of the component in the schematic. If f is blank, output is zero.</p> <p>Example:</p> <pre>f = x*x f = x * sin(t) f = P(r1) + P(r2)</pre> <p>When calculating DC operating point, and in AC analysis, output is set to specified output current IC. Please note that variable x (input current I_{in}) and variables V, I, P, and S are taken at previous calculation step. This may affect stability of the schematic with closed loop.</p>			

Model	Parameter	Units	Description
PWL	pwl		Comma-separated string, K(<i>I_{in}</i>)

Piecewise linear current controlled voltage source. **pwl** string defines gain as a function of input current K(*I_{in}*). The transfer function of the source V(*I_{in}*) is piecewise linear function, and it always goes through the origin (0,0). See *Working with PWL model* chapter for details.

Please note that K(*I_{in}*) is **piecewise constant** function, although the model and parameter are still called **pwl** for historical reasons.

Model	Parameter	Units	Description
CCO	V1	V	Voltage amplitude.
	V0	V	Voltage baseline or Off level.
	dFdl	Hz/A	Gain.
	Type		Signal type: Sin/Square/Triangle/Sawtooth.
	Phase	deg	Phase.

Current controlled oscillator. Output voltage is a signal with frequency equal to:

$$f(\text{Hz}) = \text{dFdl} * I_{in}.$$

For **Sine** signal, **V0** is baseline, and **V1** is amplitude. For **Square**, **Triangle**, and **Sawtooth** signals, **V0** is Off level, **V1** is On level. **Phase** is additional phase of the signal, in degrees.

Model	Parameter	Units	Description
One-shot	V1	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	Width	s	Pulse width.
	Threshold	A	Current threshold.

One-shot pulse generator. When increasing input current *I_{in}* crosses **Threshold** value, voltage pulse of **Width** duration is generated. **V0** is pulse Off level, **V1** is pulse On level.

If increasing *I_{in}* crosses **Threshold** value while pulse is being generated, the pulse is restarted.

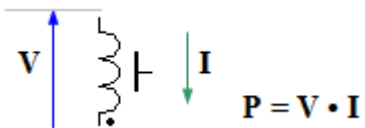
Model	Parameter	Units	Description
PWM	V1	V	Pulse On voltage.
	V0	V	Pulse Off voltage.
	F	Hz	Frequency.
	I_{max}	A	Input current corresponding to 100% duty.
	Phase	deg	Phase.
<p>Current controlled Pulse-Width Modulator. Output voltage is a pulse signal of frequency F shifted by Phase. Input current <i>I_{in}</i> is sampled at the beginning of each cycle of the signal, and width of the output pulse during this cycle is calculated according to the equation:</p> $\text{width} = 1/F * (I_{in} / I_{max})$ <p>or</p> $\text{duty} = 100\% * (I_{in} / I_{max});$ <p>If the width is equal or less than zero, a short On pulse with the width equal to the minimum calculation step at that moment will be generated. If the width is equal or greater than period of frequency F, a short Off pulse at the end of the period will be generated. As a result, the frequency of the output signal is always F.</p>			

V – Voltmeter

Symbol	Models	Signals
	Voltmeter	

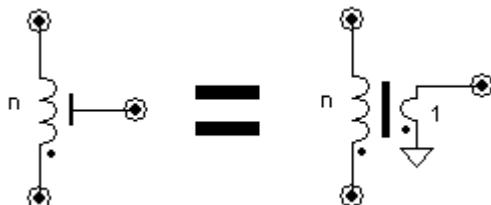
Model	Parameter	Units	Description
Voltmeter	No parameters.		
Open circuit.			

W – Winding

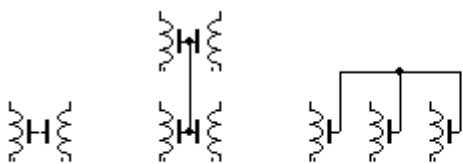
Symbol	Models	Signals
	Winding	

Model	Parameter	Units	Description
Winding	n	turns	Number of turns.

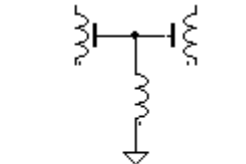
The Winding is actually an ideal transformer, with 1-turn second winding, one end of which is grounded, and another end is shown as a “core” pin of the winding:



To make an ideal transformer, connect cores of two or more windings by wire. Core magnetizing can be modeled by setting linear or non-linear inductor from core to ground:

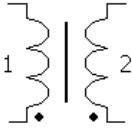
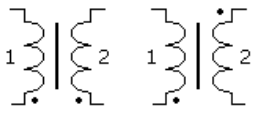


Ideal transformers



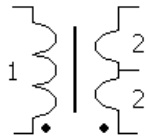
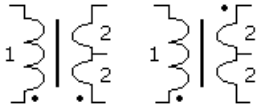
Transformer with magnetizing inductor

W – Transformer

Symbol		Models	Signals
		Transformer SubCir	
Views			


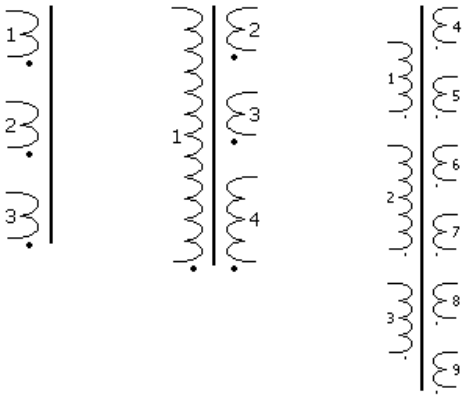
Model	Parameter	Units	Description
Transformer	n1	turns	Number of turns in the first winding.
	n2	turns	Number of turns in the second winding.
Ideal transformer with 2 windings. Coupling coefficient = 1.			

W – Differential transformer

Symbol	Models	Signals
	Transformer SubCir	
Views 		

Model	Parameter	Units	Description
Transformer	n1	turns	Number of turns in the first winding.
	n2	turns	Number of turns in second and third windings.
Ideal differential transformer with 3 windings. Coupling coefficient = 1. Second and third windings have the same number of turns n2 , and connected to form a differential transformer.			

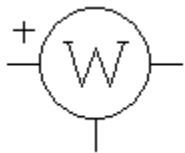
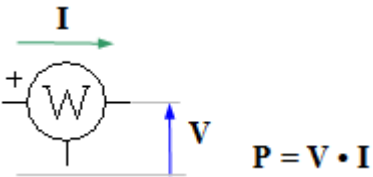

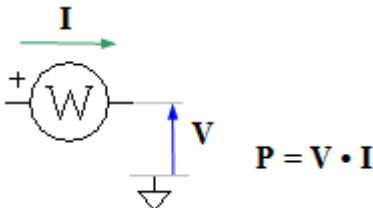
W – Custom transformer

Symbol	Models	Signals
	Transformer SubCir	
<p>This is a customized component. A component can be edited in the Edit Component dialog box. See <i>Editing customized component</i> chapter for instructions on editing a component.</p> <p>This component may have:</p> <ul style="list-style-type: none">- height from 2 to 32,- up to 32 windings (total),- arbitrary length of a winding. <p>Examples of Custom transformer component:</p> <div></div>		

Model	Parameter	Units	Description
Transformer	n1	turns	Number of turns in the first winding.

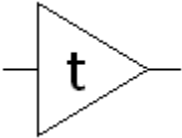
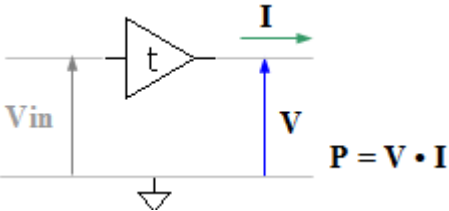
	nN	turns	Number of turns in the N th winding.
Ideal transformer with N windings. Coupling coefficient = 1.			

W – Wattmeter

Symbol	Models	Signals
	Wattmeter	 $P = V \cdot I$
	Wattmeter	 $P = V \cdot I$

Model	Parameter	Units	Description
Wattmeter	No parameters.		
Short circuit between current ports, open circuit between voltage ports. Can be used to measure power in grounded or non-grounded load.			

X – Delay

Symbol	Models	Signals
	Delay SubCir	

Model	Parameter	Units	Description
Delay	t0	s	Delay.
	IC	V	Initial condition: output voltage.

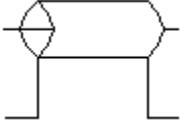
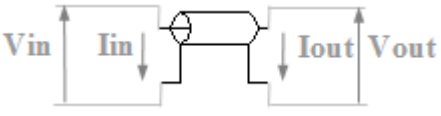
Output voltage is equal to input voltage, delayed by delay time **t0**:

$$V(t) = Vin(t - t0), \text{ where } t \text{ is current time.}$$

When calculating DC operating point, output is set to specified output voltage **IC**, or to input voltage, if **IC** is blank. Then output voltage is not changing until delay time **t0**.

The model allocates memory for storing delayed data only when needed and frees it immediately when possible.

X – Transmission line

Symbol	Models	Signals
	Line Lossy	

Model	Parameter	Units	Description
Line	t0	s	Delay.
	z0	Ohm	Characteristic impedance.
	VIC	V	Initial condition: voltage.
	IIC	A	Initial condition: current.

Lossless transmission line. The voltage and current in the line are represented as a superposition of forward and reflected waves, with V/I ratio in each wave equal to line characteristic impedance **z0**. V and I values of each wave are calculated based on boundary (input and output) conditions. The line functionality can also be described by the following equations:

$$\begin{aligned} Vin(t) &= z0 * (lin(t) - Iout(t - t0)) \\ Vout(t) &= z0 * (Iout(t) - lin(t - t0)) \end{aligned}$$

where t is current time.

Input and output are galvanically isolated: no current is flowing between input and output, and any voltage difference between input and output may exist.

When calculating DC operating point, initial forward and reflected voltage and current are calculated based on the following conditions:

if **VIC** and **IIC** are blank : $Vin = Vout, lin = -Iout$.
if **VIC** is specified and **IIC** is blank . . : $Vin = Vout = VIC$.
if **VIC** is blank and **IIC** is specified . . : $lin = IIC, Iout = -IIC$.
if **VIC** and **IIC** are specified : $Vin = Vout = VIC, lin = IIC, Iout = -IIC$.

The model allocates memory for storing forward and reflected wave data only when needed, and frees it immediately when possible.

If real line characteristics are given in line capacitance and inductance per length, the following equations can be used to derive **t0** and **z0** parameters:

$$\begin{aligned} t0 &= \text{sqrt}(L * C) * D \\ z0 &= \text{sqrt}(L / C) \end{aligned}$$

where:

C – line capacitance per length, F/m
L – line inductance per length, H/m
D – line length, m

Model	Parameter	Units	Description
Lossy	t0	s	Delay.
	z0	Ohm	Characteristic impedance.
	R	Ohm/ns	Series resistance per ns.
	fr	MHz	Skin losses cutoff (3 dB) frequency.
	G	1/Ohm/ns	Shunt conductance per ns.
	fG	MHz	Dielectric losses cutoff (3 dB) frequency.
	VIC	V	Initial condition: voltage.
	IIC	A	Initial condition: current.

Lossy transmission line. Lossy line modeling is similar to lossless transmission line, with addition of losses due to series resistance, skin effect, shunt conductance, and dielectric losses.

Constant series resistance is defined by **r** parameter. Skin losses are modeled by a number of RL chains, providing series impedance increase as a square root of frequency. The number of chains is automatically optimized based on calculation step value; however, the maximum impedance increase due to skin effect is limited to 40 dB (100 times). **fr** parameter defines a frequency where effective series impedance is approximately 3 dB higher than **r**. Skin losses are calculated only if **r** > 0, and **fr** is not infinite.

Constant shunt conductance is defined by **G** parameter. Dielectric losses are modeled by a shunt capacitance, providing shunt admittance increase proportional to frequency. **fG** parameter defines a frequency where effective shunt admittance is approximately 3 dB higher than **G**. Dielectric losses are calculated only if **G** > 0, and **fG** is not infinite.

Input and output are galvanically isolated: no current is flowing between input and output, and any voltage difference between input and output may exist.

When calculating DC operating point initial forward and reflected voltage and current are calculated based on the following conditions:

if **VIC** and **IIC** are blank : $V_{in} = V_{out}$, $I_{in} = -I_{out}$.

if **VIC** is specified and **IIC** is blank . . : $V_{in} = V_{out} = \mathbf{VIC}$.

if **VIC** is blank and **IIC** is specified . . : $I_{in} = \mathbf{IIC}$, $I_{out} = -\mathbf{IIC}$.

if **VIC** and **IIC** are specified : $V_{in} = V_{out} = \mathbf{VIC}$, $I_{in} = \mathbf{IIC}$, $I_{out} = -\mathbf{IIC}$.

The model allocates all the required memory immediately at transient start. The amount of memory is proportional to line delay and inverse proportional to calculation step.

If real line characteristics are given in line capacitance and inductance per length, the following equations can be used to derive **t0** and **z0** parameters:

$$\mathbf{t0} = \sqrt{L * C} * D$$

$$\mathbf{z0} = \sqrt{L / C}$$

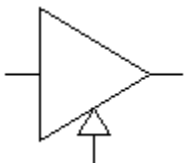
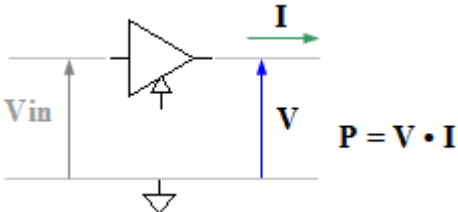
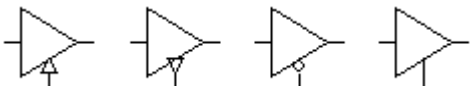
where:

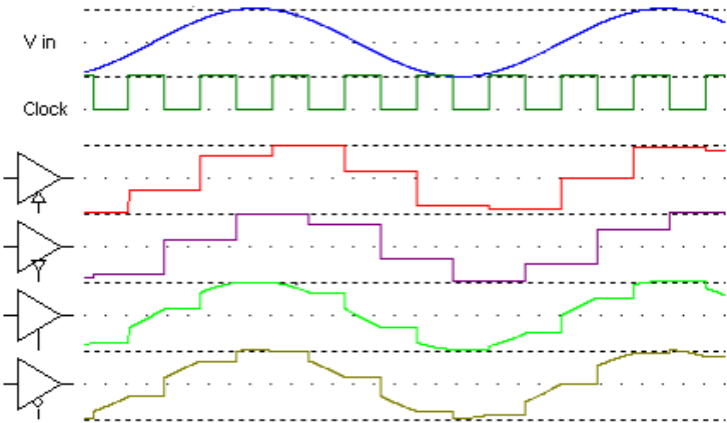
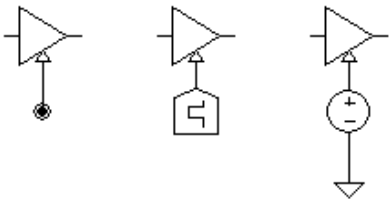
C – line capacitance per length, F/m

L – line inductance per length, H/m

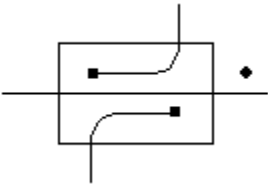
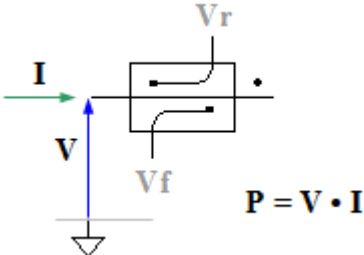
D – line length, m

X – Sample/hold

Symbol	Models	Signals
	SH SubCir	
Views		

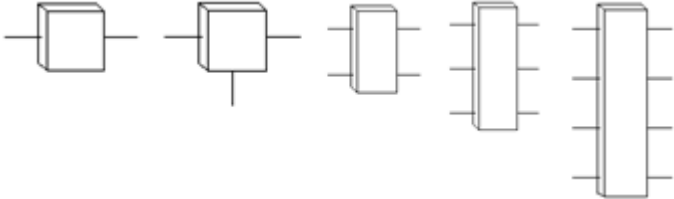
Model	Parameter	Units	Description
SH	IC	V	Initial condition: output voltage.
<p>Depending on view, the model is functioning as a sample/hold, or as a track/hold. In sample/hold mode, input voltage is sampled at rising (or falling) edge of a logical clock signal. In track/hold mode, output voltage tracks input voltage while clock signal is above (or below, for inverted control pin) the logical threshold, and holds it while clock signal is below (or above, for inverted control pin) the logical threshold.</p> <p>A waveforms example for different modes:</p>  <p>When calculating DC operating point output is set to specified output voltage IC. When calculating Linearized AC, and the clock pin is connected to the Label, Voltage source, or Logic generator with Pulse or Clock model, the component operates as a linear buffer with a delay equal to the Period of the source:</p> 			

X – Directional coupler

Symbol	Models	Signals
	Coupler	

Model	Parameter	Units	Description
Coupler	z0	Ohm	Characteristic impedance
	CF	dB	Coupling factor
<p>Directional coupler is a short circuit (no insertion loss) with two output ports: forward Vf, and reflected Vr. Output ports are voltage sources with zero output impedance and coupling factor CF. The output voltages are calculated as follows:</p> $V_f = K * (V + I * z_0) / 2$ $V_r = K * (V - I * z_0) / 2$ <p>where $K = 10^{-CF/20}$.</p> <p>All voltages are referenced to ground.</p>			

X – Block-2...Block-8

Symbol	Models
	SubCir
Blocks with predefined size and number of pins, to be used as subcircuits (SubCir model).	

X – Custom block

Symbol	Models	Signals
	SubCir	
<p>This is a customized component. A component can be edited in the Edit Component dialog box. See <i>Editing customized component</i> chapter for instructions on editing a component.</p> <p>This component may have:</p> <ul style="list-style-type: none">- arbitrary size up to 32(width) X 32(height),- up to 32 pins on each side <p>Examples of Custom block component:</p>		

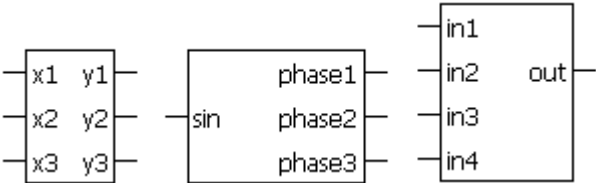
X – NL5 circuit

Symbol	Models	Signals
	SubCir	

This is a customized component. A component can be edited in the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

- This component may have:
- arbitrary size up to 32(width) X 256(height),
 - up to 256 inputs on the left side,
 - up to 256 outputs on the right side,

Examples of NL5 circuit component:



Model	Parameter	Units	Description
SubCir	File		File name of subcircuit schematic.
	Cmd		Subcircuit start-up command string
	IC		Subcircuit Initial conditions string

This model is similar to a standard **SubCir** model, with one difference: instead of parameters, subcircuit names are entered in the **Edit Component** dialog as a component pin names. See *Working with Subcircuits* chapter for details on subcircuit operation.

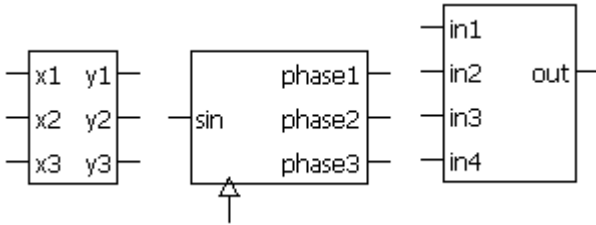
X – C-code

Symbol	Models	Signals
	C File	

This is a customized component. A component can be edited in the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

- This component may have:
- arbitrary size up to 32(width) X 256(height),
 - up to 256 inputs on the left side,
 - up to 256 outputs on the right side,
 - one or no clock pins on the bottom side.
 - custom or default input and output names.

Examples of C-code component:



Model	Parameter	Units	Description
C	Code		C-code.
	IC		Initial conditions.

C-code block. The model contains code written in simplified C language.

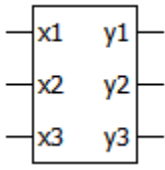
Code contains global variables declaration, initialization code, and main code.

IC may contain assignments of initial values to global variables. If not empty, **IC** will be executed after initialization code.

See *Working with C-code* chapter for details of the model functionality and instructions on creating the code.

Model	Parameter	Units	Description
File	File		C-code file name
	IC		Initial conditions.
<p>The model executes C-code from the text file. If File parameter does not have a full path, NL5 will search for the file in the directory where schematic file is located, then in the Library directories, and then in the Library directories relative to schematic file directory (see NL5 Manual, <i>Schematic Properties</i>).</p> <p>IC may contain assignments of initial values to global variables. If not empty, IC will be executed after initialization code.</p> <p>See <i>Working with C-code</i> chapter for details of the model functionality and instructions on creating the code.</p>			

X – DLL

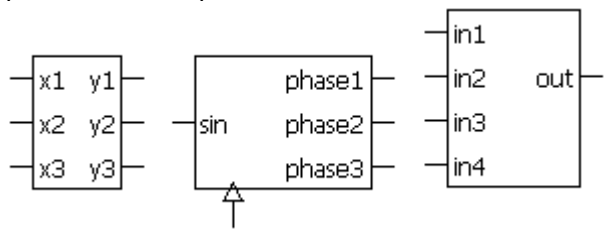
Symbol	Models	Signals
	DLL	

This is a customized component. A component can be edited in the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

This component may have:

- arbitrary size up to 32(width) X 256(height),
- up to 256 inputs on the left side,
- up to 256 outputs on the right side,
- one or no clock pins on the bottom side.
- custom or default input and output names.

Examples of DLL component:



Model	Parameter	Units	Description
DLL	DLL		DLL file name
	Init		Initialization function name.
	Main		Main function name.
	Pause		Pause function name.
	Exit		Exit function name.
	IC		Initial conditions.

DLL block. Component's code is written in C, compiled, and placed in the **64-bit** DLL file. DLL functions will be called by NL5 during transient simulation.

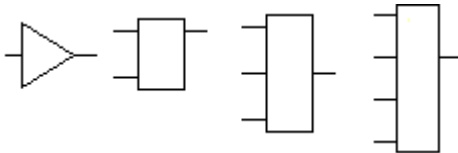
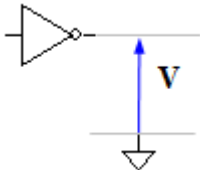
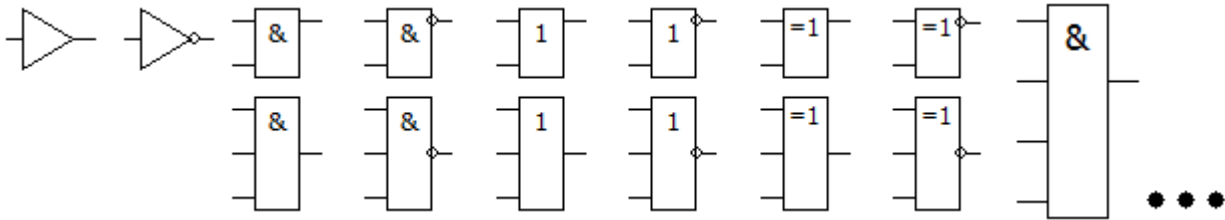
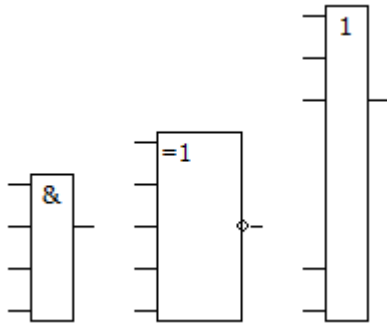
DLL parameter is a DLL file name. If **DLL** parameter does not have a full path, NL5 will search for the file in the directory where schematic file is located, then in the Library directories, and then in the Library directories relative to schematic file directory (see NL5 Manual, *Schematic Properties*).

Init - initialization function. Called once at the beginning of transient simulation at t=0. Leave it blank if not used.
Main - main function. Called on every simulation step, or in rising edge of the clock, if clock pin exists.
Pause - called when transient simulation is paused. Leave it blank if not used.
Exit - called when DLL is being closed, and DLL component destroyed. Leave it blank if not used.

IC may contain assignments of initial values to outputs and component variables. If not empty, **IC** will be executed after initialization code.

See *Working with DLL* chapter for details of the model functionality and instructions on creating code and DLL.

Y – Gates

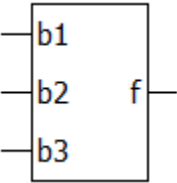
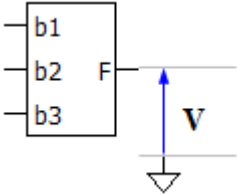
Symbol				Models		Signals		
				Logic Delay Delay2				
Views								
<p>For all gates, select view for logical function (AND, OR, XOR), and output pin inversion.</p> <p>XOR (=1) function is odd function (modulo-2 sum): the output is high if odd number of inputs are high.</p> <p>When calculating DC operating point, output is set to specified IC level.</p> <p>Custom gate component can be edited in the Edit Component dialog box. See <i>Editing customized component</i> chapter for instructions on editing a component.</p> <p>This component may have:</p> <ul style="list-style-type: none">- arbitrary size up to 32(width) X 32(height),- up to 32 inputs on the left side,- one output on the right side. <p>Examples of Custom gate component:</p> 								

Model	Parameter	Units	Description
Logic	IC		Initial condition: Low/High.
Output signal is delayed by one calculation step.			

Model	Parameter	Units	Description
Delay	Delay	s	Output delay.
	IC		Initial condition: Low/High.
Output signal is delayed by specified Delay time. Short signals (shorter than Delay) may not pass through and will not affect output.			

Model	Parameter	Units	Description
Delay2	Up	s	Output delay of rising edge.
	Down	s	Output delay of falling edge.
	IC		Initial condition: Low/High.
Output signal is delayed by Up time for rising edge of output signal, and Down time for falling edge of output signal. Short signals (comparable or shorter than Up or Down) may not pass through and will not affect output.			

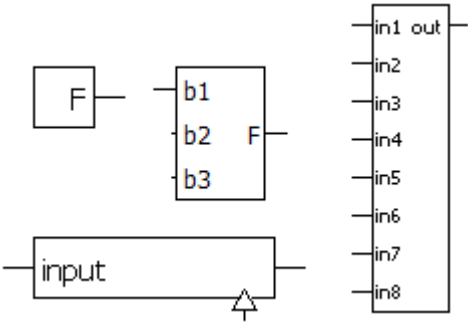
Y – Logical function

Symbol	Models	Signals
	Function	

This is a customized component. A component can be edited in the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

- This component may have:
- arbitrary size up to 32(width) X 8(height),
 - up to 8 inputs on the left side,
 - one output on the right side,
 - one or no clock pins on the bottom side.
 - custom input and output names.

Examples of Logical function component:



Model	Parameter	Units	Description
Function	f	V	Output as function of the logical inputs.
	IC	V	Initial condition: Low/High.

Arbitrary logical function **f** defines output logical state as a function of the following variables:

pin_name – logical value of the input voltage on the input pin **pin_name**.
t - current time

First, input voltages are converted to logical values, then the function is calculated. A logical result of the function is converted to the output voltage, which may have only Low or High logical level.

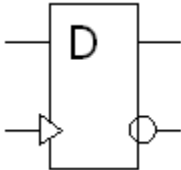
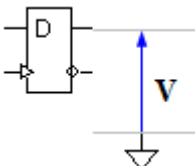
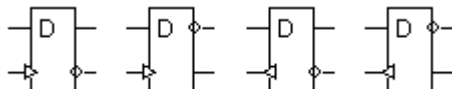
Example:

```
f = (b1 && b2) || b2
f = selector ? in1 : in2
```

If **clock** pin does not exist, the model operates in **continuous** mode: the function is calculated and applied to the output on every calculation step. If **clock** pin exists, the model operates in **synchronized** mode: the function is calculated and applied to the output only on rising (or falling) edge of logical clock signal. As a result, this mode may provide faster simulation than **continuous** mode.

When calculating DC operating point, and in AC analysis, output is set to specified output voltage **IC**. Output voltage is always delayed by one calculation step.

Y – D flip-flop

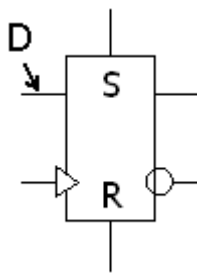
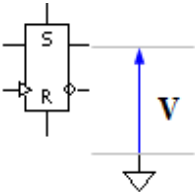
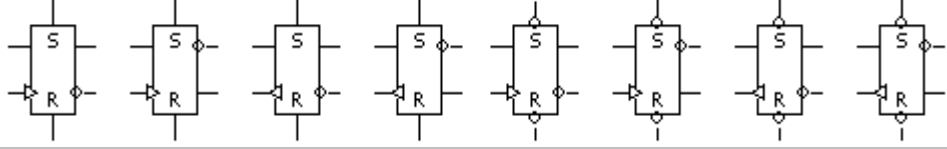
Symbol		Models	Traces
		Logic Delay Delay2	
Views			
For all models, when calculating DC operating point, output is set to specified IC level.			

Model	Parameter	Units	Description
Logic	IC		Initial condition: Low/High.
Output signal is delayed by one calculation step.			

Model	Parameter	Units	Description
Delay	Delay	s	Output delay.
	IC		Initial condition: Low/High.
Output signal is delayed by specified Delay time. Short signals (shorter than Delay) may not pass through and will not affect output.			

Model	Parameter	Units	Description
Delay2	Up	s	Output delay of rising edge.
	Down	s	Output delay of falling edge.
	IC		Initial condition: Low/High.
Output signal is delayed by Up time for rising edge of output signal, and Down time for falling edge of output signal. Short signals (comparable or shorter than Up or Down) may not pass through and will not affect output.			

Y – SR trigger

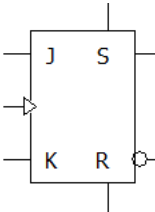
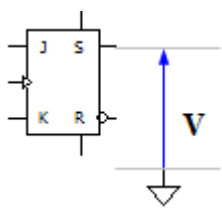
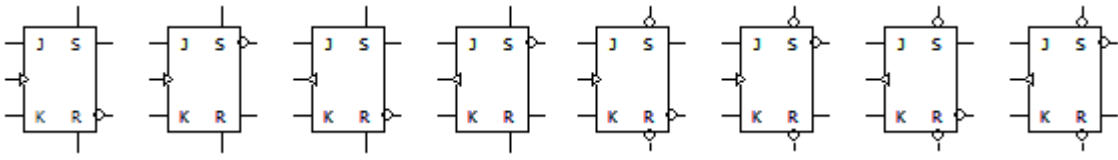
Symbol		Models	Traces
		Logic Delay Delay2	
Views			
<p>For all models, For all models, when calculating DC operating point, output is set to specified IC level.</p> <p>Dominance defines trigger output states in case both Set and Reset inputs are active.</p> <p>Please note that 'D' input is not shown on the symbol due to limited space.</p>			

Model	Parameter	Units	Description
Logic	Dominance		Dominance: None/Set/Reset
	IC		Initial condition: Low/High.
Output signal is delayed by one calculation step.			

Model	Parameter	Units	Description
Delay	Delay	s	Output delay.
	Dominance		Dominance: None/Set/Reset
	IC		Initial condition: Low/High.
Output signal is delayed by specified Delay time. Short signals (shorter than Delay) may not pass through and will not affect output.			

Model	Parameter	Units	Description
Delay2	Up	s	Output delay of rising edge.
	Down	s	Output delay of falling edge.
	Dominance		Dominance: None/Set/Reset
	IC		Initial condition: Low/High.
Output signal is delayed by Up time for rising edge of output signal, and Down time for falling edge of output signal. Short signals (comparable or shorter than Up or Down) may not pass through and will not affect output.			

Y – JK trigger

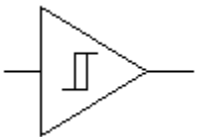
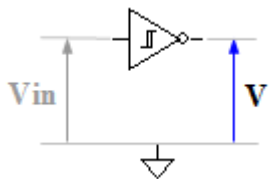
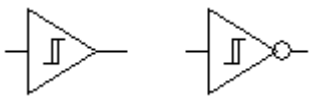
Symbol		Models	Traces
		Logic Delay Delay2	
Views			
For all models, For all models, when calculating DC operating point, output is set to specified IC level. Dominance defines trigger output states in case both Set and Reset inputs are active.			

Model	Parameter	Units	Description
Logic	Dominance		Dominance: None/Set/Reset
	IC		Initial condition: Low/High.
Output signal is delayed by one calculation step.			

Model	Parameter	Units	Description
Delay	Delay	s	Output delay.
	Dominance		Dominance: None/Set/Reset
	IC		Initial condition: Low/High.
Output signal is delayed by specified Delay time. Short signals (shorter than Delay) may not pass through and will not affect output.			

Model	Parameter	Units	Description
Delay2	Up	s	Output delay of rising edge.
	Down	s	Output delay of falling edge.
	Dominance		Dominance: None/Set/Reset
	IC		Initial condition: Low/High.
Output signal is delayed by Up time for rising edge of output signal, and Down time for falling edge of output signal. Short signals (comparable or shorter than Up or Down) may not pass through and will not affect output.			

Y – Schmitt trigger

Symbol	Models	Traces
	Logic Delay Delay2	
Views		

For all models, when calculating DC operating point, output is set to specified **IC** level.

Output is set to Low or High level as follows (non-inverted output, **Threshold** is a logical level threshold):

$V_{in} > \text{Threshold} + \text{Hysteresis}/2 \dots : V = \text{High}$
 $V_{in} < \text{Threshold} - \text{Hysteresis}/2 \dots : V = \text{Low}$
 Otherwise : $V = \text{previous state}$

Model	Parameter	Units	Description
Logic	Hysteresis	V	Hysteresis.
	IC		Initial condition: Low/High.

Output signal is delayed by one calculation step.

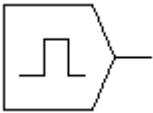
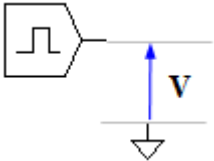
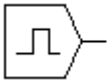

Model	Parameter	Units	Description
Delay	Hysteresis	V	Hysteresis.
	Delay	s	Output delay
	IC		Initial condition: Low/High.

Output signal is delayed by specified **Delay** time.
Short signals (shorter than **Delay**) may not pass through and will not affect output.

Model	Parameter	Units	Description
Delay2	Hysteresis	V	Hysteresis.
	Up	s	Output delay of rising edge.
	Down	s	Output delay of falling edge.
	IC		Initial condition: Low/High.

Output signal is delayed by **Up** time for rising edge of output signal, and **Down** time for falling edge of output signal. Short signals (comparable or shorter than **Up** or **Down**) may not pass through and will not affect output.

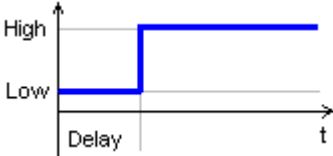
Y – Logic generator

Symbol	Models		Traces
	Low High Step Single	Pulse Clock List File	
Views	 		

Model	Parameter	Units	Description
Low	No parameters.		
Output is always Low , regardless of output inversion state.			

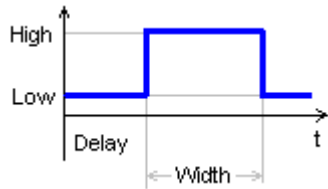
Model	Parameter	Units	Description
High	No parameters.		
Output is always High , regardless of output inversion state			

Model	Parameter	Units	Description
Step	Delay	s	Delay before active state.
Output goes to active state after Delay time. The following signal will be generated for non-inverted output:			



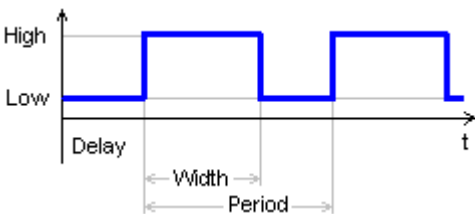
Model	Parameter	Units	Description
Single	Width	s	Pulse width.
	Delay	s	Delay before first pulse starts.

The pulse starts after **Delay** time. The following signal will be generated for non-inverted output:



Model	Parameter	Units	Description
Pulse	Period	s	Period.
	Width	s	Pulse width.
	Delay	s	Delay before first pulse starts.

Pulses start after **Delay** time. The following signal will be generated for non-inverted output:

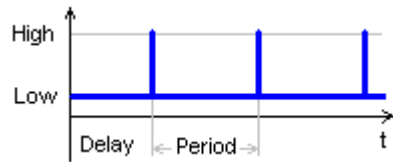


Model	Parameter	Units	Description
Clock	Period	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.

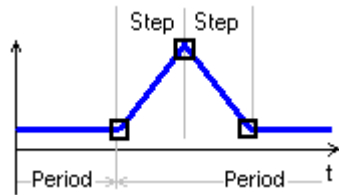
Pulses start after **Delay** time. Output goes to active state for one simulation step only.

Clock model is recommended to produce a constant frequency clock signal for C-code, DLL, logical components, etc. Unlike **Pulse** model, it won't force unnecessary step reduction at the end of the pulse, which may help to accelerate simulation.

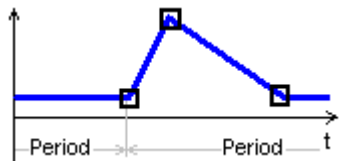
The following signal will be generated for non-inverted output:



If **Step** parameter is not zero **and** is less than current schematic simulation step, the step is adjusted to provide rise and fall of clock pulse to be equal to **Step**:



Otherwise, the clock pulse is created using current schematic simulation step, which depends on many factors, and cannot be easily predicted:



Model	Parameter	Units	Description
List	List		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Output sequence is defined in the **List** parameter in the **csv** (comma-separated values) format, as follows:

`t0,s0,t1,s1,...,tn,sn`

s0...sn defines output logical level: positive number corresponds to High, zero or negative number to Low.

If $t < t_0$, output level is s0.

At t_0 output level is s0, at t_1 output level is s1, and so on.

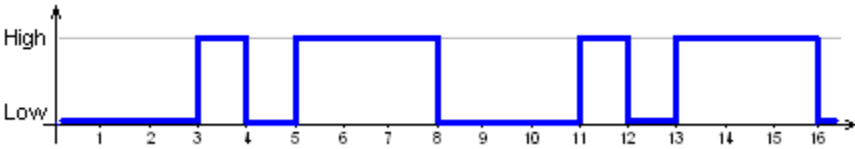
If $t > t_n$, and **Cycle** parameter is set to **No**, output level remains at sn. Otherwise the sequence is repeated continuously.

Sequence start is delayed by **Delay** time.

Example:

`List = 0,0,3,1,4,0,5,1,8,0`

If **Cycle = Yes**, **Delay = 0**, the following logical output will be generated:



See *Working with List model* chapter for more details.

Model	Parameter	Units	Description
File	File		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.

Output sequence defined in the text file. If **File** parameter does not have a full path, NL5 will search for the file in the directory where schematic file is located, then in the Library directories, and then in the Library directories relative to schematic file directory (see NL5 Manual, *Schematic Properties*).

Logical output sequence is defined in the **csv** (comma-separated values) format, as follows:

```
<if first line does not start with a number, it is ignored>
t0,s0
t1,s1
.....
tn,sn
```

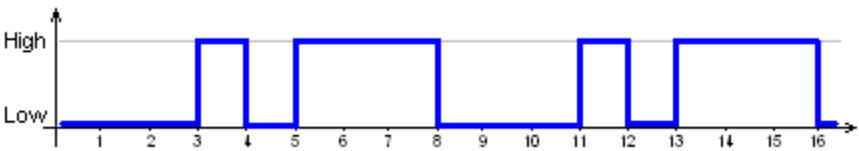
s0...sn defines output logical level: positive number corresponds to High, zero or negative number to Low.
If $t < t_0$, output level is s0.
At t_0 output level is s0, at t_1 output level is s1, and so on.
If $t > t_n$, and **Cycle** parameter is set to **No**, output level remains at sn. Otherwise the sequence is repeated continuously.

Sequence start is delayed by **Delay** time.

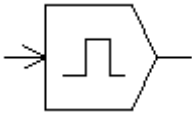
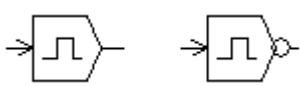
Example:

```
0,0
3,1
4,0
5,1
8,0
```

If **Cycle = Yes**, **Delay = 0**, the following sequence will be generated:



Y – Logic controlled logic generator

Symbol	Models	Signals
	Gate Low High One-shot Step	Single Pulse Clock List File
Views		

Model	Parameter	Units	Description
Gate	IC		Initial condition: Low/High.
Component operates similar to Logic model of Gate component.			
When calculating DC operating point, output is set to the state defined in IC .			

Model	Parameter	Units	Description
Low	No parameters.		
Output is always Low , regardless of output inversion state.			

Model	Parameter	Units	Description
High	No parameters.		
Output is always High , regardless of output inversion state			

Model	Parameter	Units	Description
One-shot	Width	s	Pulse width.
One-shot pulse generator. When increasing input voltage V_{in} crosses logical threshold, logical pulse of Width duration is generated.			
If increasing V_{in} crosses logical threshold value while pulse is being generated, the pulse is restarted.			

Model	Parameter	Units	Description
Step	Delay	s	Delay before active state.
<p>When control signal <i>Vin</i> is below logical threshold, output is in Off state. When increasing control signal <i>Vin</i> crosses logical threshold, a signal similar to Step model of Logic generator component is generated. When decreasing control signal <i>Vin</i> drops below logical threshold, output goes to Off state immediately.</p>			

Model	Parameter	Units	Description
Single	Width	s	Pulse width.
	Delay	s	Delay before first pulse starts.
<p>When control signal <i>Vin</i> is below logical threshold, output is in Off state. When increasing control signal <i>Vin</i> crosses logical threshold, a signal similar to Single model of Logic generator component is generated. When decreasing control signal <i>Vin</i> drops below logical threshold, output goes to Off state immediately.</p>			

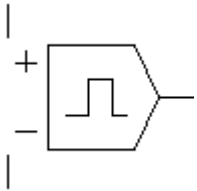
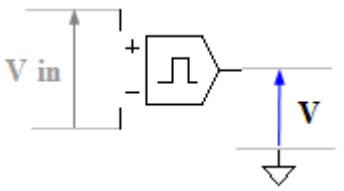
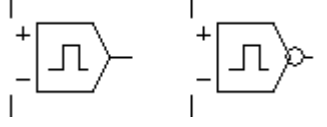
Model	Parameter	Units	Description
Pulse	Period	s	Period.
	Width	s	Pulse width.
	Delay	s	Delay before first pulse starts.
<p>When control signal <i>Vin</i> is below logical threshold, output is in Off state. When increasing control signal <i>Vin</i> crosses logical threshold, a signal similar to Pulse model of Logic generator component is generated. When decreasing control signal <i>Vin</i> drops below logical threshold, output goes to Off state immediately.</p>			

Model	Parameter	Units	Description
Clock	Period	s	Period.
	Step	s	Simulation step of rise and fall.
	Delay	s	Delay before first pulse starts.
<p>When control signal <i>Vin</i> is below logical threshold, output is in Off state. When increasing control signal <i>Vin</i> crosses logical threshold, a signal similar to Clock model of Logic generator component is generated. When decreasing control signal <i>Vin</i> drops below logical threshold, output goes to Off state immediately.</p>			

Model	Parameter	Units	Description
List	List		Comma-separated string.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.
<p>When control signal <i>Vin</i> is below logical threshold, output is equal to s0 value of List signal. When increasing control signal <i>Vin</i> crosses logical threshold, a signal similar to List model of Logic generator component is generated. This moment is also considered as $t=0$ for the List signal. When decreasing control signal <i>Vin</i> drops below logical threshold, output goes to s0 immediately.</p>			

Model	Parameter	Units	Description
File	File		File name.
	Cycle		Cycling (repeat): No/Yes.
	Delay	s	Delay.
<p>When control signal V_{in} is below logical threshold, output is equal to s0 value specified in the File. When increasing control signal V_{in} crosses logical threshold, a signal similar to File model of Logic generator component is generated. This moment is also considered as $t=0$ for the File signal. When decreasing control signal V_{in} drops below logical threshold, output goes to s0 immediately.</p>			

Y – Voltage controlled logic generator

Symbol	Models	Signals
	Gate Low High One-shot	
Views		

Model	Parameter	Units	Description
Gate	Threshold	V	Voltage threshold.
	Hysteresis	V	Hysteresis.
	IC		Initial condition: Low/High.

Gate with hysteresis. Output is set to Low or High level as follows (non-inverted output):

$V_{in} > \text{Threshold} + \text{Hysteresis}/2 \dots : V = \text{High}$
 $V_{in} < \text{Threshold} - \text{Hysteresis}/2 \dots : V = \text{Low}$
 Otherwise $\dots : V = \text{previous state}$

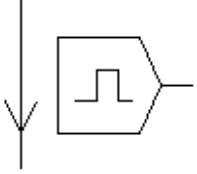
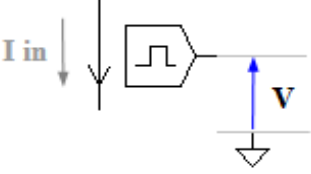
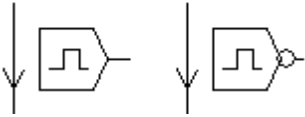
When calculating DC operating point, output is set to the state defined in **IC**.

Model	Parameter	Units	Description
Low	No parameters.		
Output is always Low , regardless of output inversion state.			

Model	Parameter	Units	Description
High	No parameters.		
Output is always High , regardless of output inversion state			

Model	Parameter	Units	Description
One-shot	Threshold	V	Voltage threshold.
	Width	s	Pulse width.
<p>One-shot pulse generator. When increasing input voltage V_{in} crosses Threshold, logical pulse of Width duration is generated.</p> <p>If increasing V_{in} crosses Threshold value while pulse is being generated, the pulse is restarted.</p>			

Y – Current controlled logic generator

Symbol	Models	Signals
	Gate Low High One-shot	
Views		

Model	Parameter	Units	Description
Gate	Threshold	A	Current threshold.
	Hysteresis	A	Hysteresis.
	IC		Initial condition: Low/High.

Gate with hysteresis. Output is set to Low or High level as follows (non-inverted output):

$I_{in} > \text{Threshold} + \text{Hysteresis}/2 \dots : V = \text{High}$
 $I_{in} < \text{Threshold} - \text{Hysteresis}/2 \dots : V = \text{Low}$
 Otherwise : $V = \text{previous state}$

When calculating DC operating point, output is set to the state defined in **IC**.

Model	Parameter	Units	Description
Low	No parameters.		
Output is always Low , regardless of output inversion state.			

Model	Parameter	Units	Description
High	No parameters.		
Output is always High , regardless of output inversion state			

Model	Parameter	Units	Description
One-shot	Threshold	A	Current threshold.
	Width	s	Pulse width.
<p>One-shot pulse generator. When increasing input current I_{in} crosses Threshold, logical pulse of Width duration is generated.</p> <p>If increasing I_{in} crosses Threshold value while pulse is being generated, the pulse is restarted.</p>			

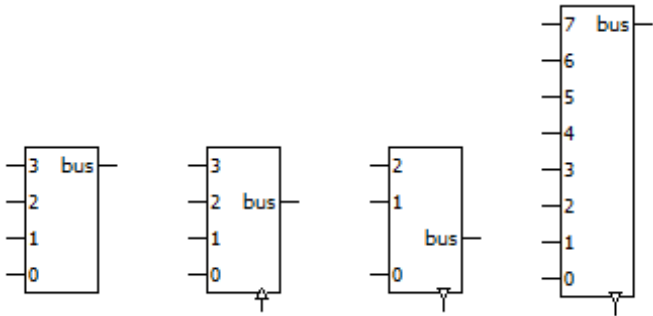
Y – Bus

Symbol	Models	Traces
	Bus	

This is a customized component. A component can be edited in the **Edit Component** dialog box. See *Editing customized component* chapter for instructions on editing a component.

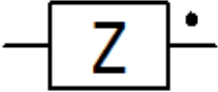
- This component may have:
- arbitrary size up to 32(width) X 32(height),
 - up to 32 inputs on the left side,
 - one output on the right side,
 - one or no clock pins on the bottom side.

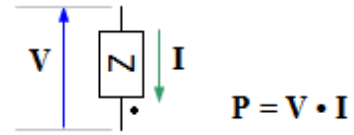
Examples of **Bus** component:



Model	Parameter	Units	Description
Bus	Format		Bus format: Signed/Unsigned
	IC	V	Initial condition: output voltage.
<p>Digital-to-bus converter. Converts logical inputs to a bus signal, considering logical inputs as bits of binary number. Input 0 is LSB. If Format is set to Signed, an input number is considered to be in two's complement format.</p> <p>When calculating DC operating point, and in AC analysis, output is set to specified output voltage IC. Output voltage is always delayed by one calculation step.</p>			

Z – Impedance

Symbol	Models	Signals
	Function Poly1 Poly2 Poly3 Poly4	Poly5 Roots Table File



Model	Parameter	Units	Description
Function	f	Ohm	Impedance.

Arbitrary function **f** defines impedance in **s** domain. The following variables can represent frequency in the function:

- f** – current AC frequency, Hz
- w** – angular AC frequency, $w = 2\pi f$.
- s** or **p** – Laplace parameter, $s = p = j*2\pi f$.

Example:

```
f = 1/(1+s)
f = exp(-R1*C1**s)
```

Only operators and functions that support complex numbers can be used in this function. If **f** is blank, it is assumed to be zero.

At transient and DC operation point calculation for AC (if enabled), the impedance is equal to **f(0)**.

Model	Parameter	Units	Description
Poly1	b0		Numerator polynomial coefficients 0.
Poly2
Poly3	a0		Denominator polynomial coefficients 0.
Poly4
Poly5	IC		Initial condition.

Impedance is a ratio of polynomials of Laplace parameter **s**:

$$f(s) = (b0 + b1*s + b2*s^2 + \dots) / (a0 + a1*s + a2*s^2 + \dots)$$

These models support transient as well.

Initial condition **IC** consists of internal model values. It should not be manually edited, except clearing it to blank (no **IC**).

At DC operation point calculation, **f(0)** is used.

Model	Parameter	Units	Description
Roots	K		Gain.
	Roots		Roots (zeroes and poles).
	IC	V	Initial condition..

Impedance is defines by zeroes and poles:

$$f(s) = K * (s-z1)*(s-z2).../(s-p1)/(s-p2)...$$

where **K** is gain, z1...zn are zeroes, p1...pN are poles.

Roots are defined by **Roots** parameter in the **csv** (comma-separated values) format, as follows:

$$Nz,Rez1,Imz1,...,Np,Rep1,Imp1,...$$

where:

Nz - number of zeroes
 Rezi – real part of zi
 Imzi – imaginary part of zi
 Np - number of poles,
 Repi – real part of pi
 Impi – imaginary part of pi

There could be any number of zeroes and poles, however the resulting numerator and denominator polynomials order should not exceed 5. See *Working with Roots model* chapter for details on entering/editing roots.

The model supports transient as well.

Initial condition **IC** consists of internal model values. It should not be manually edited, except clearing it to blank (no **IC**).

At DC operation point calculation, **f(0)** is used.

Model	Parameter	Units	Description
Table	Table		Comma-separated string.
	R	Ohm	Resistance used for transient simulation.

Look-up table. Impedance is defined by **Table** parameter in the **csv** (comma separated values) format, as follows:

$$f1,mag1,phase1,f2,mag2,phase2,...,fN,magN,phaseN$$

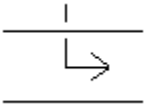
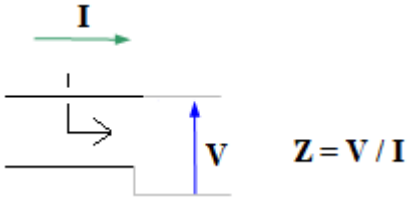
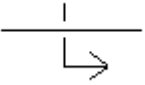
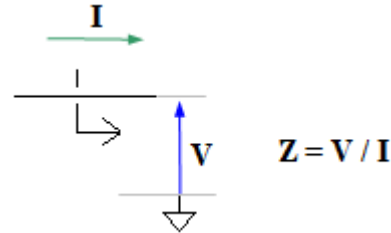
where mag and phase are impedance magnitude and phase at specified frequency. Output value between specified frequencies is interpolated using linear or logarithmic interpolation, depending on simulation scale. Below first and above last specified frequency, simulation result is not provided. Frequencies should be given in ascending order.

At transient simulation, constant resistance **R** is used.

See *Working with Table model* chapter of NL5 Manual for more details.

Model	Parameter	Units	Description
File	File		File name.
	R	Ohm	Resistance used for transient simulation.
<p>Impedance is defined in the text file in the following format.</p> <p>f1,mag1,phase1 f2,mag2,phase2 ...</p> <p>At transient simulation, constant resistance R is used</p>			

Z – Z-meter

Symbol	Models	Signals
	Z-meter	 $Z = V / I$
	Z-meter	 $Z = V / I$

Model	Parameter	Units	Description
Z-meter	No parameters.		
Impedance meter (floating or grounded load) for AC, Linearized method only. Short circuit for transient.			

Operators

Operators are listed in descending precedence order (1 - most, 14 - least).

The table is based on http://en.cppreference.com/w/cpp/language/operator_precedence

Precedence	Operator	Description
1	() [] x++ x-- ++x --x	Function call Array subscripting Postfix increment: x=x+1 after use Postfix decrement: x=x-1 after use Prefix increment: x=x+1 before use Prefix decrement: x=x-1 before use
2	+ - ! ~ (bool) (int) (int64) (float) (double) (complex)	Unary plus Unary minus Logical NOT Bitwise NOT Type cast to bool Type cast to int Type cast to int64 Type cast to float Type cast to double Type cast to complex
3	* / %	Multiplication Division Remainder
4	+ -	Addition Subtraction
5	<< >>	Bitwise left shift Bitwise right shift
6	< <= > >=	Relation operator "less than" Relation operator "less than or equal to" Relation operator "greater than" Relation operator "greater than or equal to"
7	== !=	Relation operator "equal to" Relation operator "not equal to"
8	&	Bitwise AND
9	^	Bitwise XOR (exclusive OR)
10		Bitwise OR
11	&&	Logical AND
12		Logical OR
13	?:	Ternary conditional operator
14	= += -= *= /= %= <<= >>= &= ^= =	Assignment Assignment by sum Assignment by difference Assignment by product Assignment by quotient Assignment by remainder Assignment by bitwise left shift Assignment by bitwise right shift Assignment by bitwise AND Assignment by bitwise XOR Assignment by bitwise OR

Functions

abs, mag

Prototype	<code>double abs(<i>complex</i>)</code> <code>double abs(<i>double</i>)</code> <code>int64 abs(<i>int64</i>)</code> <code>int abs(<i>int</i>)</code>
Description	<p>Absolute value (magnitude).</p> <p>For complex argument: $\text{abs} = \sqrt{re^2 + im^2}$.</p> <p><code>mag()</code> can be used instead of <code>abs()</code>.</p>
Examples	<pre>abs(3.0+4.0j) = 5.0 abs(-3j) = 3.0 abs(1.0) = 1.0 abs(-10) = 10</pre>

sign

Prototype	<code>int sign(<i>double</i>)</code>
Description	<p>Indicates whether a numeric value is positive, negative, or zero.</p> <p><code>sign(x)</code> returns:</p> <ul style="list-style-type: none"> • 0 if <code>x=0</code> • 1 if <code>x>0</code> • -1 if <code>x<0</code>
Examples	<pre>sign(1.234) = 1 sign(0) = 0 sign(-5) = -1</pre>

re, im

Prototype	<code>double re(<i>complex</i>)</code> <code>double im(<i>complex</i>)</code>
Description	Real and imaginary part of complex number.
Examples	<pre>re(1.2+3.4j) = 1.2 im(1.2+3.4j) = 3.4</pre>

phase

Prototype	<code>double phase(<i>complex</i>)</code>
Description	Phase of complex number. Returns phase in the range $-\text{Pi} \dots +\text{Pi}$.
Examples	<code>phase(1+1j) = 0.785398 (Pi/4)</code>

sqrt

Prototype	<code>complex sqrt(<i>complex</i>)</code> <code>double sqrt(<i>double</i>)</code>
Description	Square root. If argument is <code>double</code> , negative argument will cause error.
Examples	<code>sqrt(4.0) = 2</code> <code>sqrt(-4.0) : math error</code> <code>sqrt(2j) = 1+1j</code>

sqr

Prototype	<code>double sqr(<i>double</i>)</code>
Description	“Signed” square root. <code>sqr(x)</code> returns: <ul style="list-style-type: none"> \sqrt{x} if $x \geq 0$ $-\sqrt{-x}$ if $x < 0$
Examples	<code>sqr(4) = 2</code> <code>sqr(-4) = -2</code>

sq

Prototype	<code>complex sq(<i>complex</i>)</code> <code>double sq(<i>double</i>)</code>
Description	<code>sq(x)</code> calculates $x*x$: square of the argument.
Examples	<code>sq(2) = 4</code> <code>sq(1+1j) = 0+2j</code>

lim, limit

Prototype	<i>double</i> lim(<i>double</i> x, <i>double</i> min, <i>double</i> max)
Description	<p>Limiting function. lim(x, min, max) returns:</p> <ul style="list-style-type: none"> • x, if min≤x≤max • min, if x<min • max, if x>max <p>limit() can be used instead of lim().</p>
Examples	<pre>lim(0,-1,2) = 0 lim(-2,-1,2) = -1 lim(10,-1,2) = 2</pre>

islow, ishigh

Prototype	<i>bool</i> islow(<i>double</i>) <i>bool</i> ishigh(<i>double</i>)
Description	<p>Compares argument with logical threshold.</p> <p>islow(x) returns true if x is less than circuit logical threshold, otherwise false.</p> <p>ishigh(x) returns true if x is greater than circuit logical threshold, otherwise false.</p>
Examples	<pre>islow(1.0) = true ishigh(1.0) = false</pre>

sum

Prototype	<i>complex</i> sum(<i>complex</i> ,...) <i>complex</i> sum(<i>complex</i> []) <i>double</i> sum(<i>double</i> ,...) <i>double</i> sum(<i>double</i> [])
Description	<p>sum(x,...) returns sum of arguments. Number of arguments is not limited.</p> <p>If x is an array x[N], sum(x) returns sum of all array elements.</p>
Examples	<pre>sum(1.0,2.0,3.0) = 6.0 sum(1.0+1.0j,2.0+2.0j) = 3.0+3.0j double x[] = { 1.0, 2.0, 3.0, 4.0 }; sum(x) = 10.0;</pre>

mean, average

Prototype	<i>complex</i> mean(<i>complex</i> ,...) <i>complex</i> mean(<i>complex</i> []) <i>double</i> mean(<i>double</i> ,...) <i>double</i> mean(<i>double</i> [])
Description	mean(x,...) returns mean (average) value of arguments. Number of arguments is not limited. If x is an array x[N], sum(x) returns mean (average) value of all array elements. average() can be used instead of mean().
Examples	<pre> mean(1.0,2.0,3.0) = 2.0 mean(1.0+1.0j,2.0+2.0j) = 1.5+1.5j double x[] = { 1.0, 2.0, 3.0, 4.0 }; mean(x) = 2.5; </pre>

min

Prototype	<i>double</i> min(<i>double</i> ,...) <i>double</i> min(<i>double</i> []) <i>int64</i> min(<i>int64</i> ,...) <i>int64</i> min(<i>int64</i> []) <i>int</i> min(<i>int</i> ,...) <i>int</i> min(<i>int</i> []) <i>bool</i> min(<i>bool</i> ,...) <i>bool</i> min(<i>bool</i> [])
Description	min(x,...) returns smaller value of arguments. Number of arguments is not limited. If x is an array x[N], min(x) returns smaller value of all array elements.
Examples	<pre> min(1.0,2.0,3.0) = 1.0 min(1,2,3) = 1 min(false, true, true) = false double x[] = { -1.0, 2.0, -3.0, 4.0 }; min(x) = -3.0; </pre>

max

Prototype	<pre> double max(double,...) double max(double[]) int64 max(int64,...) int64 max(int64[]) int max(int,...) int max(int[]) bool max(bool,...) bool max(bool[]) </pre>
Description	<p>max(x,...) returns larger value of arguments. Number of arguments is not limited.</p> <p>If x is an array x[N], max(x) returns larger value of all array elements.</p>
Examples	<pre> max(1.0,2.0,3.0) = 3.0 max(1,2,3) = 3 max(false, true, true) = true double x[] = { -1.0, 2.0, -3.0, 4.0 }; max(x) = 4.0; </pre>

exp

Prototype	<pre> complex exp(complex) double exp(double) </pre>
Description	<p>exp(x) calculates the exponential e to the x.</p>
Examples	<pre> exp(3.0) = 20.0855 exp(PI*0.5j) = 0+1j </pre>

pow

Prototype	<pre> complex pow(complex x, double y) double pow(double x, double y) </pre>
Description	<p>pow(x,y) calculates x^y : x to the power of y.</p> <p>If double argument x is negative, math error may occur.</p>
Examples	<pre> pow(10.0,2.0) = 100.0 pow(1j,3) = 0-1j pow(-4.0,0.5) : math error pow(-4.0+0j,0.5) = 0+2j </pre>

pwr

Prototype	<i>double</i> pwr(<i>double</i> x, <i>double</i> y)
Description	<p>"Signed" power function. pwr(x,y) returns:</p> <ul style="list-style-type: none"> • x^y if $x \geq 0$, • $-(-x)^y$ if $x < 0$
Examples	<pre>pwr(10.0,2.0) = 100.0 pwr(-10.0,2.0) = -100.0</pre>

log(x,y)

Prototype	<i>complex</i> log(<i>complex</i> x, <i>double</i> y) <i>double</i> log(<i>double</i> x, <i>double</i> y)
Description	Calculates logarithm x to base y.
Examples	<pre>log(128,2) = 7 log(PI,PI) = 1.0 log(-10.0,10.0) : math error log(-10.0+0j,10.0) = 1+1.36437j log(1j,10.0) = 0+682.1e-3j</pre>

ln, log

Prototype	<i>complex</i> ln(<i>complex</i>) <i>double</i> ln(<i>double</i>)
Description	<p>Calculates the natural logarithm.</p> <p>log() with one argument can be used instead of ln().</p>
Examples	<pre>ln(100) = 4.60517 ln(-1.0) : math error ln(-1.0+0j) = 0+3.14159j</pre>

lg, log10

Prototype	<i>complex</i> lg(<i>complex</i>) <i>double</i> lg(<i>double</i>)
Description	<p>Calculates logarithm to base ten.</p> <p>log10() can be used instead of lg().</p>
Examples	<pre>lg(100.0) = 2 lg(-100.0) : math error lg(-100.0+0j) = 2+1.36437j</pre>

lb, log2

Prototype	<i>complex</i> lb(<i>complex</i>) <i>double</i> lb(<i>double</i>)
Description	Calculates logarithm to base two. log2() can be used instead of lb().
Examples	lb(128) = 7 lb(-8.0) : math error lb(-8.0+0j) = 3+4.53236j

db

Prototype	<i>double</i> db(<i>double</i>) <i>double</i> db(<i>double</i> x, <i>double</i> y)
Description	db(x) calculates value of x in decibel, as: $20 \cdot \log_{10}(\text{abs}(x))$ db(x, y) calculates value of the ratio x/y in decibel, as: $20 \cdot \log_{10}(\text{abs}(x/y))$
Examples	db(100)=40 db(0.1,20.0) = -46.0205999133

par

Prototype	<i>complex</i> par(<i>complex</i> ,...) <i>double</i> par(<i>double</i> ,...)
Description	Parallel connection of real or complex impedances. Number of arguments is not limited.
Examples	par(1.0,1.0) = 0.5 par(1.0,2.0,3.0,4.0) = par(par(1.0,2.0),par(3.0,4.0)) = 0.48

sin, cos, tan, tg

Prototype	<i>double</i> sin(<i>double</i>) <i>double</i> cos(<i>double</i>) <i>double</i> tan(<i>double</i>)
Description	Calculates sine, cosine, tangent. tg() can be used instead of tan().
Examples	sin(1.570796327) = 1.0 cos(1.570796327) = 0.0 tan(0.78539816339) = 1.0

asin, acos, atan

Prototype	<i>double</i> asin(<i>double</i>) <i>double</i> acos(<i>double</i>) <i>double</i> atan(<i>double</i>)
Description	Calculates arcsine, arccosine, arctangent. asin returns angle in the range $-\pi/2 \dots \pi/2$ acos returns angle in the range $0 \dots \pi$. atan returns angle in the range $-\pi/2 \dots \pi/2$.
Examples	asin(1.0) = 1.57079632679 acos(1.0) = 0 atan(1.0) = 0.785398163397

atan2

Prototype	<i>double</i> atan2(<i>double</i> x, <i>double</i> y)
Description	Calculates arctangent of x/y . Returns angle in the range $-\pi \dots \pi$.
Examples	atan2(1.0,1.0) = 0.785398163397

random, rand

Prototype	<i>double</i> random(<i>double</i>)
Description	random(x) returns random number with uniform distribution in the range $0 \dots x$. rand() can be used instead of random().
Examples	rand(3.0) = 1.2937463

gauss

Prototype	<i>double</i> gauss(<i>double</i> m, <i>double</i> d)
Description	gauss(m,d) returns normally distributed random number with mean value m and standard deviation d.
Examples	gauss(0,2) = -.8678275

round

Prototype	<i>double</i> round(<i>double</i>) <i>double</i> round(<i>double</i> x, <i>double</i> y)
Description	round(x) rounds x to the nearest integer. round(x,y) rounds x to the nearest multiple of y. Returns x if y<=0.
Examples	round(1.5) = 2.0 round(-1.5) = -1.0 round(3.1415,0.1) = 3.1

floor

Prototype	<i>double</i> floor(<i>double</i>)
Description	Rounds down: finds the largest integer not greater than the argument, and returns it as a <i>double</i> .
Examples	floor(1.6) = 1.0 floor(-1.6) = -2.0

ceil

Prototype	<i>double</i> ceil(<i>double</i>)
Description	Rounds up: finds the smallest integer not less than the argument, and returns it as a <i>double</i> .
Examples	ceil(1.6) = 2.0 ceil(-1.6) = -1.0

bool

Prototype	<i>bool</i> bool(<i>bool</i>) <i>bool</i> bool(<i>int</i>) <i>bool</i> bool(<i>int64</i>) <i>bool</i> bool(<i>double</i>) <i>bool</i> bool(<i>complex</i>)
Description	Returns false if argument is equal to zero, returns true if argument is non-zero. bool(x) works exactly the same as type-casting operator (bool)x.
Examples	bool(0) = false bool(1.5) = true bool(1.0+2.0j) = true

bool *C-keyword*

Description	Declares boolean variable or array.
Examples	<pre>bool b; bool var = false; bool array[10]; bool array[] = { true, false, true };</pre>

(bool) *type-casting operator*

Description	Declares boolean variable or array.
Examples	<pre>bool b; bool var = false; bool array[10]; bool array[] = { true, false, true };</pre>

int

Prototype	<pre>int int(bool) int int(int) int int(int64) int int(double) int int(complex)</pre>
Description	<p>Returns argument value converted to <code>int</code> type.</p> <p><code>int(bool x)</code> returns 0 if <code>x=false</code>, and returns 1 if <code>x=true</code>. <code>int(double x)</code> converts double to <code>int</code> by truncating (discarding the fractional part).</p> <p><code>int(complex x)</code> converts double real part of a complex number to <code>int</code> by truncating (discarding the fractional part).</p> <p><code>int(x)</code> works exactly the same as type-casting operator <code>(int)x</code>.</p>
Examples	<pre>int(true) = 1 int(1.6) = 1 int(-1.6) = -1 int(1.1+2.2j) = 1</pre>

int *C-keyword*

Description	Declares integer variable or array.
Examples	<pre>int i; int var = 10; int array[10]; int array[] = { 0, 1, 2, 3 };</pre>

(int) *type-casting operator*

Description	Converts a number of arbitrary type to an integer. (int)x works exactly the same as function int(x).
Examples	i = (int)x;

int64

Prototype	int64 int64(bool) int64 int64(int) int64 int64(int64) int64 int64(double) int64 int64(complex)
Description	Returns argument value converted to int64 type. int64(bool x) returns 0i64 if x=false, and returns 1i64 if x=true.int64(double x) converts double to int64 by truncating (discarding the fractional part). int64(complex x) converts double real part of a complex number to int64 by truncating (discarding the fractional part). int64(x) works exactly the same as type-casting operator (int64)x.
Examples	int64(true) = 1i64 int64(1.6) = 1i64 int64(-1.6) = -1i64 int64(1.1+2.2j) = 1i64

int64 *C-keyword*

Description	Declares 64-bit integer variable or array.
Examples	int64(true) = 1i64 int64(1.6) = 1i64 int64(-1.6) = -1i64 int64(1.1+2.2j) = 1i64

(int64) *type-casting operator*

Description	Converts a number of arbitrary type to an 64-bit integer. (int64)x works exactly the same as function int64(x).
Examples	i = (int64)x;

double

Prototype	double double(<i>bool</i>) double double(<i>int</i>) double double(<i>int64</i>) double double(<i>double</i>) double double(<i>complex</i>)
Description	<p>Returns argument value converted to <code>double</code> type.</p> <p>double(<i>bool</i> <i>x</i>) returns 0.0 if <i>x</i>=false, and returns 1.0 if <i>x</i>=true.</p> <p>double(<i>complex</i> <i>x</i>) returns real part of a complex number <i>x</i>.</p> <p>double() works exactly the same as type-casting operator (double).</p>
Examples	<pre>double(true) = 1.0 double(1) = 1.0 double(1.1+2.2j) = 1.1</pre>

double C-keyword

Description	Declares double variable or array
Examples	<pre>double x; double var = 12.34; double array[10]; double array[] = { 1.2, 3.4, 5.6 };</pre>

(double) type-casting operator

Description	<p>Converts a number of arbitrary type to a double.</p> <p>(double)<i>x</i> works exactly the same as function double(<i>x</i>).</p>
Examples	<pre>x = (double)1/2;</pre>

complex

Prototype	complex complex(<i>bool</i>) complex complex(<i>int</i>) complex complex(<i>int64</i>) complex complex(<i>double</i>) complex complex(<i>complex</i>)
Description	<p>Returns argument value converted to <code>complex</code> type.</p> <p>complex(<i>bool</i> <i>x</i>) returns 0.0 if <i>x</i>=false, and returns 1.0 if <i>x</i>=true.</p> <p>complex(<i>x</i>) works exactly the same as type-casting operator (complex)<i>x</i>.</p>
Examples	<pre>complex(true) = 1.0+0j complex(2) = 2.0+0j</pre>

complex *C-keyword*

Description	Declares complex variable or array.
Examples	<pre>complex c; complex var = 1.2+3.4j; complex array[10]; complex array[] = { 1.0, 1.0j, -1.0, -1.0j };</pre>

(complex) *type-casting operator*

Description	Converts a number of arbitrary type to a complex. (complex)x works exactly the same as function complex(x).
Examples	<pre>x = sqrt((complex)(-1));</pre>

Script commands

In alphabetical order.

ac

Usage	<pre>ac; ac from; ac from, to; ac from, to, points; ac from, to, points, scale;</pre>
Description	<p>Set AC analysis parameters and perform AC analysis.</p> <p><i>from</i> : start frequency <i>to</i> : stop frequency <i>points</i> : number of points <i>scale</i> = log or lin : logarithmic or linear frequency scale.</p> <p>If called from the script, command will not return until AC analysis is completed. If called from console or HTTP link, returns immediately. Use <code>ready</code> command to check for analysis completion.</p>
Examples	<pre>ac; ac 1M; ac 1M, 100M; ac 1M, 100M, 500; ac 1M, 100M, 500, lin;</pre>

clear

Usage	<code>clear;</code>
Description	Clear storage.

close

Usage	<code>close;</code>
Description	Close active document.

cmd

Usage	<code>cmd <i>command_line</i>;</code>
Description	Execute Windows command line <i>command_line</i> .
Examples	<pre>cmd nl5.exe rc.nl5; cmd "C:\Arduino\arduino.exe" --upload "C:\Arduino\demo\demo.ino";</pre>

cont

Usage	<code>cont;</code> <code>cont screen;</code> <code>cont screen, step;</code>
Description	Continue transient. <i>screen</i> : screen size <i>step</i> : calculation step If called from the script, command will not return until transient is completed. If called from console or HTTP link, returns immediately. Use <code>ready</code> command to check for transient completion.
Examples	<code>cont;</code> <code>cont 1m;</code> <code>cont 1m, 10n;</code>

cursors

Usage	<code>cursors left, right;</code> <code>cursors on;</code> <code>cursors off;</code>
Description	<code>cursors left, right</code> : set cursors (transient or AC) to specified positions and show cursors. <i>left</i> : position of the left cursor <i>step</i> : position of the right cursor <code>cursors on</code> : show cursors. <code>cursors off</code> : hide cursors.
Examples	<code>cursors 1.5, 2.5;</code> <code>cursors off;</code>

display

Usage	<code>display on;</code> <code>display off;</code>
Description	<code>display on</code> : show transient and AC windows. <code>display off</code> : hide transient and AC windows.

exit

Usage	<code>exit;</code>
Description	Close all documents and exit NL5. Cannot be called from console command line.

export (*transient*)

Usage	<pre> export; export filename; export filename, from; export filename, from, to; export filename, from, to, step; </pre>
Description	<p>Export transient traces into csv file.</p> <p><i>filename</i> : name of the file to export traces <i>from</i> : start of the data interval <i>to</i> : end of the data interval <i>step</i> : time step</p> <p>If <i>filename</i> is omitted, name of the file to export is the same as script file name, with “csv” extension. If file path is not specified, export in the script file directory. Extension “csv” can be omitted.</p> <p>Number of points cannot exceed Max number of points value defined in the Preferences dialog box, Transient page.</p> <p>If <i>step</i> is omitted, 101 points will be exported.</p> <p>Only traces currently shown on the graph will be exported.</p>
Examples	<pre> export; export rc_traces; export rc_traces, 0, 100; export rc_traces, 0, 100, 0.1; </pre>

export (*AC*)

Usage	<pre> export; export filename; export filename, from; export filename, from, to; export filename, from, to, points; export filename, from, to, points, scale; </pre>
Description	<p>Export AC traces into csv file.</p> <p><i>filename</i> : name of the file to export traces. <i>from</i> : start frequency. <i>to</i> : end frequency. <i>points</i> : number of points. <i>scale</i> = log or lin : logarithmic or linear frequency scale.</p> <p>If <i>filename</i> is omitted, name of the file to export is the same as script file name, with “csv” extension. If file path is not specified, export in the script file directory. Extension “csv” can be omitted.</p> <p>Only traces currently shown on the graph will be exported.</p>
Examples	<pre> export; export ac_traces; export ac_traces, 1m, 1k; export ac_traces, 1m, 1k, 100; export ac_traces, 1m, 1k, 100, lin; </pre>

import (*transient*)

Usage	<pre>import filename; import filename, cf, cn, rf, rn, hr, tc, ts;</pre>
Description	<p>Import transient traces from text file or scope data file.</p> <p><i>filename</i> : name of the file to import traces (with extension).</p> <p><i>filename</i> should contain extension to specify type of data file. The following extensions/types are supported:</p> <ul style="list-style-type: none"> "txt" and "csv" – text file, comma-separated. "wfm" – Tektronix waveform format. "isf" – Tektronix interval format. "bin" – Keysight Technologies (Agilent) binary format. "trc" – LeCroy binary format. <p>More parameters can be used for import from text (comma-separated) files only.</p> <ul style="list-style-type: none"> <i>cf</i> : first data column to import (1 is first column of the file). <i>cn</i> : number of data columns to import. If <i>cn</i> = -1, import all available columns after first. <i>rf</i> : first data row to import (1 is first row of the file). <i>rn</i> : number of data rows to import. If <i>rn</i> = -1, import all available rows after first. <i>hr</i> : header row. If <i>hr</i> = 0, add header row: "<i>trace(s), trace1, trace2, ...</i>" <i>tc</i> : time column. If <i>tc</i> = 0, add time column with time step <i>ts</i>. <i>ts</i> : time step of added time column floating point (set any value if not used).
Examples	<pre>import scope_traces.isf; import rc_traces.csv, 2, -1, 2, -1, 1, 1, 0;</pre>

logdata

Usage	<pre>logdata filename, expr1,...; logdata +, filename, expr1,...; logdata;</pre>
Description	<p><code>logdata</code> with parameters is the first data logging command.</p> <p><i>filename</i> : name of the file to export traces <code>+</code> : flag to append the data into existing file <i>exprN</i> : expression to be logged</p> <p>If a file <i>filename</i> does not exist, creates a new log file and writes a header. If a file <i>filename</i> already exists, and a first parameter is <code>+</code>, a new data will be appended to existing data, otherwise old data will be overwritten. Extension "csv" in the file name can be omitted. If file path is not specified, creates log file in the script file directory.</p> <p><code>logdata</code> without parameters evaluates expressions <i>exprN</i> specified in the first <code>logdata</code> command and writes results into the log file as comma-separated string.</p>
Examples	<pre>logdata rclog, r1, v(r1), v(c1).rms; logdata +, rcapp, r1, v(r1), v(c1).rms; logdata;</pre>

open

Usage	<pre>open filename;</pre>
Description	Open schematic file <i>filename</i> . Extension "nl5" can be omitted. If file path is not specified, search in the script file directory.
Examples	<pre>open "c:\Project files\nl5\rc.nl5"; open rc;</pre>

pause

Usage	<pre>pause;</pre>
Description	Pause transient. Command can be called from console command line and HTTP link only.

ready

Usage	<pre>ready;</pre>
Description	<p>Check if transient or AC analysis is completed. Returns "0" if analysis is still running, returns "1" if completed.</p> <p>Command can be called from console command line and HTTP link only.</p>

return

Usage	<code>return;</code>
Description	Stop executing the script
Examples	<code>return;</code>

rununtil

Usage	<code>rununtil;</code> <code>rununtil <i>expr</i>;</code>
Description	Set up “run until” transient mode. If parameter <i>expr</i> is omitted, turn off “run until” mode and clear “run until” expression. Otherwise turn on “run until” mode and use parameter <i>expr</i> as “run until” expression.
Examples	<code>rununtil;</code> <code>rununtil V(C1)<0;</code>

save

Usage	<code>save;</code> <code>save <i>filename</i>;</code>
Description	Save schematic into a file <i>filename</i> . Extension “nl5” can be omitted. If file path is not specified, save in the script file directory. If parameter <i>filename</i> is omitted, save into the same file.
Examples	<code>save;</code> <code>save rcnew;</code>

savedata

Usage	<code>savedata;</code> <code>savedata <i>filename</i>;</code>
Description	Save traces into “nlt” data file. Extension “nlt” can be omitted. If parameter <i>filename</i> is omitted, name of the file to save data is the same as script file name, with “nlt” extension. If file path is not specified, save in the script file directory. Only traces currently shown on the graph will be saved.
Examples	<code>savedata;</code> <code>savedata rctraces;</code>

saveic

Usage	<code>saveic;</code>
Description	Save Initial Conditions (IC).

scope.cmd

Usage	<code>scope.cmd <i>command</i>;</code>
Description	Send command <i>command</i> to the scope, returns scope response.
Examples	<code>scope.cmd :CHAN1:LABEL?;</code>

scope.get

Usage	<code>scope.get <i>number</i>;</code>
Description	Get a name of an instrument number <i>number</i> , <i>number</i> = 0... <i>number_of_instruments</i> -1

scope.getn

Usage	<code>scope.getn;</code>
Description	Get number of instruments.

scope.image

Usage	<code>scope.image;</code>
Description	Read scope screen image.

scope.log

Usage	<code>scope.log;</code>
Description	Read content of the Log tab of Scope window.

scope.off

Usage	<code>scope.off;</code>
Description	Close Scope tool window.

scope.on

Usage	<code>scope.on;</code>
Description	Open Scope tool window, refresh instruments list. All scope script commands can be performed with scope window closed, however it must be opened at least once in order to load VISA Library.

scope.read

Usage	<code>scope.read;</code>
Description	Read traces from the scope.

scope.refresh

Usage	<code>scope.refresh;</code>
Description	Refresh instruments list.

scope.run

Usage	<code>scope.run;</code>
Description	Run the scope in continuous mode.

scope.select

Usage	<code>scope.select <i>number</i>;</code>
Description	Select an instrument number <i>number</i> , <i>number</i> = 0... <i>number_of_instruments</i> -1

scope.single

Usage	<code>scope.single;</code>
Description	Run the scope in single mode.

scope.status

Usage	<code>scope.status;</code>
Description	Get text from the Scope window status bar.

scope.stop

Usage	<code>scope.stop;</code>
Description	Stop the scope.

scope.update

Usage	<code>scope.update;</code>
Description	Update scope configuration (read settings from the scope, update Scope window controls).

show

Usage	<code>show window;</code>
Description	<p>Show or activate window specified by parameter <i>window</i>. The following <i>window</i> values can be used:</p> <pre> tran : transient; ac : AC; dc : DC sweep; xy : XY diagram; ed : Eye diagram; ah : Amplitude histogram; fft : FFT; th : Transient histogram; pow : Power; smith : Smith chart; ny : Nyquist plot; ach : AC histogram. </pre>
Examples	<code>show tran;</code>

silent

Usage	<pre> silent on; silent off; </pre>
Description	<pre> silent on : do not show script execution log. silent off : show script execution log. </pre>

sleep

Usage	<code>sleep time;</code>
Description	Pause script execution for <i>time</i> ms.
Examples	<code>sleep 1000;</code>

stop

Usage	<code>stop;</code>
Description	Stop transient. This command can be used to free memory allocated for transient analysis. Transient cannot be continued after this command.

store

Usage	<code>store;</code> <code>store <i>expr</i>;</code>
Description	Move run into storage. The parameter <i>expr</i> is evaluated as an expression, and the result is used as a storage name. If parameter <i>expr</i> is omitted, a default storage name "RunN" is used.
Examples	<code>store;</code> <code>store R1*C1;</code>

storetext

Usage	<code>storetext;</code> <code>storetext <i>text</i>;</code>
Description	Move run into storage with parameter <i>text</i> as a storage name. If parameter <i>text</i> is omitted, a default storage name "RunN" is used.
Examples	<code>storetext;</code> <code>storetext This is a first run;</code>

traces

Usage	<code>traces <i>stateN</i>,...;</code>
Description	Hide or show traces on the graph. The parameter <i>stateN</i> specifies show/hide status of the trace number N (traces are listed in the same order as in the Transient/Data or AC/Data window). <i>stateN</i> = 0 – hide trace; otherwise – show trace.
Examples	<code>traces 0,1,1,0,0,1;</code>

tracename (*transient*)

Usage	<pre>tracename; tracename from; tracename from, to; tracename from, to, step;</pre>
Description	<p>Request transient trace data as a comma-separated string.</p> <p><i>from</i> : start of the data interval. <i>to</i> : end of the data interval. <i>step</i> : step.</p> <p><i>tracename</i>; - returns 101 points of entire <i>tracename</i> interval. <i>tracename from</i>; - returns only one trace value at <i>t=from</i>. <i>tracename from, to</i>; - returns 101 points in specified interval. <i>tracename from, to, step</i>; - returns data points in specified interval with specified step.</p> <p>Trace <i>tracename</i> should be specified in the Transient Data, however it does not need to be displayed on the graph or in the table.</p> <p>Please note that length of the returned string may be limited, and the limit may be different for different applications. If you got a time-out on this command, please reduce the number of data points requested by one command.</p> <p>This command can be called from HTTP link only.</p>
Examples	<pre>V(R1) ; V(R1) 1.23; V(R1) 0, 100; V(R1) 0, 10, 0.1;</pre>

tracename (AC)

Usage	<pre>tracename; tracename from; tracename from, to; tracename from, to, points; tracename from, to, points, scale;</pre>
Description	<p>Request AC trace data as a comma-separated string.</p> <p><i>from</i> : start frequency. <i>to</i> : end frequency. <i>points</i> : number of points. <i>scale</i> = log or lin : logarithmic or linear frequency scale.</p> <p><i>tracename</i>; - returns all calculated data points of <i>tracename</i> trace. <i>tracename from</i>; - returns only one trace value at $f=from$. <i>tracename from, to</i>; - returns all calculated data points in the specified interval. <i>tracename from, to, points</i>; - returns specified number of points in the specified interval. <i>tracename from, to, points, scale</i>; - returns data with specified scale type.</p> <p>Trace <i>tracename</i> should be specified in the AC Data, however it does not need to be displayed on the graph or in the table.</p> <p>Please note that length of the returned string may be limited, and the limit may be different for different applications. If you got a time-out on this command, please reduce the number of data points requested by one command.</p> <p>This command can be called from HTTP link only.</p>
Examples	<pre>V(R1); V(R1) 12.34; V(R1) 1, 100; V(R1) 1, 10, 100; V(R1) 1, 10, 100, lin;</pre>

tran

Usage	<pre>tran; tran <i>start</i>; tran <i>start</i>, <i>screen</i>; tran <i>start</i>, <i>screen</i>, <i>step</i>;</pre>
Description	<p>Set transient parameters and start transient.</p> <p><i>start</i> : start of transient display <i>screen</i> : screen size <i>step</i> : calculation step</p> <p>If called from the script, command will not return until transient is completed. If called from console or HTTP link, returns immediately. Use <code>ready</code> command to check for transient completion.</p>
Examples	<pre>tran; tran 0; tran 0, 10m; tran 0, 10m, 1u;</pre>

Script examples

Set component parameters. Component parameters have been calculated in external application (for instance, Excel), or entered manually and saved into the text file in the *name=value* format:

```
R1 = 5.1;
C1 = 12e-9;
V3.period = 0.01;
```

Run the script to apply new parameters to components.

Sweep parameter. Component parameter is changing in specified range, transient analysis performed for each parameter, results placed into storage:

```
for( R1=1; R1<=10; R1+=1 )
{
    tran;
    store R1;
}
```

Sweep parameter from the list. Component parameter is assigned value from the list, AC analysis performed for each parameter, results placed into storage:

```
for( V1.period = 1m, 2m, 10m, 50, 100m )
{
    ac;
    store V1.period;
}
```

Sweep variable. Local variable is changing in some range, component parameters modified, transient analysis performed, results placed into storage:

```
double freq;
for( freq=1; freq<=10; freq*=1.1 )
{
    V2.period = 1 / freq;
    R2 = 1 / (freq * C5);
    tran;
    store freq;
}
```

Wait for condition. Transient is running until peak-to-peak value of the trace is less than specified threshold. When done, Initial Conditions are saved.

```
double threshold = 1e-6;
tran;
while( V(C1).pp > threshold ) cont;
saveic;
```

Perform analysis for specified file, save data, exit application. Schematic file is loaded into NL5, component parameters changed, transient analysis performed, traces exported into “csv” file, NL5 closed. This script can be executed from command line.

```
open lcr.nl5;
```

```
R1=100;  
C1=1n5;  
tran;  
export data.csv;  
exit;
```

Perform analysis for specified file, log data, exit application. Schematic file is loaded into NL5, component parameter swept, transient analysis performed, traces data logged into text file, NL5 closed. This script can be executed from command line.

```
open lcr.nl5;  
logdata lcrdata.csv, r1, V(R1).mean, V(R1).rms;  
for( R1=100; R1<=1000; R1+=100 )  
{  
    tran;  
    logdata;  
}  
exit;
```

HTTP functions

List of HTTP functions.**General**

NL5_GetInfo
 NL5_New
 NL5_Open
 NL5_Save
 NL5_Close
 NL5_Show
 NL5_GetActive
 NL5_GetParam
 NL5_SetParam
 NL5_DisableCmp
 NL5_EnableCmp

Transient

NL5_GetTracesSize
 NL5_AddTrace
 NL5_DeleteTrace
 NL5_GetTrace
 NL5_GetTraceValue
 NL5_SetTrace
 NL5_ShowTran
 NL5_SetScreen
 NL5_Cursors
 NL5_Start
 NL5_Pause
 NL5_Continue
 NL5_Stop
 NL5_SaveIC
 NL5_IsRunning
 NL5_GetDataSize
 NL5_GetDataAt
 NL5_GetData
 NL5_AddData
 NL5_Export
 NL5_GetStorageSize
 NL5_Store
 NL5_DeleteStorage

AC

NL5_GetACTracesSize
 NL5_AddACTrace
 NL5_DeleteACTrace
 NL5_GetACTrace
 NL5_GetACTraceValue
 NL5_SetACTrace
 NL5_ShowAC
 NL5_SetACScreen
 NL5_ACCursors
 NL5_StartAC
 NL5_StopAC
 NL5_GetACDataSize
 NL5_GetACDataAt
 NL5_GetACStorageSize
 NL5_StoreAC
 NL5_DeleteACStorage

NL5_GetInfo

Get NL5 version info

Request: NL5_GetInfo

Response: 0,Version Ver.Rev.Core.GUI, Date MM/DD/YYYY

NL5_New

Create new document

Request: NL5_New

Response: *ndoc*

ndoc : document handle

NL5_Open

Open document *name*

Request: NL5_Open, *name*

Response: *ndoc*

ndoc : document handle

NL5_Save

Save active document

Request: NL5_Save

NL5_Save, *name*

Response: 0

name : file name. If not provided, save active document with existing file name.

NL5_Close

Close active document

Request: NL5_Close

Response: 0

NL5_Show

Show schematic of active document, or activate document and show schematic.

Request: NL5_Show

NL5_Show, *ndoc*

Response: *ndoc*

ndoc : document handle

If *ndoc* not provided, show schematic of active document.

If *ndoc* provided, activate document with document handle *ndoc* and show schematic.

NL5_GetActive

Get document handle of active document

Request: NL5_GetActive

Response: *ndoc*

ndoc : document handle

NL5_GetParam

Get value of component parameter.

Request: NL5_GetValue, *param*

Response: 0, *value*
0, *value*, *text*

param : *cmp.model* – return model name of component *cmp*
cmp.param – return value of *cmp.param*
cmp - return value of *cmp* first parameter

If parameter is a number, return number: 0,1.23.

If parameter has a formula, return number and text of the formula: 0,2.46,=R1.R*2.

If parameter is a text, return text: 0,filename.nl5.

If parameter is a text from the list, return index of the text and text: 0,1,On.

NL5_SetParam

Set value of component parameters.

Request: NL5_SetParam, *expr1*, *expr2*, ...

Response: 0

expr : *name=value*
name : *cmp.model* – set model of component *cmp*
cmp.param – set value of *cmp.param*
cmp - set value of *cmp* first parameter
value : number or text

Any number of *expr* can be provided.

Text with commas should be placed into double quotes: "1,2,3,4", or HEX formatted.

Text or number with forbidden characters should be HEX formatted.

Set formula for parameter as "*=formula*": =R1.R*2.

To clear the parameter text, give empty *value*: IC=.

If parameter is a text from the list, it can be set as an index of the text or as a text: D1.IC=1, D1.IC=On.

NL5_DisableCmp

Disable one or more components.

Request: NL5_DisableCmp, *name1*, *name2*, ...

Response: 0

name : component name: R1

Any number of components can be disabled by one function.

If Label component is being disabled, all Labels with that name will be disabled.

NL5_EnableCmp

Enable one or more components.

Request: NL5_EnableCmp, *name1*, *name2*, ...

Response: 0

name : component name: R1.

Any number of components can be enabled by one function.

If Label component is being enabled, all Labels with that name will be enabled.

NL5_GetTracesSize

Get number of transient traces.

Request: NL5_GetTracesSize

Response: *number*

number: number of traces

NL5_AddTrace

Add new transient trace.

Request: NL5_AddTrace, *type*, *expr*

Response: *index*, *handle*, *name*

type : V, I, P, State, Variable, Function, Data (*text*)

expr :
for V, I, P, State: component name
for Variable: variable name
for Function: function expression
for Data: trace name (optional)

index : trace index (≥ 0)

handle : trace handle (< 0)

name : trace name

NL5_DeleteTrace

Delete all or one transient trace.

Request: NL5_DeleteTrace, *trace*

Response: 0

trace : all (*text*) – delete all traces
 number >= 0 - trace index
 number < 0 - trace handle
 text - trace name

NL5_GetTrace

Get transient trace index, handle, and name.

Request: NL5_GetTrace, *trace*

Response: *index, handle, name*

trace : *number* >= 0 - trace index
 number < 0 - trace handle
 text - trace name

index : trace index (>=0)
handle : trace handle (<0)
name : trace name

NL5_GetTraceValue

Get value *name* of transient trace, or trace value at specified time.

Request: NL5_GetTraceValue, *trace, name* [, storage=*index*]

Response: 0, *value*

trace : *number* >= 0 - trace index
 number < 0 - trace handle
 text - trace name

name : *number*: get trace value at time=*number*
 last, left, right, delta, min, max, pp, mean, rms, sd, freq, period, integral:
 name of requested trace value (text, same as trace values displayed in the table).

storage=*index* : (optional) storage index. If not provided, return value for the last Run.

value : trace value

NL5_SetTrace

Set properties of all or selected transient traces.

Request: NL5_SetTrace, *trace1*,..., *tracen*, *expr1*, *expr2*, ...

Response: 0

trace : all (*text*) – set all traces
 number >= 0 - trace index
 number < 0 - trace handle
 text - trace name

expr : *property_name* = *value* . All expressions are optional.

show=on, off – display trace on the screen

name=*text* – set trace name to *text*. To erase name, provide “name=” without text.

type=analog, digital, bus

analog=1,2,3,4 – set trace analog section

scale=*number*

mid=*number*

shift=*number* - time shift

color=*text* or *number*

Trace color can be defined using the following predefined texts:

aqua, black, blue, gray, green, lime, navy, olive, purple, red, white, yellow

or hexadecimal number 0xBBGGRR, where BB, GG, RR are hexadecimal values of blue, green, and red parts of the color. For example, 0xff0000 = blue, etc.

width=*number* – trace width

line=on, off - show line

dp=on, off - show data points

NL5_ShowTran

Show transient window.

Request: NL5_ShowTran

NL5_ShowTran, *ndoc*

Response: *ndoc*

ndoc : document handle. Activate document and show transient window.
If not provided, show transient window of active document.

NL5_SetScreen

Set properties of transient screen.

Request: NL5_SetScreen, *expr1*, *expr2*, ...

Response: 0

expr : *name=value*. All expressions are optional.

fit=hor, vert, screen : fit horizontal, fit vertical, fit the screen

analog=add : add analog section

analog=remove: remove active analog section

analog=*n* : activate analog section *n* = 1...4

maximize=on, off : maximize active analog section

separate=on, off : separate traces in active analog section

start=*number* : horizontal start, s

screen= *number* : horizontal screen, s

NL5_Cursors

Show/hide transient cursors, set cursors to specified positions, set Table mode to Cursors. Return current cursors position.

Request: NL5_Cursors, *expr1*, *expr2*, ...

Response: 0, *left*, *right*

expr : *name* or *name=value*. All expressions are optional.

on : show cursors

off : hide cursors

left=*number* : set left cursor position

right= *number* : set right cursor position

left : left cursor position.

right : right cursor position.

NL5_Start

Start transient simulation.

Request: NL5_Start, *extr1*, *expr2*, ...

Response: 0

expr : *name=value*. All expressions are optional.

start= *number* : left edge of the screen simulation

screen= *number* : transient screen size

step= *number* : simulation step

NL5_Pause

Pause transient simulation.

Request: NL5_Pause

Response: 0

NL5_Continue

Continue transient simulation. Optional transient settings can be provided.

Request: NL5_Continue, *extr1*, *expr2*, ...

Response: 0

expr : *name=value*. All expressions are optional.

screen= number : transient screen size

step= number : simulation step

NL5_Stop

Stop transient simulation.

Request: NL5_Stop

Response: 0

NL5_SaveIC

Save Initial Conditions of transient simulation.

Request: NL5_SaveIC

Response: 0

NL5_IsRunning

Check status of transient simulation, get last simulation time if paused or running.

Request: NL5_IsRunning

Response: 0 - no active simulation

0,*time* - transient simulation is paused at *time*

1,*time* - transient simulation is running at *time*

NL5_GetDataSize

Get size of transient trace data (last Run or storage), and time of last data point.

Request: NL5_GetDataSize, *trace* [,storage=*index*]

Response: *size,tlast*

trace : *number* >= 0 - trace index
 number < 0 - trace handle
 text - trace name

storage=*index* : (optional) storage index. If not provided, return data size of the last Run

size : number of data points
tlast : time of last data point

NL5_GetDataAt

Get transient trace data points as *time,value* comma-separated values.

Request: NL5_GetDataAt, *trace, expr1, expr2, ...*

Response: *points,t1,v1,t2,v2,...*

trace : *number* >= 0 - trace index
 number < 0 - trace handle
 text - trace name

expr : *name=value*. All expressions are optional.

storage=*index* : storage index. If not provided, return data of the last Run

start=*number* : start data point >= 0. If not provided, start = 0

points=*number* : number of data points. If not provided, points=1

points : number of data points returned (>=0)
t,v : comma-separated time and value

NL5_GetData

Get interpolated transient trace data points

Request: NL5_GetData, *trace*, *expr1*, *expr2*, ...

Response: *points*, *t1*, *v1*, *t2*, *v2*, ...

trace : *number* ≥ 0 - trace index

number < 0 - trace handle

text - trace name

expr : *name=value*. All expressions are optional.

storage=index : storage index. If not provided, return data of the last Run

from=number : start time. If not provided, from = 0

to=number : end time. If not provided, to = 0

step=number : time step. If not provided, step = 0

points : number of data points returned (≥ 0)

t,v : comma-separated time/value

NL5_AddData

Add data points to the transient trace of Data type.

Request: NL5_AddData, *trace*, *t1*, *v1*, *t2*, *v2*, ...

Response: *size*

trace : *number* ≥ 0 - trace index

number < 0 - trace handle

text - trace name

t,v : comma-separated time/value

size : new data size

NL5_Export

Export transient data of all or selected traces to a text file (comma-separated).

Request: NL5_Export, *file*, *trace1*, ..., *tracen*, *expr1*, *expr2*, ...

Response: *points* – number of points exported

file : file name

trace : all (*text*) – set all traces
number >= 0 - trace index
number < 0 - trace handle
text - trace name

expr : *name=value*. All expressions are optional.

storage=index : storage index. If not provided, return data of the last Run
from=number : start time. If not provided, from = 0
to= number : end time. If not provided, to = 0
step= number : time step. If not provided, step = 0

points: number of points exported

NL5_GetStorageSize

Get size of the storage, last Run not included.

Request: NL5_GetStorageSize

Response: *size*

size: storage size

NL5_Store

Copy Run data to storage

Request: NL5_Store, *name*

Response: *index*

name : storage name. If not provided, default automatic name is used

index: index of added storage

NL5_DeleteStorage

Delete all or one storage.

Request: NL5_DeleteStorage, *index*
NL5_DeleteStorage

Response: *size*

index : index of storage to delete. If not provided, delete all storage.

size: storage size after deleting

NL5_GetACTracesSize

Get number of AC traces.

Request: NL5_GetACTracesSize

Response: *number*

number: number of traces

NL5_AddACTrace

Add new AC trace.

Request: NL5_AddTrace, *type*, *expr*

Response: *index*, *handle*, *name*

type : V, I, Function, Z, Y, Gamma, VSWR, Loop: trace type (text)

expr : for V, I, Z: component name

for Function: function expression

for Z, Y, Gamma, VSWR, Loop – not used (relative to AC source)

index : trace index (≥ 0)

handle : trace handle (< 0)

name : trace name

NL5_DeleteACTrace

Delete all or one AC trace.

Request: NL5_DeleteACTrace, *trace*

Response: 0

trace : all (*text*) – delete all traces

number ≥ 0 - trace index

number < 0 - trace handle

text - trace name

NL5_GetACTrace

Get AC trace index, handle, and name

Request: NL5_GetACTrace, *trace*

Response: *index, handle, name*

trace : *number* \geq 0 - trace index
 number $<$ 0 - trace handle
 text - trace name

index : trace index (≥ 0)
handle : trace handle (< 0)
name : trace name

NL5_GetACTraceValue

Get value *name* of AC trace.

Get value *name* of AC trace, or trace value at specified frequency.

Request: NL5_GetACTraceValue, *trace, name* [, storage=*index*]

Response: 0, *value*

trace : *number* \geq 0 - trace index
 number $<$ 0 - trace handle
 text - trace name
name : *number*: get trace value at frequency=*number*
 last, left, right, delta, min, max, pp, slope:
 name of requested trace value (text, same as trace values displayed in the table).
storage=*index* : (optional) storage index. If not provided, return value for the last Run.

value : trace value

NL5_SetACTrace

Set properties of all or selected AC traces.

Request: NL5_SetACTrace, *trace1*,..., *tracen*, *expr1*, *expr2*, ...
 NL5_SetACTrace, all, *expr1*, *expr2*, ...

Response: 0

trace : all (*text*) – set all traces
number >= 0 - trace index
number < 0 - trace handle
text - trace name

expr : *property_name* = *value*. All expressions are optional.

show=on, off – display trace on the screen

name=*text* – to erase name, provide “name=” without text

color=*text* or *number*

Trace color can be defined using the following predefined texts:

aqua, black, blue, gray, green, lime, navy, olive, purple, red, white, yellow

or hexadecimal number 0xBBGGRR, where BB, GG, RR are hexadecimal values of blue, green, and red parts of the color. For example, 0xff0000 = blue, etc.

width=*number* – trace width

phase_width=*number*

line=on, off - show line

dp=on, off - show data points

NL5_ShowAC

Show AC window.

Request: NL5_ShowAC
 NL5_ShowAC, *ndoc*

Response: *ndoc*

ndoc : document handle. Activate document and show AC window.
 If not provided, show AC window of active document.

NL5_SetACScreen

Set properties of AC screen.

Request: NL5_SetACScreen, *expr1*, *expr2*, ...

Response: 0

expr : *name=value*. All expressions are optional.

fit=hor, vert, screen, phase : fit horizontal, fit vertical, fit the screen, fit phase vertical

hor_scale=lin, log : horizontal scale

left= *number*: screen left, Hz

right= *number*: screen right, Hz

vert_scale=lin, log : vertical scale

db=off, on : show vertical scale in dB

top= *number*: screen top

bot= *number*: screen bottom

phase=off, on, separate : phase display mode

phase_top= *number*: phase top

phase_bot= *number*: phase bottom

NL5_ACCursors

Show/hide AC cursors, set cursors to specified positions, set Table mode to Cursors, get current cursors position.

Request: NL5_Cursors, *expr1*, *expr2*, ...

Response: 0, *left*, *right*

expr : *name* or *name=value*. All expressions are optional.

on : show cursors

off : hide cursors

left= *number*: set left cursor position

right= *number*: set right cursor position

left : left cursor position.

right : right cursor position.

NL5_StartAC

Start AC simulation.

Request: NL5_StartAC, *extr1*, *extr2*, ...

Response: 0

expr : *name=value*. All expressions are optional.

source=text : AC source component name

from= number: start frequency

to= number: end frequency

points= number: number of points

NL5_StopAC

Stop AC simulation.

Request: NL5_StopAC

Response: 0

NL5_GetACDataSize

Get size of AC trace data (last Run or storage), and frequency of last data point.

Request: NL5_GetACDataSize, *trace* [,*storage=index*]

Response: *size,flast*

trace : *number* >= 0 - trace index

number < 0 - trace handle

text - trace name

storage=index : (optional) storage index. If not provided, return data size of the last Run

size : number of data points

flast : frequency of last data point

NL5_GetACDataAt

Get AC trace data points as *freq,mag,phase* comma-separated values.

Request: NL5_GetACDataAt, *trace, expr1, expr2, ...*

Response: *points,f1,mag1,phase1,f2,mag2,phase2,,...*

trace : *number* >= 0 - trace index

number < 0 - trace handle

text - trace name

expr : *name=value*. All expressions are optional.

storage=index : storage index. If not provided, return data of the last Run

start= number: start data point >= 0. If not provided, start = 0

points= number: number of data points. If not provided, points=1

points number of data points returned (>=0)

f,mag,phase : comma-separated frequency, magnitude, phase

NL5_GetACStorageSize

Get size of AC storage, last Run not included.

Request: NL5_GetACStorageSize

Response: *size*

size: storage size

NL5_StoreAC

Copy AC Run data to storage

Request: NL5_StoreAC, *name*

Response: *index*

name : storage name. If not provided, default automatic name is used

index: index of added storage

NL5_DeleteStorage

Delete all or one AC storage.

Request: NL5_DeleteACStorage, *index*

NL5_DeleteACStorage

Response: *size*

index : index of AC storage to delete. If not provided, delete all AC storage.

size: storage size after deleting