CASTLE 3.5 Help and Documentation

USER MANUAL

© 2023 Raymond Allen
Raymond Allen

This page is intentionally left blank.
Remove this text from the manual
template if you want it completely blank.

1. Introduction to CASTLE 3.5 7
1.1 What's New in CASTLE 3.5 ... 8
1.2 CASTLE.DLL ... 10
1.3 Comparison with TL Codes ... 11
1.4 Units ... 12
1.5 Stella ... 13
1.5.1 Stella Trace Operators ... 14
1.6 Notes ... 15
2. CASTLE Main Window 17
2.1 Overview ... 18
2.2 Ribbon Bar ... 19
2.2.1 Application Button ... 19
2.2.2 Quick Access Toolbar ... 20
2.3 Document Pane ... 21
2.4 Status Bar ... 21
3. Schematic Entry 23
3.1 Keyboard Shortcuts ... 28
3.2 Ground ... 28
3.3 Simple RC Example ... 29
3.4 TL Circuit Example ... 36
3.5 Parameters ... 40
3.6 Bubbles and Sub-Circuits ... 40
3.7 Control Commands ... 43
3.8 Math Parser ... 43
3.9 TL Array ... 44
4. Netlist Editor & Simulator 47
4.1 Plot Traces Window ... 48
4.2 Control Commands Lines ... 49
4.3 Simulate Window ... 50
4.4 Plot Select Window ... 51
5. Plot Windows 53
6. CASTLE Simulator Details 55

 © 2023 Raymond Allen

6.1 Nodal Analysis 56
6.2 Modified Nodal Analysis (MNA) 57
6.3 Matrix Solution Methods 59
6.3.1 Pivoting 60
6.4 Transmission-Line Models 60

7. Component Reference 65

7.1 Transmission Line 66
7.2 Transmission Line Inductor 67
7.3 Transmission Line Capacitor 68
7.4 Voltage Source 68
7.5 Current Source 70
7.6 Variable Resistor 72
7.7 Switch 73
7.8 Variable Capacitor 76
7.9 Variable Inductor 78
7.10 Voltage Probe 79
7.11 Current Probe 79
7.12 Transformer 79
7.13 Semiconductor Diode 80
7.14 Simple Diode 81
7.15 SubCircuit 82
7.16 Lossy TL 83
7.17 Lossy TL Transformer 83
7.18 Lossless TL Transformer 85
7.19 Lossy TL Resistance 85
7.20 Railgun 86
7.21 Rod Pinch Diode 87
7.22 Pinched Beam Diode 90
7.23 Cylindrical Diode 92
7.24 Planar Diode 95
7.25 Ekdahl Diode 96
7.26 Sandia Switch 97
7.27 SNL Tom Martin Switch 99
7.28 SNL Resistive Wall 101

 © 2023 Raymond Allen

7.29 SNL Plasma Opening Switch 103
7.30 Fuse 106
7.31 Imploding Load 106
7.32 Saturable Core 109
7.33 MOV 111

Index 113

© 2023 Raymond Allen

This page is intentionally left blank.
Remove this text from the manual
template if you want it completely blank.

Introduction to CASTLE 3.5

1 Introduction to CASTLE 3.5

CASTLE (Circuit Analysis and Simulation with Transmission-Line Emphasis) is a circuit code that bridges the gap between transmission-line codes, such as Bertha, and MNA (modified nodal analysis) codes such as SPICE. CASTLE currently only does transient analysis of a circuit, but other types of analysis may be added in the future.

See CASTLE Main Window 18 for help using the various tabs and buttons on the CASTLE main window.

CASTLE is a relatively new circuit code that is a hybrid of transmission-line (TL) and modified nodal analysis (MNA) circuit codes for transient analysis. The CASTLE program runs on the Windows platform and comprises a graphical schematic entry code, a simulator code, and an analysis and graphing code. The purpose of CASTLE is to provide a modern, graphical schematic entry interface for TL based circuits and also a simulator with the speed benefits of TL codes but with the flexible circuit topology allowed in MNA codes.

For details on how the CASTLE circuit solver works, see CASTLE Simulator Details 56.

For more information, see Comparison with TL Codes 11

Units 12 in CASTLE follow the convention of SPICE (unless otherwise stated)

Note: At this time, the analysis and graphing portions of the code are very limited. Basic graphs can be generated, but CASTLE currently relies on external programs (such as Stella 13) for analysis. A text file output option was recently added so that users without access to Stella can output simulation results to other programs (such as Origin or Excel).

1.1 What's New in CASTLE 3.5

Here are the major improvements in CASTLE 5

- Added TL Array element. Can configure a large array of transmission lines in pairs as either a Chain Link Fence or Rectangular Grid.
- Added final resistance input control for the Railgun model.
- Added significantly more Stella 13 integration. Adding a text box with the ".stella" control keyword 49 invokes Stella integration and Stella is used for waveform interactions. ".PlotStella" text can be used to automatically save waveforms to Stella after simulation.
  - In this mode, stella.dll is loaded and used for things like waveform processing and scalar math functions

© 2023 Raymond Allen

  - If Stella is loaded, all math expressions are first evaluated by Stella's scalar evaluator and the result is used if valid. So, there is now another way to include expressions without using the new math parser 43.
  - Stella scalar values from the given shot can be now be used in values and expressions.
- CASTLE now has a dynamically loaded library version, castle.dll 10.
  - This allows castle to be called to simulate netlists from Stella, Python, Chicago and more.
- Added a way to use MPI (Message Passing Interface) to speed simulations by running in parallel on computers with multiple cores. This is a bit complex and requires the use of the new castle.dll file. You can then invoke CASTLE using Stella from the command line. New "Block" circuit element is added to encapsulate the circuit for a process at the top level, similar to a subcircuit with exactly two transmission-line ends, and a new "Junction" circuit element is added to represent the two ends inside the subcircuit. See Stella 13 section for more info.
- CASTLE from now be invoked from within Stella using the "Run PlugIn" macro command with either "Castle" for regular CASTLE or "CastleMPI" for the MPI version of CASTLE and the number of processes used can be specified.
- A new math parser 43 is added that can evaluate expressions inside curly braces.
- The netlist code is separated from the schematic code so that a saved netlist can once again be run without a Schematic.
- Larger page sizes D and E are added.
- Improvements to the plotting window are made.
- Zooming in and out of schematic with mouse wheel is now supported.
- Stella path is now saved to registry, saving the need to browse for it with new files.
- Restricting releases to 64-bit only
- Fixed Current Source (was generating voltage source instead of current source)
- Added ability to ingore unicode BOM added by external text editors
- Added "Component Reference 66" section to help
- Added ".plottext" command 49 to automatically save output waveforms to a text file
- Added ".inc" command 49 to automatically include lines from specified file into netlist when simulating
- Added ".log" command to enable generation of a log file.
- Added Rotate button (acts the same as pressing the R button on keyboard)
- Bug fixes:
  - Fixed big file bug
  - Fixed memory leak
  - Fixed zoom to area in plots
  - Improved stability of semiconductor diode model
  - Fixed some pointer handling
  - Fixed compiler warning

© 2023 Raymond Allen

- Opening and editing of large netlists fixed
- Fixed problems running netlists without a schematic
- Fixed reiteration bug in Imploding load model (21Feb23)

1.2 CASTLE.DLL

The new CASTLE.DLL allows one to run simulation in a variety of new ways. One can now simulate netlists from Stella, the command line, and other codes (like C++, Python, Chicago).
At this moment, simulation results can only be saved using Stella, so a Stella experiment and shot must be provided to say where input will come from and output waveforms will go to.

Command Line
Running CASTLE from the command line is made possible with a stub program called StellaDllRun.exe.
Doing this only makes sense for now using a Stella shot to save data to (and maybe get input from).
Here's an example of how to invoke castle.dll to run a netlist in regular (non-mpi) mode:

stelladllrun castle;E:\experiments\CastleTest;New_LRU;Injector_Full_11a;v

where "E:\experiments\CastleTest" is the Stella experiment, "New_LRU" is the shot, "Injector_Full_11a" is the netlist, and the "v" is for verbose mode.

Here's an example of how to invoke castle.dll to run a netlist in MPI mode (mpich must be installed):

mpiexec -n 15 stelladllrun castle;E:
\experiments\CastleTest;New_LRU;Injector_MpiFull_11a;mv

where "-n 15" tells mpiexec to use 15 threads, "E:\experiments\CastleTest" is the Stella experiment, "New_LRU" is the shot, "Injector_MpiFull_11a" is the netlist, the "m" is for MPI mode, and the "v" is for verbose mode.

Stella Macro
The CASTLE dll can also be invoked from a Stella macro, either in regular or MPI mode.
Below is an example of running a netlist in regular mode using the "Castle" plugin option and setting scalar values to use as parameters in the simulation. Also is MPI mode with "CastleMPI" plugin, note the # of threads/processes to using is all the way in the right most column.

 © 2023 Raymond Allen

	En	Routine	Option	Config	Shot	Field1	Value1	Field2	Value2	Field3	Va	Fi	Va	Fie	V	Fiel	Valu	Field	Val	Fie	Va	Field9	Field10	Value10	nProcs
1	☑	Run PlugIn	Castle	Mod_Trig_GenNeg11f_ForBasis		pw21	20e-9	td21	380e-9																
2	☑	Run PlugIn	CastleMPI	Injector_MpiNeg_11b																					8
3	☑	2TrMath	aX+bY	v(rbeam)_NegNoMod	v(rbeam)	NegBasis20	-1	1	1e30	2-trace calc															
4	☑	Run PlugIn	Castle	Mod_Trig_GenNeg11f_ForBasis		pw21	25e-9	td21	380e-9																
▶	☑	Run PlugIn	CastleMPI	Injector_MpiNeg_11b																					8
6	☑	2TrMath	aX+bY	v(rbeam)_NegNoMod	v(rbeam)	NegBasis25	-1	1	1e30	2-trace calc															

Note: Parameters defined in netlist will override those in Stella. Also, if shot is defined in netlist usning ".stella" control, then shot will be changed to that in netlist.

1.3 Comparison with TL Codes

CASTLE combines the simplicity and speed advantages of TL codes with the free topology of MNA codes. CASTLE is a MNA code in principle, but inductors and capacitors can either be represented as TL or lumped elements. CASTLE uses a fixed time step, like most TL codes (most MNA codes use a variable time step). In this way, the output of CASTLE will EXACTLY match that of any circuit that can be represented in a TL code. Also, specialized routines written to control the impedance of an element in a TL code can be included in CASTLE with minimal modification. CASTLE intelligently utilizes the isolated aspect of TL elements to subdivide a large nodal matrix into smaller matrices. Only those submatrices that contain time varying impedances need to be inverted at every time step. In this way, the solution speed of CASTLE can rival that of TL codes. CASTLE also includes current and voltage sources, which are sometimes difficult to implement in TL codes. The graphical schematic entry module of CASTLE provides a front-end similar to that of many professional MNA codes (but still a work in progress!). But, schematic entry is improved over MNA codes with respect to TL elements because the ground node does not need to be specified for simple connections between TL elements.

TL codes generally represent all circuit elements as either transmission-lines of variable but quantized time lengths or as resistors. Inductors and capacitors can be represented by TL elements of unit time length (sometimes capacitors are two units long). Also, TL codes generally allow only simple connections between elements, which restrict the possible topology of the circuit. TL codes take fixed time steps during simulation. The time step chosen is sometimes based on the minimum common quantized length of the TL elements or the speed at which time dependent models change. Simulation in TL codes is very efficient because the ends of each element are time isolated from each other so and the end of each TL element forms an isolated subcircuit where only other connected TL ends and connected resistors need be included. Because of the usual topology limitations, only certain connections at TL ends are allowed and so the solution to the each subcircuit can be calculated in advance without the need for matrix inversion. Also, a subcircuit's solution need only be recalculated for those time steps during which a resistance or impedance value is changed. TL codes are very well suited to many pulsed-power circuits where most circuit elements are represented very well by TL elements or resistors and generally have a simple topology.

© 2023 Raymond Allen

MNA codes have no topology limitation. But, solution involves inverting a large matrix with one column and row for each node and voltage source in the circuit. Also, the time step is variable and iteration is used within each time step. MNA codes often have difficulty with pulsed-power circuits because of the rapid switching often involved. It is possible that the simulation will fail because the matrix solution fails to converge. Note that TL elements can be represented in MNA codes and that simple circuits involving TL elements and fixed resistors will have identical solutions to that of TL codes (and CASTLE). Inductor and capacitor models are created from their differential equations. Schematic entry of circuits with many TL elements in MNA codes is generally difficult because the ground node must be specified at every connection. Also, the simulation speed is often very slow compared to TL codes.

1.4 Units

Within CASTLE, units follow the same convention as SPICE (except for some special devices where units other units are explicitly shown):

The default units are Volts, Amps, Ohms, Farads, Henries, Watts, etc.
Values may be entered decimal form (e.g., 0.00245) or exponential form (e.g., 2.45e-3).
Since Castle is not case sensitive 2.45e-3 is the same as 2.45E-3.
You can also use some unit prefix abbreviations, such as "m" for "milli" and therefore write 2.45e-3 as 2.45m.

The following scale abbreviations are recognized (case insensitive):

Abbreviation	Factor	Unit Prefix
F	E-15	femto
P	E-12	pico
N	E-9	nano
U	E-6	micro
M	E-3	milli
K	E+3	kilo
MEG	E+6	mega
G	E+9	giga
T	E+12	tera

Notes:
- Do not leave any space between number and unit prefix abbreviation (write "2.45m" and not "2.45 m").
- Be careful with "M" because "M" and "MEG" are often confused.
- Letters after the first letter and unrecognized letters are ignored. For example, you can write "2.45uF" for a capacitors value, but the "F" character is simply ignored by the simulator.

© 2023 Raymond Allen

1.5 Stella

Stella Overview
Stella is a data acquisition and processing code created and maintained by Naval Research Laboratory. It is not currently available to the public, however CASTLE has options for using Stella traces as input and output. Starting with CASTLE 3.5, more coupling between Stella and CASTLE is made possible. There are new ways to run CASTLE described below that require specifying a Stella folder to output traces to. CASTLE simulations can now be called from inside Stella or from the command line, or from castle.dll library function calls.

The new DLL version of CASTLE
CASTLE 3.5 includes the new possibility of further Stella integration by using the ".stella" control command 49 at the beginning of a text object. When this control is present, CASTLE netlists can be run from Stella using new "Castle" and "CastleMPI" options for the "Run Plugin" macro routine which calls a new dll version of CASTLE, castle.dll. This allows for automated simulation. The "CastleMPI" version uses MPI (Message Passing Interface) to speed simulation by running code in parallel using multiple threads/cores. CASTLE can also be used with things like Python or Chicago/LSP using castle.dll.

New Stella Control Command
In the regular, GUI form of CASTLE 3.5, using the ".stella" control command loads stella.dll library. This Stella library is then used for things like saving of waveforms to and from the listed Stella shot. Also, all values are first attempted to be evaluated by Stella before the regular evaluation. This allows for more the use of Stella scalar values as object values and also another way to include complex mathematical expressions for values, besides the math parser 43.

Note: The ".STELLA" control command is disabled in the .dll version of CASTLE

Running CASTLE simulations without the GUI
In order to use CASTLE in the non-gui way, you must save your circuit as a netlist. The netlist needs to contain the .tran, .stella, and .plotstella control commands 49. The ".tran" is needed to specify simulation temporal parameters. The ".stella" command is needed to specify location of Stella dll, experiment, and shot. The ".plotstella" is needed to save output waveforms. Running CASTLE from inside of Stella is described above and running from the command line is described next.

Running simulations from the Command Line with MPI (or without MPI)
To run a CASTLE simulation from the command line, open a command prompt and then change directory to the Stella install location (where stella.dll can be found), usually something like C:\Stella64. Next, run mpiexec with number of nodes to use (15 in this example) and stelladllrun as parameters, followed by castle as the dll to run and then Stella shot information and netlist filename. End

 © 2023 Raymond Allen

with ";mv" for MPI mode or ";v" for regular (non-MPI) mode. It can be useful to put this into a .bat file with lines such as this:

```
c:
cd c:\Stella64
mpiexec -n 15 stelladllrun castle;C:
\experiments\MyExperiment;MyShotName;MyCastleNetlistName;mv
pause
```

Using Stella to evaluate expressions and Stella scalar values
As an example of scalar and expression evaluation, consider a transmission-line element with a value text of "Z0=[v(rbeam)\mx], TD=(3+4)*1e-9". When Stella is loaded using the ".stella" control command, both the Z0 and TD expressions are evaluated by Stella. The Z0 value becomes the maximum value of the Stella trace v(rbeam). See the Stella help for "macro expressions" for more info. Note that the brackets are required in this case because of the use of parenthesis in the trace name "v(rbeam)". If the trace name did not include parenthesis then the brackets would not be needed. The value of TD is also evaluated by Stella. If the Stella evaluation were to fail, the regular expression evaluation is used.

The Stella expression evaluator includes these functions:
sin, cos, tan, log, exp, atn, and bol (for boolean, returns 1.0 or 0.0).

and these operators: ^, *, /, E, e, +, -

Parenthesis can be used.

Brackets are needed around operations on trace names that contain parenthesis.
Brackets are also used by some trace operators.

1.5.1 Stella Trace Operators

Summary of special Stella characters and syntax that can be used to get scalar values from a trace.

Use any of the following trace operators to return a scalar value from a trace.
For example: [V(RLoad)\mx] will return the maximum value of a trace with the name, "V(RLoad)".
Note that brackets are required when using trace names that contain parenthesis.

Trace operators:

 © 2023 Raymond Allen

Note: all can be used as inputs, but only tags can be set by outputs
\pt, \is, \it the pretrigger time, the initial shift, and their sum
\st trigger time+initial shift+timeshift - the beginning of the trace
\ft the time of the last point.
\ts the timeshift
\su the scale units
\mx the maximum value
\mn the minimum value
\tm the time of the maximum value
\tn the time of the minimum value
\cl the calibration
\at the attenuation
\tn one of the trace tags, where n = 1,2, or 3
\tp the time per point
\av the average value of the trace
\fh the full width at half maximum
\in the integral of the trace
\pi the peak value of the integral of the trace
\19 the 10-90 risetime of the trace
\28 the 20-80 risetime
\#n the trailing integer in a trace name. For example, VCOR-101\#n will return 101
\#1 the integer before a "-" in the trace name. For example, PIN3-5\n1 will return 3.
Arguments in brackets can themselves be expressions
\t%[nn] the time to reach nn percent of the trace maximum
\b%[nn] the time to last reach nn percent of the trace maximum
\vt[nn] the value of the trace at time nn
\tv[nn] the time when a trace first reaches value nn
\bv[nn] the time when a trace last reaches value nn
\la[nn] the average of the trace from the start up to the time nn.

See the Stella help topic "Summary of special characters and syntax in routine arguments" for more information.

1.6 Notes

A Note about TL Inductors

TL inductors are treated as single timestep long transmission-line elements inside the simulator. But, only two terminals are provided for the schematic symbol. This is because both outer conductors are internally treated as ground. It may be that this could cause an issue in some circuit topology, but no topology has yet been found where this was a problem. But, if you happen to find a problem topology, just use a regular transmission-line element to represent the inductor.

Variable Capacitor

© 2023 Raymond Allen

A variable capacitor is a new addition to the available parts. It can either follow a user supplied polynomial function (as in SPICE) or one or two known capacitor types.

Custom Parts

It is relatively easy to program new, custom parts. Several parts have already been transferred from the transmission-line code, Bertha, into CASTLE. Several are not provided at this time because they are still being tested. Future versions may have the ability for user generated parts based on script files.

Iteration

A few parts, such as the semiconductor diode, require iteration within a time-step. It is possible for such iteration to fail to converge to a stable result before proceeding to the next time step.

© 2023 Raymond Allen

CASTLE Main Window

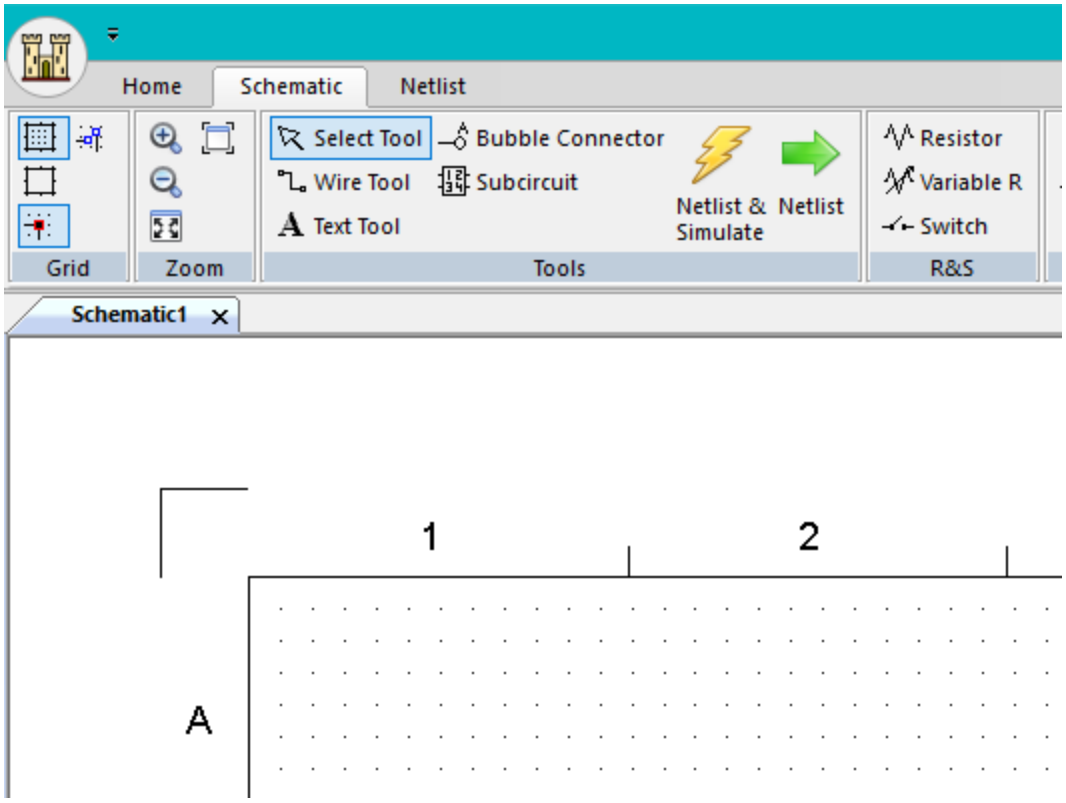
Main Window

2 CASTLE Main Window

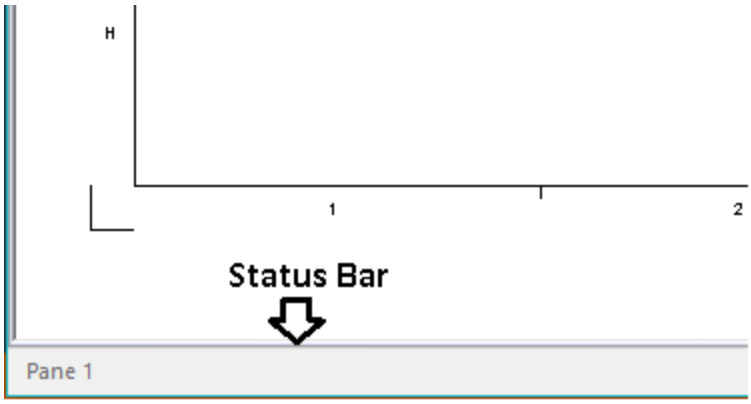
Go to Overview 18 to see a click-able image of the controls.
The Main Window consists of the Ribbon Bar 19 at the top, the Document Pane 21 in the middle and the Status Bar 21 along the bottom.

2.1 Overview

The CASTLE main window consists of three main parts, the Ribbon Bar 19 at the top, the Document Pane 21 in the middle, and the Status Bar on the bottom. The Status Bar 21 will be used in the future to provide basic information about your document. You can click on areas in the images below to identify and jump to specific help areas.

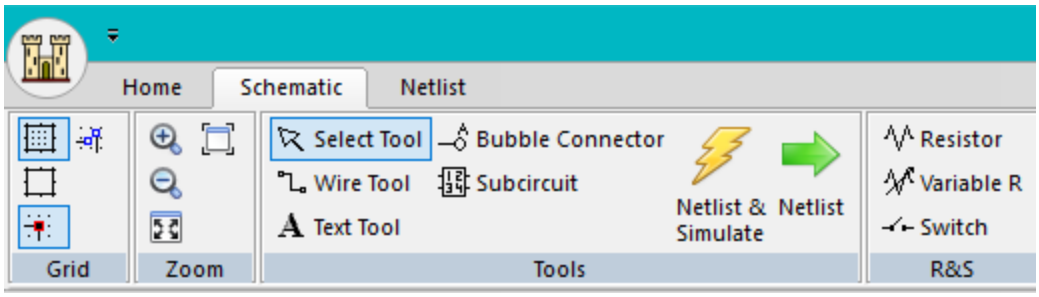


© 2023 Raymond Allen



2.2 Ribbon Bar

Click on the image of the Ribbon Bar below to jump to specific areas (if not already shown in this window).

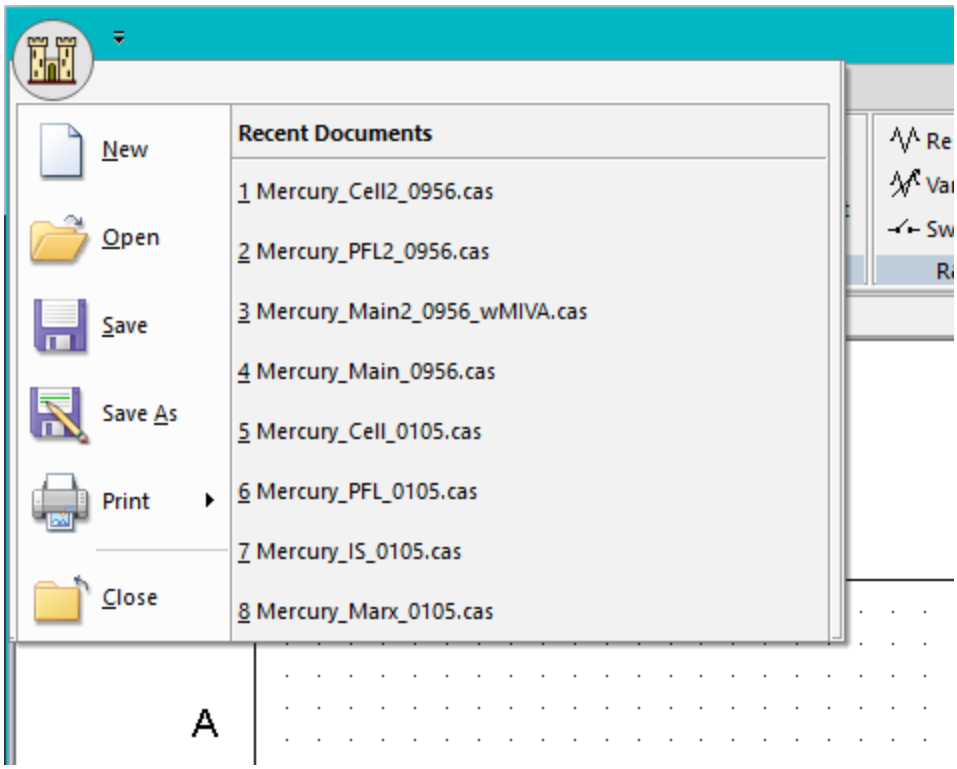


The Ribbon Bar contains all the tools needed to edit, simulate, plot and save your circuit. In the top-left corner is the Application Button 19 and the Quick Access Toolbar 20.

2.2.1 Application Button

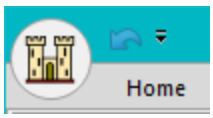
In the top-left corner is the Application button. It's vertical location spans the title bar and the ribbon tabs area. Push the Application Button to bring up a menu for saving, opening and printing the current document. It also brings up the Recent Documents list. To the right of the Application button is the Quick Access Toolbar 20.

© 2023 Raymond Allen



2.2.2 Quick Access Toolbar

The Quick Access Toolbar is located at the top of the CASTLE window, just right of the Application Button. You can customize this toolbar to add often used buttons.



Undo button added to Quick Access Toolbar

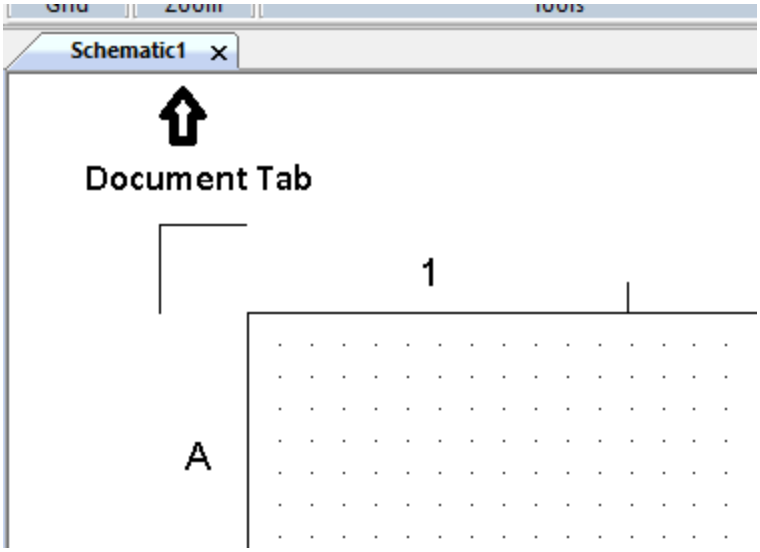
Click the black access menu button to add other buttons.

Location of menu for Quick Access Toolbar

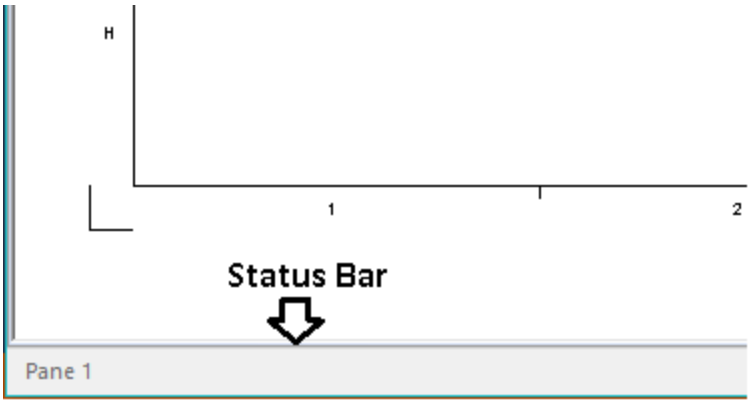
© 2023 Raymond Allen

2.3 Document Pane

The Document Pane is the main window where you edit documents. Multiple documents and be open at the same time. Each open document has it's own Document Tab.



2.4 Status Bar



The Status Bar is located along the bottom of the main window. It is currently unused. It may be used in the future to provide some information while editing documents.

© 2023 Raymond Allen

This page is intentionally left blank.
Remove this text from the manual
template if you want it completely blank.

Schematic Entry

Schematic entry help

3 Schematic Entry

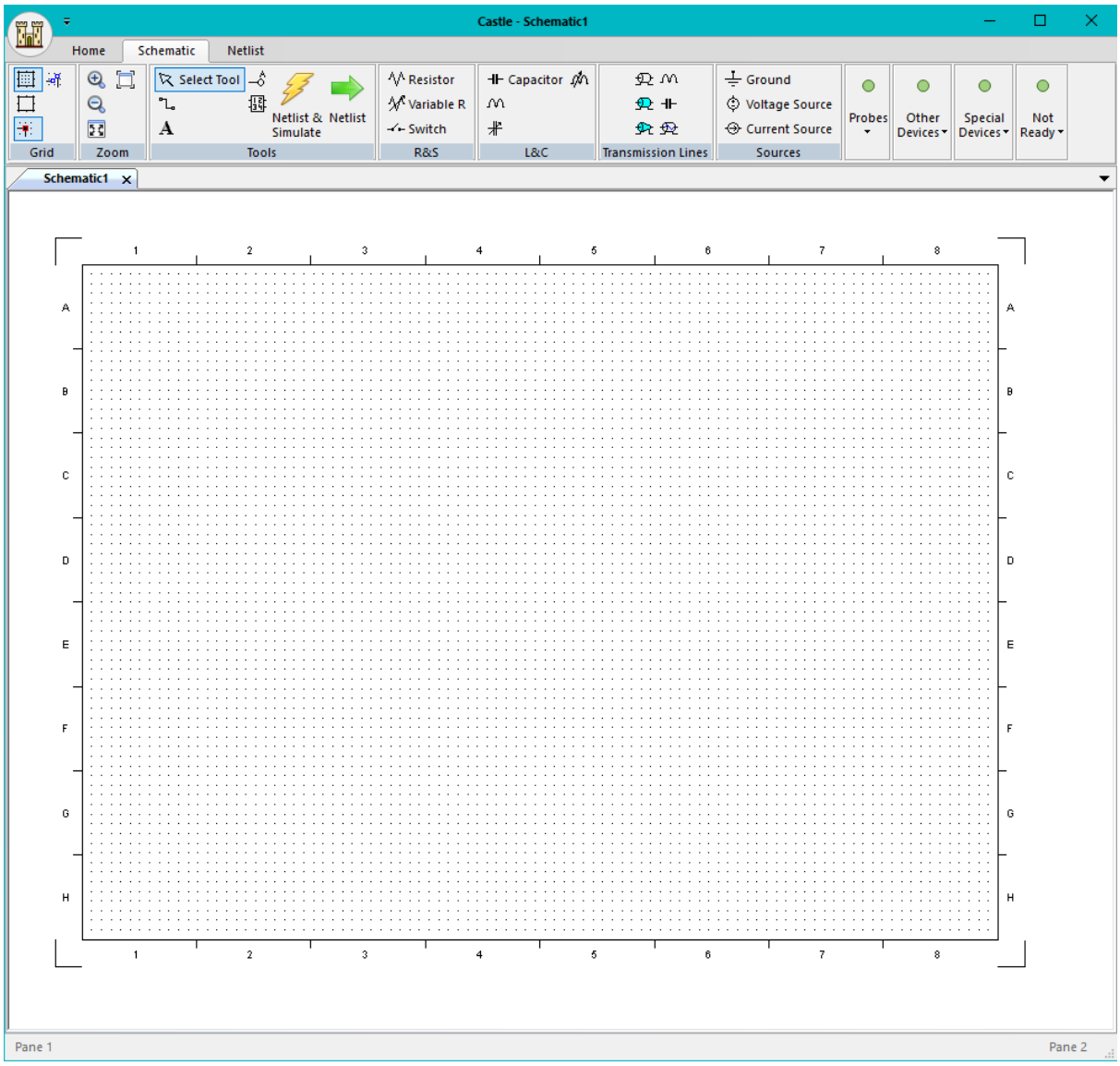
The Schematic Editor Window is the graphical interface where you create your circuit to be simulated. This section provides help for using the the buttons on the Schematic tab of the Ribbon Bar 19 to pick, place and edit parts such as resistors, inductors, capacitors, etc., and on using the Wire Tool to connect parts with wires.

Note: Because CASTLE is an MNA code (like Spice), you most likely need to insert at least one Ground 28 part in your schematic to tell the simulator where ground is. An exception may be a circuit that contains only transmission lines with simple connections, where CASTLE will automatically assign ground connections.

Tips: *See Complete RC Example 29 and TL Circuit Example 36 for a full walk through of schematic entry and simulation. You can add text Parameters 35 to your schematic to more easily change part values. Bubble Connectors 40 and Sub-Circuits 41 can make it easier to simulate large schematics.*

When CASTLE is started, it opens with a blank, **Page Size A** (11"x8.5"), schematic. Enter your schematic using the parts from the "Schematic" tab of the ribbon bar. If you want a larger schematic sheet, go to the "Home" tab of the Ribbon Bar 19 and select the **B or C or D or E page size**. The page size can be changed at any time.

© 2023 Raymond Allen



Screen shot of empty schematic created at program start (Click to expand).

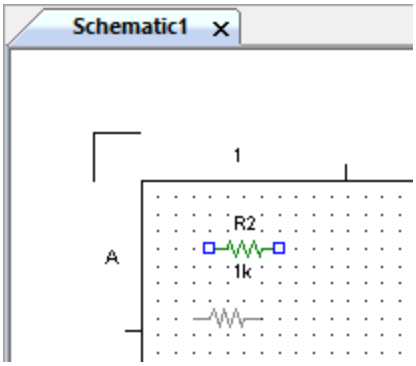
The default file name for the schematic, "Schematic1", appears in the document tab. Use the Application Button 19 to bring up the save button and save your schematic with a different file name.

Note that CASTLE is programmed using Microsoft's standard MFC (Microsoft Foundation Classes) doc/view program architecture and classes so that usage of CASTLE is very similar to other familiar Windows programs. Several schematic windows can be simultaneously opened, each with it's own tab so you can easily switch between them.

Part Placement

© 2023 Raymond Allen

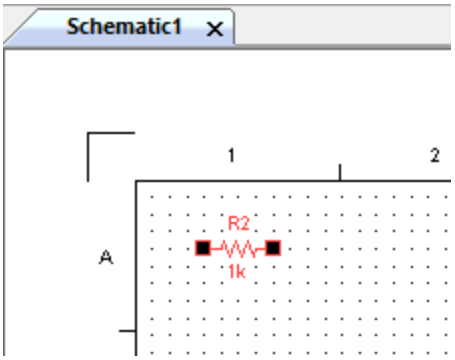
To place a part in the schematic, first click on the desired part in the Ribbon Bar. The mouse cursor will then change from an arrow to the selected part. Move this cursor to the desired position in the schematic and then left-click the mouse to place the part. To exit part placement mode, either press the escape key on the keyboard ("Esc" in top-left corner of keyboard), or select a different option from the ribbon bar (such as the Select Tool).



A resistor (R2) is placed in the schematic by left-clicking the mouse when in part placement mode. Note: Press "Esc" key to exit part placement mode.

Part Selection, Editing, Movement and Rotation

To modify a part, first it must be selected by clicking on it with the part **Select Tool** (located in the Tools section of the Schematic tab of the ribbon bar). When selected the part will turn red. Once a part is selected, you can use the Cut and Copy buttons in the "Home" tab of the ribbon bar. **Double-click** on a part to edit the part properties, such as name, value and initial conditions. See Units 12 for ways to specify a part value.



To edit a part, first select it by clicking on it with the part "Select Tool" and it will turn red.

Most parts can be **rotated** using the "r" key on the keyboard. (Note that parts rotate around their first terminal.). Parts can be **deleted** by pressing the "Delete" key on the keyboard while selected or using the Cut button on the ribbon.

© 2023 Raymond Allen

The select tool can also be used to select a group of parts. To do this, hold the left mouse button down and drag to create a box around the parts to select.

Part Names

Note that CASTLE assigns a default name to new parts. CASTLE uses the first letter of the part name as a code for the part type. Therefore, you cannot change the first letter of a part name.

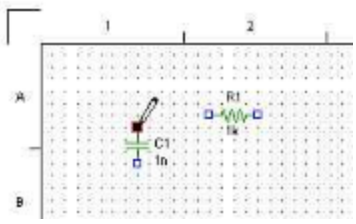
The Schematic Grid

By default placed parts are snapped to a grid and the grid is visible. Buttons in the **Grid** section of the Ribbon Bar can be used to hide the grid and remove the snap to grid feature.

Wire Placement

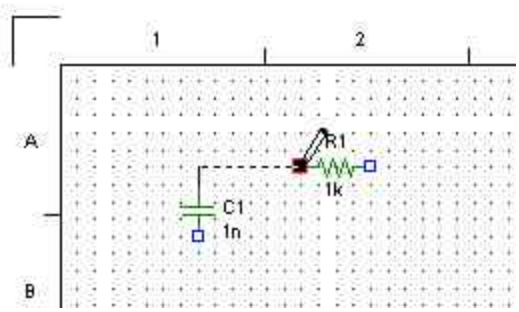
Use the **Wire Drawing Tool** to connect parts with wires. The mouse cursor will change to a pen when in wire drawing mode. CASTLE assists you in connecting parts by changing the **appearance of the part nodes**. Unconnected terminals of a part (or wire) appear as empty blue squares. When the wire drawing tool is over an unconnected terminal, it will change to a black-filled, red square. Left-click on the terminal to begin drawing a wire. You can move the mouse to change the initial direction of the wire. Move the mouse to the terminal you wish to connect to and left-click again. The terminals are now connected by a black wire.

You can make several bends in your connecting wires by left-clicking at intermediate points. Press "Esc" key or a button in the Ribbon Bar to end wire placement mode. Hold the **Shift key** to force intermediate wire points to snap to the grid.



Begin drawing a wire by clicking on a terminal of a part with the Wire Drawing Tool. Note that the appearance of a terminal changes when tool is over the terminal.

© 2023 Raymond Allen



Finish drawing a wire by left-clicking on the node you wish to connect to. Note: The initial direction of a wire can be changed by hovering directly in the desired direction.

Text items can be used for notes and also for control command lines. Text that starts with "params: " (not sensitive to case). There are new control commands, such as ".param", ".plot", ".tran", ".StellaPlot", and ".Stella". Both "params:" and ".param" are ways to set parameters. But, "params:" takes effect while generating the netlist and ".param" is just passed to the netlist as a line of text. See here for a list of commands 49.

See the Simple RC Example 29 for a more complete guide to schematic entry and simulation.

3.1 Keyboard Shortcuts

There are keyboard shortcuts for some functions. Some functions can only be done with the keyboard.

Keyboard functions:

Key	**Function**
Delete	Delete part (same as Cut)
r	Rotate part (only via keyboard)
w	Start wire placement
Esc	Exit current mode and enter Part Select mode
Shift	Forces wire points to snap to grid

3.2 Ground

Because CASTLE is an MNA code (like SPICE), you most likely need to insert at least one Ground part in your schematic to tell the simulator where ground is.

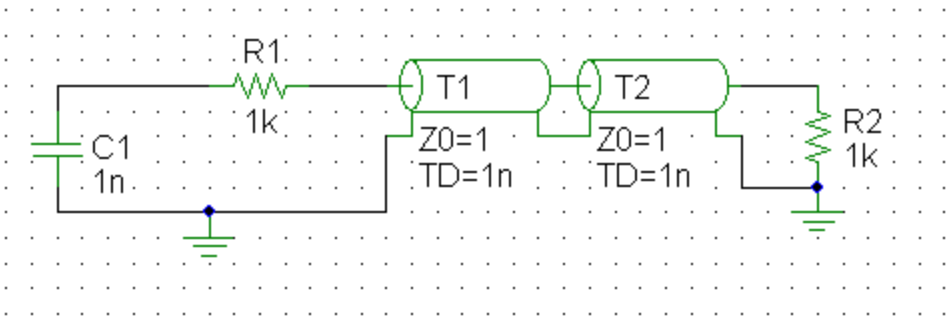
Circuits at either end of a transmission-line are electrically isolated and therefore a ground is needed at each end.

© 2023 Raymond Allen

Note: Grounds are not required when two transmission lines are joined together in a simple way as CASTLE will attempt to automatically assign ground in this case. This feature makes it easier to create schematics of circuits with many transmission line elements.

If you simulate without the required grounds, you will receive an error message that a node is floating.

Example:



Grounds are required in the circuits on the left end of T1 and the right end of T2. However, no ground is required at the junction of T1 and T2. Removing either of the grounds will result in an floating node error message when simulated.

3.3 Simple RC Example

Complete Schematic Entry and Simulation Example: Simple RC circuit

This example shows the basic method of drawing a schematic, simulating, and plotting the results for a simple RC circuit. The first step is to place the capacitor. First, click the capacitor symbol in the Ribbon Bar to switch to capacitor placement mode. At this point a capacitor and a small circle will appear in the schematic window. If you move your mouse around the schematic window the small circle and capacitor will follow the mouse. The small circle represents the mouse position and the left end of the capacitor (the positive end) will also follow the mouse, but be snapped to the grid as shown in Fig.1. Rotate the capacitor 90 degrees by pressing "R" key on your keyboard once so that the capacitor is vertical with the positive terminal on top. Note that all parts rotate about their positive terminal.

© 2023 Raymond Allen

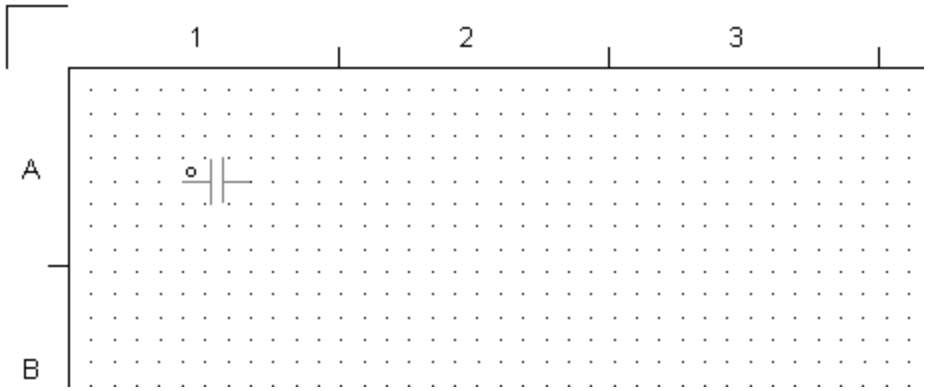


Figure.1. *Partial screenshot of Castle schematic in capacitor placement mode.*

Now, place the capacitor on the schematic by clicking the left mouse button once. Once placed, terminals will appear on the capacitor ends and default name and value labels appear. A new capacitor now follows the mouse for placement. But, since we only need one, exit capacitor placement mode and enter part select mode by either pressing the "Esc" button on your keyboard or selecting the "part select" tool on the parts section of the Ribbon Bar. Now, a standard mouse arrow icon follows the mouse. Use this tool for selecting parts or labels. Select the capacitor by clicking it. The capacitor will turn red when selected. It can now be rotated or moved (by dragging with the mouse). Double-click on the capacitor to bring up the edit dialog box. Change the value to 100n and change the IC (initial condition) to "IC=1". Click the "Visible" checkbox so that the IC will be displayed on the schematic and then push "OK". Now, your schematic should appear like that in Fig. 2. Note that all part values and units use the convention of SPICE (i.e., 100n is 100e+9, 1m = 1e-3, 1MEG=1e+6; capacitance in F, inductance in H, resistance in Ohms). (See Units 12) Note also that all fields are case insensitive in CASTLE.

Next, place a resistor to the right of the capacitor by selecting the place resistor tool from the Ribbon Bar and clicking on the schematic, just the same way as we did with the capacitor. Double-click the resistor and change it's value to 2. Your schematic should now appear similar to that in Fig. 3.

© 2023 Raymond Allen

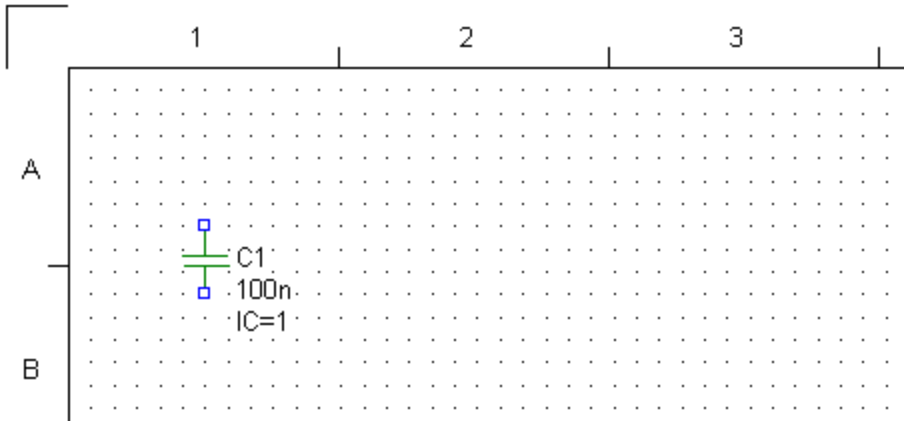


Figure 2. *Partial screenshot showing placed capacitor with value set to 100 nF and IC set to 1 V and made visible.*

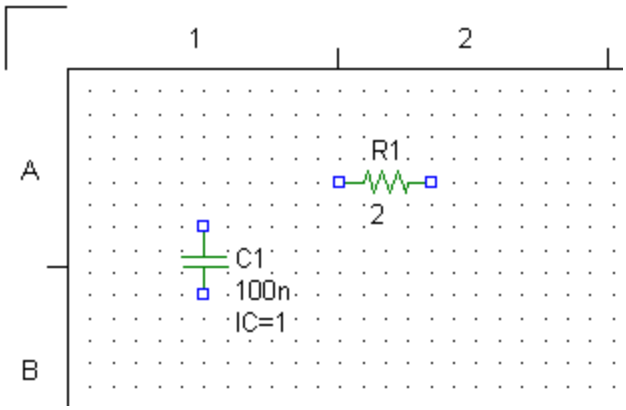


Figure 3. *Partial screenshot showing placed resistor with value set to 2 Ohms.*

Next, we need to add two wires to connect the components. Switch to wiring mode by selecting the wiring tool from the parts toolbar (fourth button from the left) or by pressing the "W" key on your keyboard. A pencil icon will now follow the mouse. Move this pencil tip to the top terminal of the capacitor until that terminal is highlighted (changes color and becomes filled in black). Now, move the mouse vertically and then to the right toward the resistor's left terminal. You should see a solid vertical wire extending from the capacitor's top terminal and a dashed horizontal wire to the current mouse position. Note that you can escape from wiring mode by selecting the part select tool (or any other tool). Move the mouse to highlight the left terminal of the resistor and then click the mouse to complete the wire. The schematic should now appear as in Fig. 4. Note that wired terminals are hidden. If a wire end does not connect to a part, a terminal appears at the end of the wire. You can delete a wire segment (or a part) by selecting them with the part select tool and pressing the "Delete" key on your keyboard. You can select multiple parts by dragging a box around several parts while in part select mode. Selected parts appear in red.

© 2023 Raymond Allen

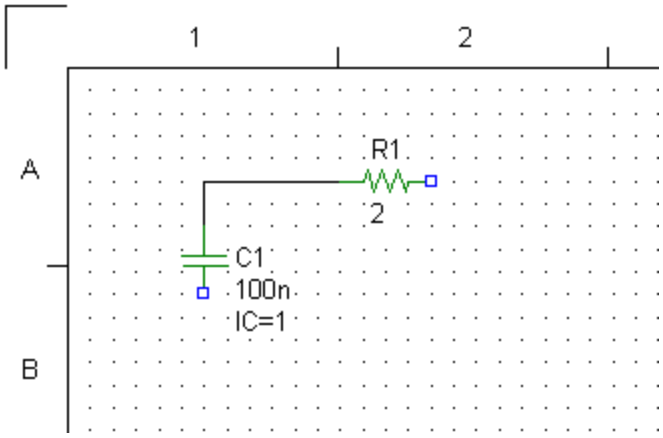


Figure 4. *Partial screenshot showing wire connecting C1 and R1.*

Next, draw a wire from the bottom terminal of the capacitor to the right terminal of the resistor using the same technique. Note that part labels can be moved by selecting them with the part select tool and then dragging them to the new location.

The final step is to place the ground node. Select the ground tool from the parts Ribbon Bar and move the mouse to the bottom terminal of the capacitor. Once over this terminal, the terminal will be highlighted. Click the left mouse button to place the ground at the bottom capacitor terminal. The ground is then placed and a node symbol appears at the junction of the capacitor terminal, wire to the resistor, and the ground's terminal. Node symbols mark the junction of more than two terminals. Your schematic should now appear as in Fig. 5. Note that the ground terminal could also have been rotated and/or placed on the resistor's right end or the wire between resistor and capacitor with the same affect. (Note: Ground terminals are not required at simple connections between transmission-line elements)

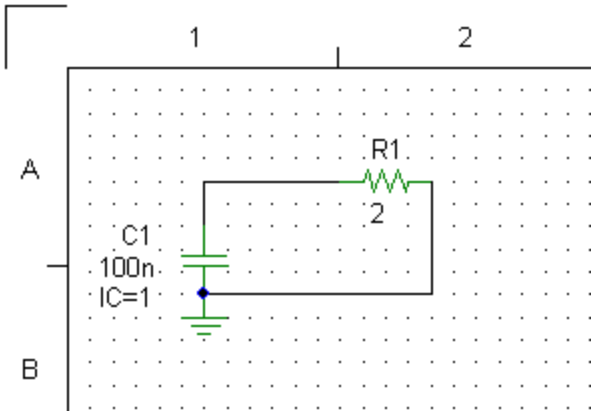


Figure 5. *Partial screenshot showing completed RC schematic.*

© 2023 Raymond Allen

Netlist Generation

The schematic entry of this simple RC circuit is now complete. Before simulating you may examine the automatically generated netlist by clicking the "Netlist" button in the Ribbon Bar. The result should appear as in Fig. 6. This netlist is very similar to that used in SPICE (in fact, simple netlists are usable without modification in SPICE). This circuit contains two nodes, "N5" and "0". "0" represents the ground node and "N5" is an automated name for the connection from the positive terminals of capacitor and resistor. Note that the positive terminal is listed first, followed by negative terminal, then part value in the netlist. Note that CASTLE internally treats capacitors as single timestep long transmission-line elements with center and outer conductors tied together to provide two terminals.

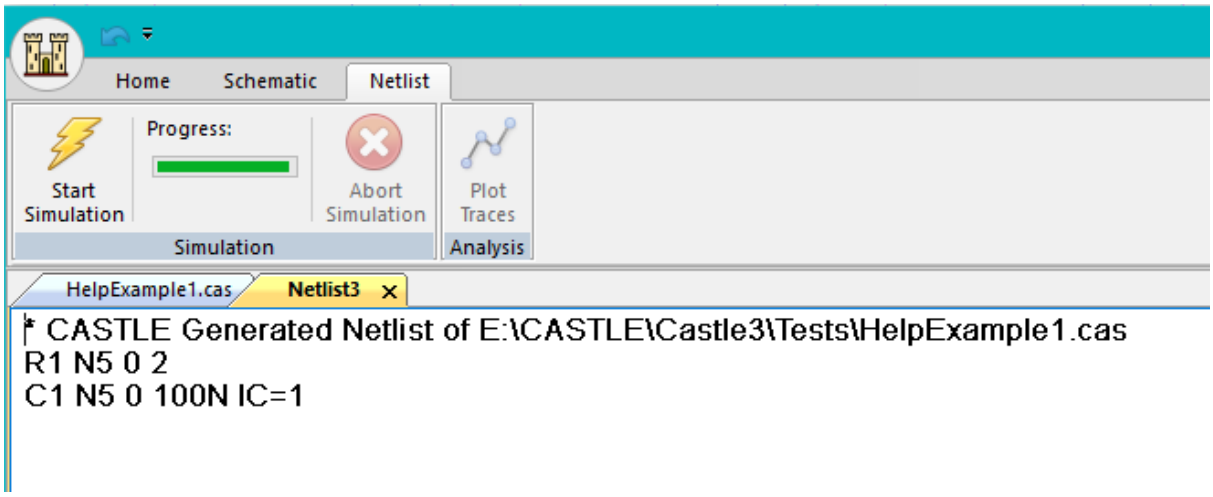


***Figure 6**. Netlist for simple RC circuit.*

Before simulating, it is advised to save your schematic. Do this by selecting "Save" from the Application Button 19 popup menu. By default, CASTLE schematics have a ".cas" filename extension.

Simulate this simple RC circuit by pushing the "Start Simulation" button the simulator toolbar. The simulator dialog box appears where you can select the time step, time span and start time (note that end time = start time + time span).

Change the "Time Span" to "1000NS" and push the "Simulate" button. You should then quickly see a "Simulation completed" message. When you click "OK", the "Select Traces" dialog automatically appears. Here you select traces to plot or export.

© 2023 Raymond Allen

The available traces are broken down into several categories in the top left list box, "Trace Type:". Select "Part Voltages" and "V(C1)" and "V(R1)" should appear in the top right list box, "Trace List", where they can be selected by double-clicking. Select both V(C1) and it should be added the "Selected Traces" text box. Now, click the "Plot Traces" button to see a plot of V(C1) versus time. You can zoom into a portion of the plot by holding down the "Shift" button and dragging a box over the portion to zoom to. Note that the regular zoom in and out buttons also work, but they scale the plot without changing the range of data displayed.

You can close this plot by selecting "Close" from the "File" menu or by clicking the smaller "x" in the top right corner inside the main window. Do not push the big "X" of the main window or you'll close CASTLE! You could also just leave the plot and return to the schematic by selecting the schematic window from the window list in the "Window" menu. Windows can also be cascaded or tiled.

TIP: You can make further plots or export data without simulating again. Just return to the schematic and push the "Trace Select" button, which becomes usable only after the schematic has been simulated. All simulation data remains in memory until the schematic is closed.

Adding Probes to the Schematic

Voltage and current probes are not really needed as all voltages and currents can be viewed after a simulation. However, you can add explicit current and voltage probes, if desired. In the simple RC schematic previously described, we can add a voltage probe to positive terminal of the capacitor by pushing the "Voltage Probe" part button on the part toolbar (10th from the left) and then placing the probe, e.g., on the top-left bend in the wire that connects C1 and R1. Note that this bend should be highlighted when placing to ensure proper connection. The schematic should now look as shown in Fig. 7.

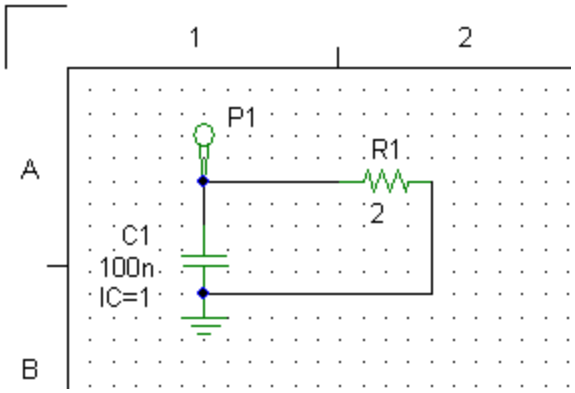


Figure 7. *Voltage probe added to simple RC circuit schematic.*

Now, you can see in the netlist that the node name for the positive ends of C1 and R1 has been renamed to "P1", the probe name. Also, after simulating, you

© 2023 Raymond Allen

will see that "V(P1)" is automatically added to the "Selected Traces" box, allowing for quick plotting or saving.

We can also add a current probe to the circuit. Select and delete the wire going downward from the right end of R1. Switch to current probe placement mode by clicking the current probe button in the part toolbar (third from the right) rotate the probe, if desired, by pressing "R" once. Then place the probe where the wire was. The probe will be labeled "V1". Note that in CASTLE (as well as SPICE) current probes are implemented as voltage sources with their voltage set to zero.

Use wires to connect the probe to the resistor and existing wire by entering wiring mode (press "W") and drawing wires between terminals. The schematic should now look similar to Fig. 8. After simulating, "I(V1)" will be added automatically to the "Selected Traces" box.

TIP: Remember that you can use Undo/Redo to correct mistakes.

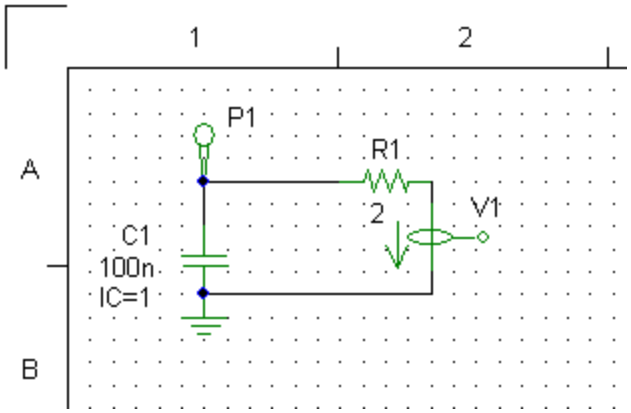


Figure 8. *Current probe added to simple RC circuit schematic.*

Adding Text Labels to the Schematic

The text tool allows you to place text labels on the schematic. Note that normal text will not affect the netlist. When you select the text tool a dialog box will prompt you to enter the desired text. You can then place this text in one or more places on the schematic. Edit text labels by double-clicking them.

Using Text Parameters

Text labels that begin with the keyword "PARAMS:" are utilized to allow easy setting of frequently changed part values or parameters 40. For example, with the simple RC schematic, change the value of the resistor from "2" to "RLOAD". Next, create and place a text label with the text "PARAMS: RLOAD=2". You can check the netlist to verify that R1 is assigned the value of 2. An example is shown in Fig. 9. Parameters are also used to pass values to subcircuits,

 © 2023 Raymond Allen

described later. Note that the SPICE ".param" control command 49 is now also supported. The main difference is that "PARAMS:" are evaluated when generating the netlist, while ".param" lines just get added to the netlist and are evaluated when simulation begins.

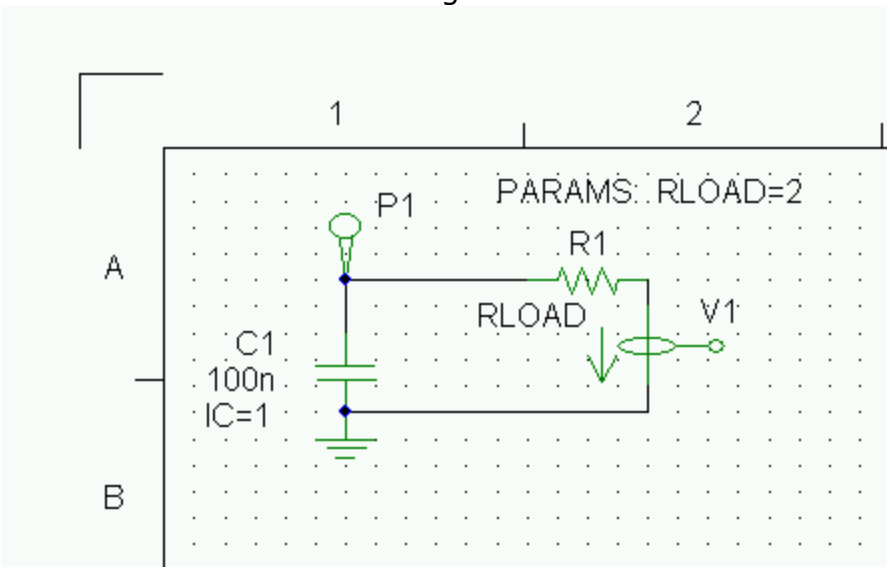


Figure 9. *Example of using text labels to set simulation parameters.*

3.4 TL Circuit Example

Transmission-Line Circuit Example: Simple Pulsed-Power Circuit

This example demonstrates how to use transmission-line elements in CASTLE.

Place three transmission-line elements on the schematic using by pushing the transmission-line placement button on the Ribbon Bar and then clicking the mouse on the schematic. Place the elements as shown in Fig. 11. Note that transmission-line elements can be rotated (using the "R" key on the keyboard) and will rotate about the left, positive (center) terminal. But, place these elements in the default orientation for this example.

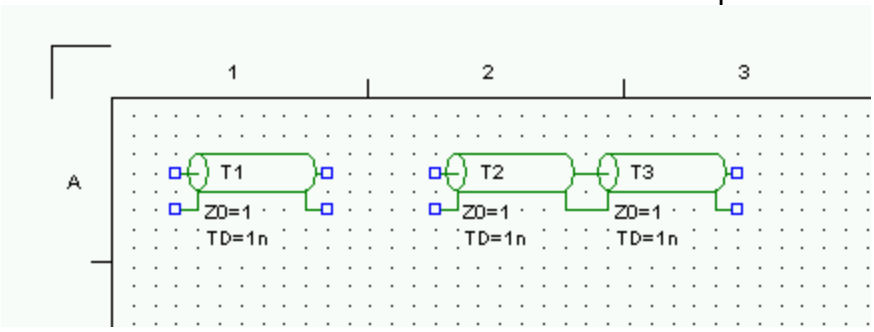


Figure 11*. Three transmission-line elements placed on the schematic.*

© 2023 Raymond Allen

Notice that the CASTLE automatically recognizes that T2 and T3 are joined and the terminals at the junction are hidden. It is not required to place a ground node at this junction.

Double-click the elements to edit their values. Change T1 to Z0= 3, TD= 25n, IC= 4MEG, and make the IC visible. Change T2 to Z0= 3, TD= 10n. Change T3 to Z0= 1.5, TD= 50n.

Place a switch element between the center terminals of T1 and T2. (the switch placement mode button is fourth from the right on the parts toolbar). Then, place a resistor at the end of T3 (resistor part button is the left button). If you want, rotate the resistor to the vertical position using the "R" button on the keyboard either before placing or after by first selecting the resistor with a mouse click. Change the resistor value to 2. You can view the switch settings by double-clicking on it. But, this example will use the default settings.

Use wires to connect center terminals of T1 and T2 to the switch ends and to connect the outer terminals of T1 and T2 to each other. Use wires to connect T3 to the resistor. The schematic should now look like Fig. 12. Note that no connection is required on the left end of T1 (Although unconnected end

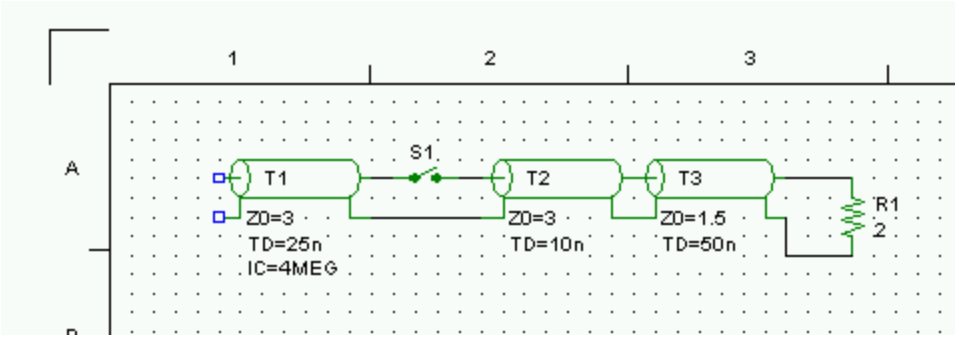


Figure 12. *Basic transmission-line circuit wired with switch and resistor.*

Explicit grounding is required for some transmission-line junctions, but CASTLE automatically assigns ground for several types of junctions. (If desired, push the "Netlist" button to examine how ground nodes are assigned). The left end of T1 is open and CASTLE will assign ground to the left outer conductor terminal. The outer conductors of T1 and T2 are connected and neither of the center conductors attached to S1 are explicitly grounded, so CASTLE automatically assigns ground to the connected outer conductors. T2 and T3 are directly connected, so CASTLE automatically assigns ground to the connected outer conductors.

T3, however, is connected only to a parallel resistance. At this time, this type of connection requires an explicit ground. If you push the "Simulate" button at this

© 2023 Raymond Allen

time, you will get an error message that a node is floating. If you then were to examine the netlist, you will see that this is one of the nodes connecting T3 and R1.
Add an explicit ground terminal to the bottom terminal of R1 using the ground tool (8th button from the left on the parts toolbar). The completed schematic is shown in Fig. 13.

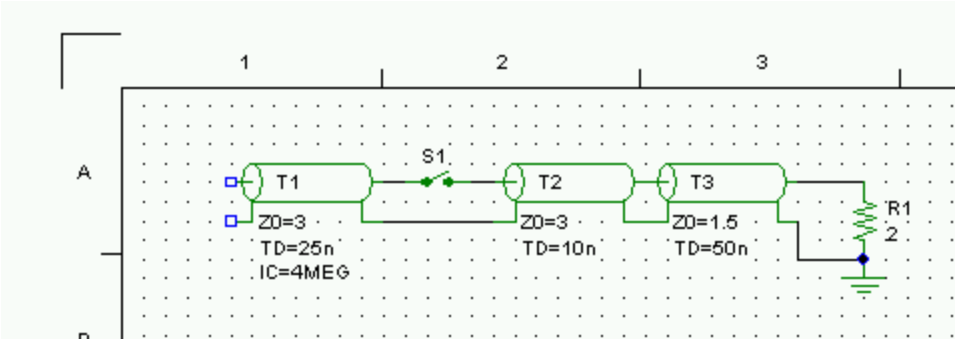


Figure 13. *Completed schematic of basic transmission-line circuit.*

You can now simulate this circuit by pressing the "Start Simulation" button on the Ribbon Bar. Change the time span to "200n" and begin the simulation by pushing the "Start Simulation" button in the simulation dialog box.

TIP: These simple simulations will finish instantly. However, you may have more complicated simulations that can take several minutes and possibly make the computer less responsive. If this happens and you wish to stop the simulation before it is complete, push the "Abort Simulation" button in the Ribbon Bar. It may take a second to take affect, but this will end the simulation early. Note that all simulations results are still available for the completed portion. A "Progress" bar in the Ribbon Bar indicates how much of the simulation has been completed.

Once the simulation is complete you push the "Plot Traces" button in the Ribbon Bar and select which traces to plot in the trace select dialog that appears. For example, to select the current in R1 click on "Part Currents" in the left list and then "I(R1)" in the right list. You can also select transmission-line voltage, current, or waves in the left list. The last letter "R" or "L" in the trace names in the right list indicates the right or left end location of the measurement. For example, to plot the voltage at the left end of T1, select "VL(T1)". There are two waves at each end of a transmission-line, V1 and V2. V1 refers to the wave leaving an end and V2 refers to the wave entering an end. For example, to select the wave entering the right end of T3, double-click on "V2R(T3)".

Note: *One advantage of CASTLE over most TL codes is the unrestricted topology. In this example, you can simply connect a capacitor across S1, if desired. You could also add resistors from either end of S1 to ground. Just be*

© 2023 Raymond Allen

aware that in more complicated junctions you may have to specify the ground node.

© 2023 Raymond Allen

3.5 Parameters

There are two ways of adding parameters to simulations, as previously discussed in the Simple RC Example 29.

Text labels that begin with the keyword "PARAMS:" are utilized to allow easy setting of frequently changed part values or parameters. For example, to make the value of a resistor a parameter, give the value a name, such as "RLoad", instead of a numeric value. Next, create and place a text label with the text "PARAMS: Rload=2". Now, when a netlist is generated, all instance of the value "RLoad" will be replaced by the text value of "2". Parameter names are case insensitive.

Multiple parameters can be placed on one line by using spaces between them, for example, "PARAMS: Rload=2 VCharge=5MEG"

The SPICE ".param" control command 49 is now also supported. The main difference is that "PARAMS:" are evaluated when generating the netlist, while ".param" lines just get added to the netlist and are evaluated when simulation begins.

Both "PARAMS:" and ".param" lines can use curly braces to define expressions to be evaluated by the Math Parser 43. This allows for things like "PARAMS: Rload={2*1.1}"

Note that if a ".Stella" control command 49 is present, all expressions are first attempted to be evaluated by Stella and can therefore include Stella scalars and Stella expressions.

3.6 Bubbles and Sub-Circuits

Using Bubble Connectors

For large schematics, you may want to use bubble connectors instead of drawing long wires across the schematic. Bubble connectors are also used sub-circuits (described later). In the previous example, one could separate the circuit in two parts at the T2-T3 junction using bubble connectors. To demonstrate, first select the right part of the circuit, including T3 and R1 and drag them to the right a few grid points.

TIP: You can drag multiple components by selecting the all and the dragging them (with the left mouse button held down). You can select multiple components by either dragging a box around them while in part select mode and/or by clicking on them with the "Ctrl" key held down. Selected parts will appear in red. You can also copy, cut, and paste selected components.

© 2023 Raymond Allen

Next, add bubble connectors to the center connectors of the right side of T2 and the left side of T3 by pushing the bubble placement mode button on the Ribbon Bar and clicking on the desired terminals once they are highlighted. Note that bubble connectors can also be rotated using the "R" key on the keyboard. Ensure that the two bubble connectors have matching labels, such as "A", so that CASTLE will recognize them as being connected. Do not use "sub1", "sub2", "sub3", or "sub4" for bubble labels as these are reserved for sub-circuit pin designations. The schematic should now look like that of Fig. 14. If you view the netlist you will notice that the only change is that the common node of T2 and T3 center conductors is now called "A", same as the bubble label. If you simulate this example you will get the same result as before.

Note that grounding is not required for the outer conductors of T2 and T3 because they have a simple connection. For a more complicated connection, either explicit grounding or bubbles must be used. Although not required, it is best practice to add grounds to outer conductors of T2 and T3. This is so that one can recognize blue squares as suspect entities in the schematic.

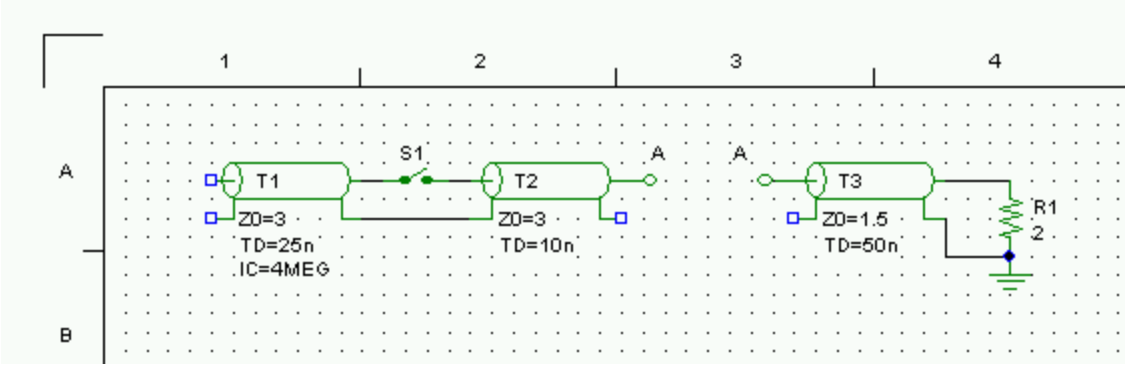


***Figure 14**. Transmission-line circuit example using bubbles to connect isolated segments.*

Sub-Circuits

Using subcircuits, you can represent a portion of one schematic with another schematic. This can be very useful when several instances of the same subcircuit in a main schematic. As an example, the same simple transmission-line example from the previous example will be split into a main schematic and a subcircuit schematic.

First, selected the entire right side of the circuit including T3, R1, and the right bubble connector. (You may want to save the current file under a different name at this point.) Click "Cut" from the "Edit" menu or push the scissors icon on the main toolbar to copy this subcircuit to the clipboard and remove it from this schematic. Next, select "New" from the "File" menu and create a new schematic. Paste the previously cut subcircuit into the new schematic. Rename the bubble connector from "A" to "sub1" (sub1, sub2, sub3, and sub4 are

© 2023 Raymond Allen

reserved keywords). The new subcircuit should look like that in Fig. 15. Note that the part labels may change when pasting them (T3 becomes T1 in the new schematic). This is to avoid duplicate part names within the same schematic. Just rename parts to their previous values. This in not really required though because subcircuit part labels can be the same as the main circuit without problems. Now, save this file with a new name and return to the main schematic (use the "Windows" menu to switch between schematics).

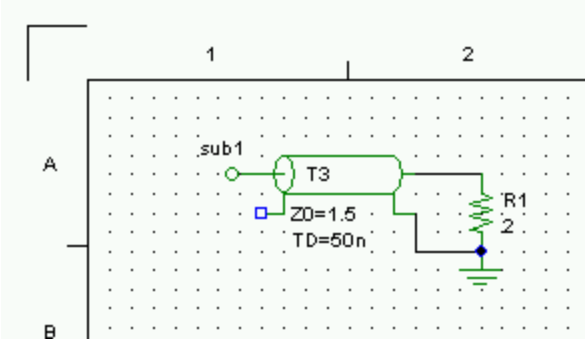


Figure 15. *An example subcircuit schematic.*

Next, delete the bubble on the main schematic and replace it with a subcircuit element (the subcircuit mode tool button is second from the right on the parts toolbar). Place the subcircuit element just to the right of T2, but leave a space between terminals. Then, draw a wire from the left, center terminal of T2 to the "1" pin of the subcircuit element, "X1". The main circuit should now appear as in Fig. 16. Note that you could have connected up the outer conductor terminal to pin 2, but this is not necessary since the ground node is common to all schematics.

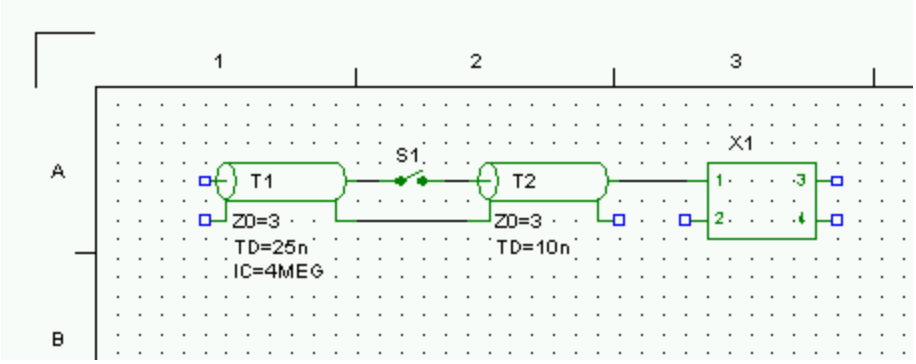


Figure 16. *An example circuit utilizing a sub-circuit element.*

Finally, you must tell the sub-circuit element which schematic file contains the sub-circuit it represents. Double-click the sub-circuit element to bring up the settings dialog. Click the "Browse" button and find the file containing the sub-circuit previously saved. Note from this dialog you can examine the sub-circuit by clicking the "Open" button and then hide it again with the "Close" button.

© 2023 Raymond Allen

One can also pass parameters to the sub-circuit. This is useful when there are several instances of the same subcircuit but with some different element values. In this example, you can enter "Rload=2" in the "Parameters" text box in the sub-circuit element edit dialog and then change the resistance of the R1 in the sub-circuit from "2" to "Rload". Note that you must save the sub-circuit schematic for changes to take affect in the main circuit.

If you new "View Netlist" in the main schematic you can verify that the parameter value was passed. Each sub-circuit element will have it's own section in the netlist and it's node names with have an underscore and then the sub-circuit label as a suffix. If you then simulate this circuit and begin selecting traces, be aware that the part names for those parts in a sub-circuit will also have their labels appended by an underscore and the sub-circuit label. This is to ensure unique names when several instances of a sub-circuit are present.

Note that this example only uses 1 of 4 available nodes for the sub-circuit. And, since the ground node is common, the sub-circuit can actually share 5 nodes with the main circuit.

TIP: It is possible to have sub-circuits within sub-circuits. Also, you can pass parameters from the main circuit to these sub-sub-circuits. This can be achieved, e.g., by passing a parameter like "RLoad2=Rload" from the sub-circuit to the sub-sub-circuit and also passing "Rload=2" from the main circuit to the sub-circuit. In a similar way, you can use text parameter labels in the main schematic, e.g., "PARAMS: Rload=2", to pass parameters to sub-circuits.

3.7 Control Commands

Control commands can be provided to the simulator using text or formatted text objects.
Any text that begins with a period, such as ".tran", will be added as a line in the netlist.

See here for a list of commands 49.

3.8 Math Parser

Text inside curly braces in a "PARAMS:" line in the schematic or ".PARAM" line in a netlist is evaluated by a math parser.

The math parser can use scientific notation for constants using "E" or "e", but not things like "n" for nano.

For example "PARAMS: tl1={5e+4/3.1}" is valid.

© 2023 Raymond Allen

Note: If Stella is loaded, you can skip the curly braces and Stella will evaluate the expressions instead.

The math parser is aware of the following operators and functions:

Operators:

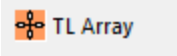
+	Addition
-	Subtraction and unary minus
*	Multiplication
/	Division
^	Power
%	Modulo
&	Logical AND
|	Logical OR
!	Logical NOT
>, >=	Greater or equal
<, <=	Smaller or equal
!=	Not equal
==	Equal

General functions:

abs	fact	rand()
acos	floor	rand(min, max)
asin	hex	round
atan	if	sin
avg(x,y,z,...)	isNaN	sinh
bin	log	sqrt
ceil	log10	sum(x,y,z,...)
cos	max(x,y,z,...)	tan
cosh	min(x,y,z,...)	tanh

3.9 TL Array

A large array of transmission lines can now be generated with a new TL Array element.



© 2023 Raymond Allen

T1

ARRAY CLF LEFT RIGHT TOP BOTTOM YES N=5 M=1
Z0=1 TD=1n

This element generates an array of identical transmission lines in the netlist, when the netlist is generated. There are two types of array configurations available, Chain Link Fence and Rectangular Grid. The Rectangular Grid type of array has been used many times in the past, note-ably by L3. For this type of array, one usually divides the time delay of each element by square root of two and also multiplies the impedance by the square root of two. This is done to make the capacitance of the array equal to what it would be as a single element. The Chain Link Fence is a new type of array being developed by Bruce Weber (NRL). This type of array does not require modification of the element values and also has the advantage that the impulse response of the array is equal to that of a single element. This is not the case for the Rectangular Grid, where a pulse sent through the array gets distorted when compared to a single element representation.

© 2023 Raymond Allen

This page is intentionally left blank.
Remove this text from the manual
template if you want it completely blank.

Netlist Editor & Simulator

4 Netlist Editor & Simulator

The Netlist Editor Window is created from the Schematic Editor Window 24 when either the "Netlist and Simulate" or the "Netlist" button is pushed on the Ribbon Bar 19. This window shows the text based "Netlist" created from the graphical schematic. The Netlist contains one line for every component in the circuit, along with parameters of the component and to which nodes it is connected to. The Netlist is the input to the Simulator. Although usually not necessary, it is sometimes useful to review the Netlist prior to simulation to make sure your schematic was interpreted correctly.

Note: Netlists are saved by CASTLE as Unicode (UTF-32) without the BOM word. Prior to Jan 2023, CASTLE could not open netlist files that were edited with an external editor that added the Unicode BOM.

Once you have a netlist window open, you can use the "Start Simulation" button to begin simulation.
If the .tran control command 43 is not present, the Simulate Window 50 will appear with simulation options.

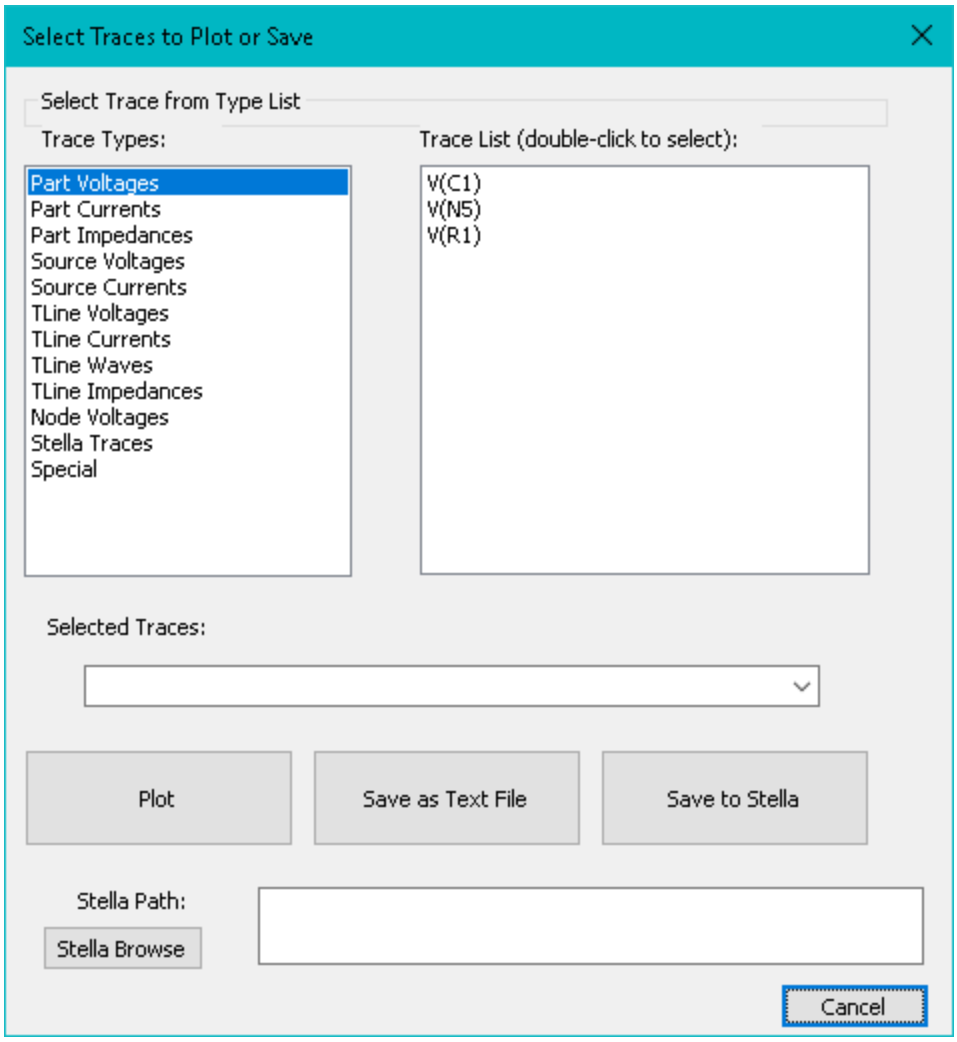
See Netlist Generation 33 in the Simple RC Circuit Example for some notes on the Netlist.

After simulation, the Plot Traces 48 dialog window will appear for Plotting and Saving simulation results. Later, you can use the "Plot Traces" button on the Ribbon Bar to reactive this window for a netlist that has been simulated. It is also possible to use the .plot control command 43 to automatically plot traces after simulation.

4.1 Plot Traces Window

Automatically after simulation, or by pressing the "Plot/Save Traces" button on the Ribbon Bar, the Plot/Save Traces dialog window will appear. Select a type of trace from the left list window and all the available traces of that type will appear in the right list window. Double click on a trace in this Trace List to have it automatically added to the "Selected Traces" line. You can also manually enter the name of the traces you wish to plot in the Selected Traces Line (with a space between names). Recently plotted traces can be selected with the drop list access at the right. Next, you can Plot the Selected Traces with the Plot button or save them as text files or Stella 13 traces with the other buttons. Use the Stella Browse button to select where Stella traces will be saved.

© 2023 Raymond Allen



The Plot Traces dialog window.

4.2 Control Commands Lines

Lines of text in the netlist that begin with a period, such as ".tran 100n", can be used to control aspects of the simulation and also plotting and saving of results. These lines can come from the schematic as noted here 43.

The following commands are recognized:

.TRAN is used to specify the time span, time step, and start time of the simulation. Very similar to SPICE syntax except have time span instead of end time.
If a .TRAN statement is found, the simulation control dialog will not be shown.
".TRAN x" with one argument will use x for the simulation time span. Time step is assumed 0.1N and start time is assumed 0.

© 2023 Raymond Allen

".TRAN y x" with two arguments will use y for time step and x for the simulation time span. Start time is assumed 0.
".TRAN y x z" with three arguments will use y for time step, x for the simulation time span and z for start time of simulation.

.PLOT is used to automatically plot waveforms at the end of the simulation. Each .PLOT statement will create a new plot window.
If a .PLOT statement is found, the plot select window will not automatically appear at the end of the simulation.

.PATH can be used to specify the path to any input files needed for the simulation.

.PARAM is used to set parameters in the netlist. These lines will appear in the netlist as entered.
Note: The simulator will evaluate parameters given as equations inside curly braces when the simulation starts.
Example: .param LThis=2E-9 CThat={5e-9/45} Zthere={60log(5/1)}
See here for more info on the math parser[43].

.STELLA invokes Stella for waveform interactions. The format is like ".Stella C:\Stella64 E:\Experiments\ECSE New_LRU".
Where the second parameter is the Stella install location, third parameter is path to experiment, and fourth parameter is the shot name.

.PLOTSTELLA has a format like .PLOT, but the files are saved to Stella, instead of plotted.

.PLOTTEXT also has a format like .PLOT, but plot data is saved to the text file, CASTLE.out, in same folder as castle.exe or castle.dll that invoked it.

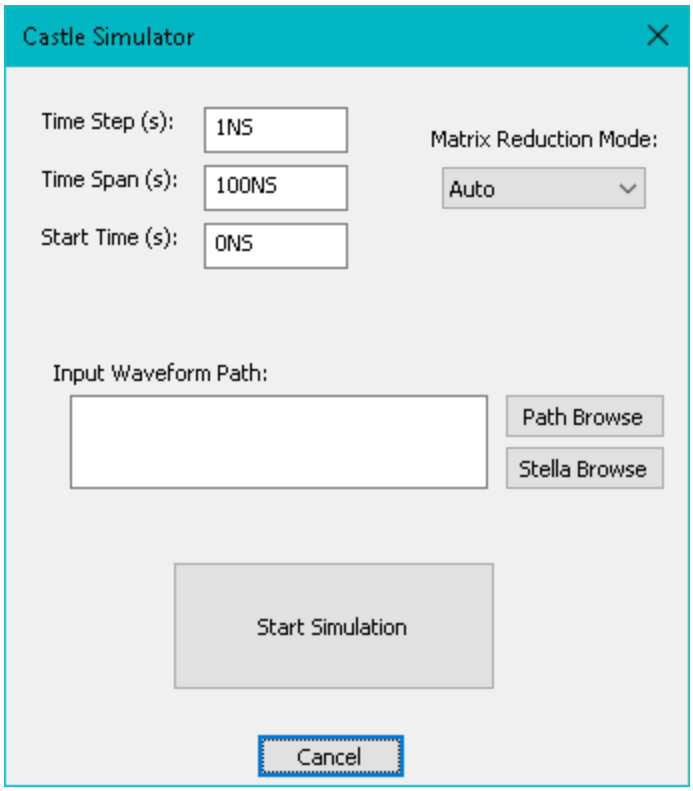
.INC filename allows you to include extra lines from file, "filename", into the netlist when simulating. Can be useful if one want to have simulation parameters in a separate file.
Note: Do not include quotes in your filename, even if it has spaces in it.

4.3 Simulate Window

The "Simulate" dialog window appears when you click on "Netlist and Simulate" button from a schematic or when you click on "Simulate" button from a Netlist. Here you can specify the parameters of the simulation, such as time step, time span and start time. You can also specify the Matrix Reduction Mode used by CASTLE. If your simulation uses text or Stella traces as inputs, you can specify the input path. Finally, press "Start Simulation" to begin.

Note: If a .tran statement is in the schematic or netlist, this dialog will not be shown.

© 2023 Raymond Allen



The Simulate Dialog Window

Matrix Reduction Mode

CASTLE can greatly increase simulation speed by breaking the main matrix to be solved many smaller matrices. This is done using the transit-time isolation properties of transmission lines. There are four options:

1. **Auto.** CASTLE will automatically determine the best approach.
2. **Maximal**. CASTLE will reduce the matrix as much as possible.
3. **Minimal.** CASTLE will reduce the matrix into two smaller matrices, one that includes all the time varying impedance elements and another with all the other components.
4. **None.** CASTLE will use a single matrix for all components.

4.4 Plot Select Window

After simulation, you can use the "Plot Traces" button on the "Netlist" tab to get the plot select window 51 where you can pick traces to plot. Data from all nodes in the simulation is saved and available.

© 2023 Raymond Allen

This page is intentionally left blank.
Remove this text from the manual
template if you want it completely blank.

Plot Windows

5 Plot Windows

Plot windows are generated from the plot select window 51 or by including a ".plot" control command 49.

The plot window is very basic at present. But, adding edit functions is planned.

Zoom:
But, you can zoom in by holding the left shift button down and drawing a rectangle over area of interest.
You can then zoom back out by double-clicking the right mouse button.

© 2023 Raymond Allen

CASTLE Simulator Details

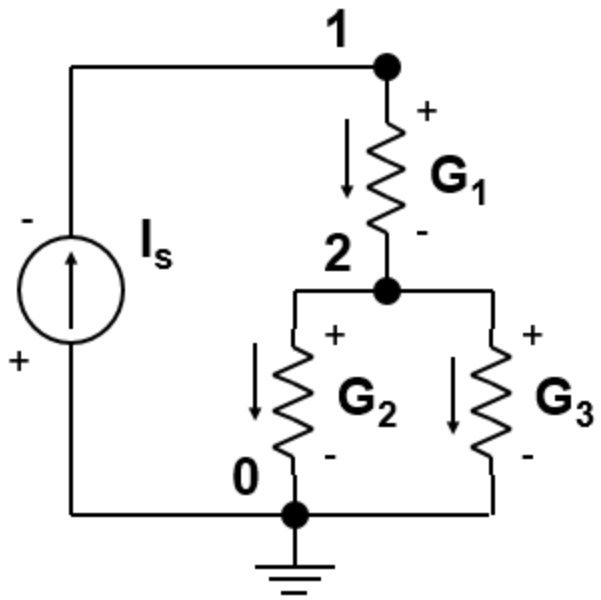
6 CASTLE Simulator Details

This section contains pages that describe the inner workings of CASTLE.

CASTLE is an MNA (Modified Nodal Analysis) based circuit solver.
To explain MNA, we first examine regular Nodal Analysis.

6.1 Nodal Analysis

Nodal Analysis solves a circuit using KCL (Kirchhoff's Current Law) on all nodes in a circuit.
Consider the circuit below:



This example circuit contains one current source, I_S, and three resistors with conductance (1/resistance) values of G_1, G_2 and G_3.
KCL dictates that the sum off all currents entering a node is zero.
The voltage at node #0 is ground, so fixed at zero. So, the
#equations=#nodes-1
T
We can write equations for KCL at nodes 1 and 2 by expressing the voltages at nodes 1 and 2 as V_1 and V_2:

G1*(V1-V2) = Is
G2*V2+G3*V2-G1*(V1-V2)=0

These equations can be solved by hand to find that:

V2 = Is /(G2+G3) = Is*(R2||R3)
V1 = Is /(G2+G3) + Is /G1 = Is*(R1+(R2||R3))

© 2023 Raymond Allen

where ($R_2||R_3$) is the parallel combination of R_2 and R_3, or $R_2*R_3/(R_2+R_3)$

For solving large circuit, it is much more convenient to write the equation in matrix form.
The solution can be viewed as solving the linear equation: **GV=I**
Where, G is the admittance matrix, V is the node voltage vector and I is the current source vector.
Note: The direction of positive current is defined as entering the + terminal and leaving the - terminal.
There is one row in the G, V and I for every node in the circuit (not counting the ground node).

So, we can express the two KCL node equations from above in matrix form as:

$$\begin{bmatrix} G_1 & -G_1 \\ -G_1 & G_1+G_2+G_3 \end{bmatrix}\begin{bmatrix} V_1 \\ V_2 \end{bmatrix}=\begin{bmatrix} I_S \\ 0 \end{bmatrix}$$

This matrix can be solved to yield the same result for V_1 and V_2 given above.
See Matrix Solving for more information on how CASTLE solves the matrix.
The next section discusses Modified Nodal Analysis 57

6.2 Modified Nodal Analysis (MNA)

The problem with regular nodal analysis is that it can't handle voltage sources. MNA adds the currents through voltages to the equation as a way to solve circuits with voltage sources.

So, the circuit equation (**GV=I)** becomes:

$$\begin{bmatrix} G & F \\ B & R \end{bmatrix}\begin{bmatrix} V \\ I \end{bmatrix}=\begin{bmatrix} C \\ E \end{bmatrix}$$

The **G** matrix is mostly the same, but we essentially add one row and one column for each voltage source in the circuit.

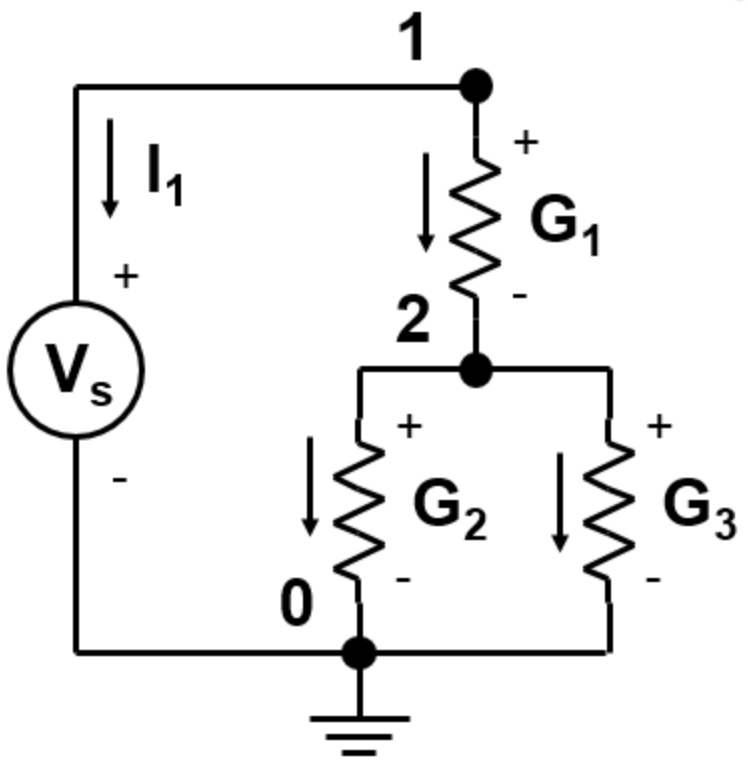
The **F** matrix has one column for every voltage source in the circuit with a 1 at the + terminal and a -1 at the - terminal.
The **B** matrix is the transpose of **F** and **R**==0.

The **V** vector is still the voltage at every node, but the **I** vector is now the current through each voltage source.

© 2023 Raymond Allen

The **C** vector (what was I in nodal analysis 56) is the current source vector and **E** is the voltage source vector (the voltage of each voltage source).

Example. We replace the current source from the nodal analysis example with a voltage source:



The circuit equations become:

G1*(V1-V2) +I1=0
G2*V2+G3*V2-G1*(V1-V2)=0
V1=Vs

Which can be converted to this matrix form:

$$\begin{bmatrix} G_1 & -G_1 & 1 \\ -G_1 & G_1+G_2+G_3 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ I_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ V_s \end{bmatrix}$$

See Matrix Solution Methods 59 for how CASTLE solves this kind of matrix.

© 2023 Raymond Allen

Note that the 0 on the main diagonal (The R section of the new G matrix) is a problem that will be solved by Pivoting 60.

6.3 Matrix Solution Methods

Here are a few ways to solve a linear matrix Ax=B (GV=I in this case):

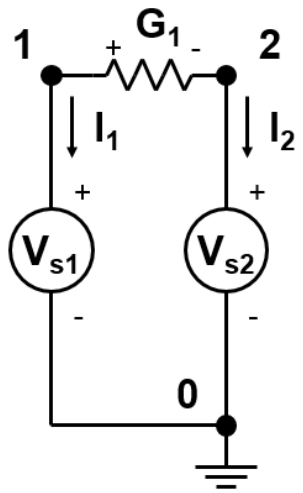
- **Hard way: Matrix Inversion**
  - A-1=adj(A)/|A| → A-1Ax=A-1b → x=A-1b
- **Better way: Gaussian Elimination**
  - $$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & a'_{32} & a'_{33} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b'_3 \end{bmatrix} \rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & 0 & a''_{33} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b''_3 \end{bmatrix}$$
- **Best way: LU Decomposition (what CASTLE uses)**
  - Ax=b → LUx=b → Ux=L^{-1}b → y=L^{-1}b → x=U^{-1}y

    $$U = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & 0 & a''_{33} \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 \\ a_{21}/a_{11} & 1 & 0 \\ a_{31}/a_{11} & a'_{32}/a'_{22} & 1 \end{bmatrix}$$
  - No extra memory required using Crout's Method as L and U stored in A's memory
  - Can reuse L and U if only b changes
  - Trouble if there is a 0 on the main diagonal → Need Pivoting 60

© 2023 Raymond Allen

6.3.1 Pivoting

Pivoting Required for LU Decomp



- **Second voltage source adds a zero to the main diagonal**

$$\begin{bmatrix} G_1 & -G_1 & 1 & 0 \\ -G_1 & G_1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ V_{S1} \\ V_{S2} \end{bmatrix}$$

- **Simplest pivoting scheme, *preordering*, solves this by swapping the source current equation row with the node voltage equation row of the positive terminal of the source**

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ G_1 & -G_1 & 1 & 0 \\ -G_1 & G_1 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} V_{S1} \\ V_{S2} \\ 0 \\ 0 \end{bmatrix}$$

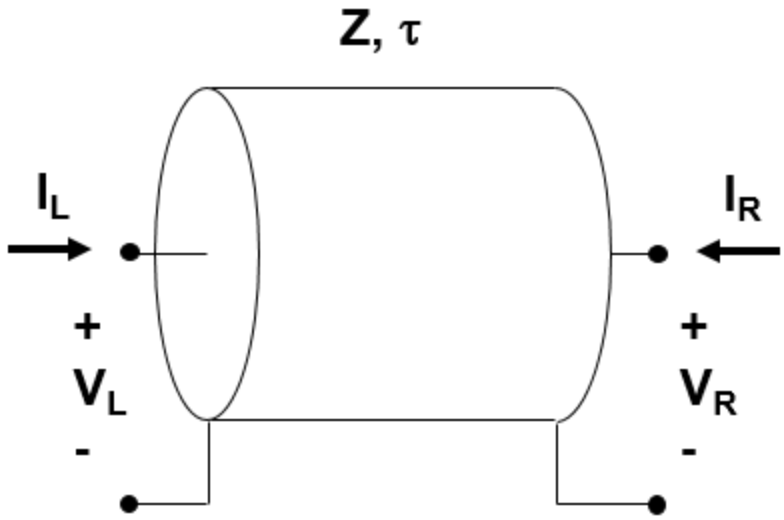
- **Numerical Recipes' LUDCMP routine reorders rows for maximum values on diagonal.**

6.4 Transmission-Line Models

Here we describe how CASTLE handles transmission lines.
From the outside of a transmission line we can envision an External Model that knows only the currents and voltages at each end:

© 2023 Raymond Allen

External Model

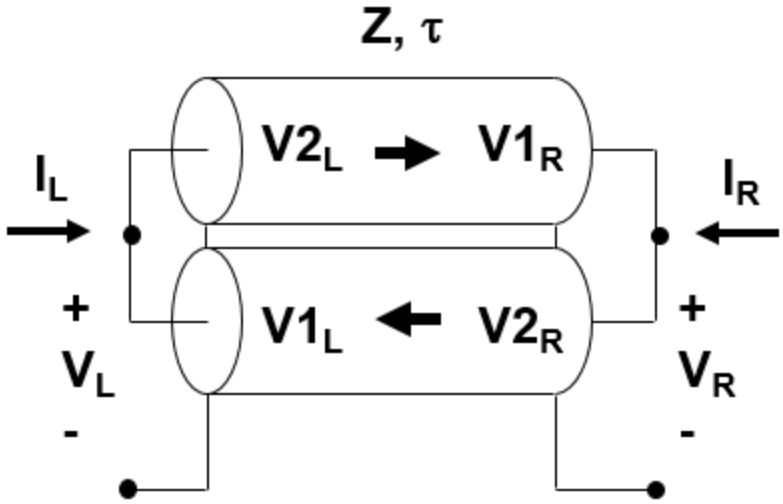


- Z is the impedance of the line
- τ is the one-way transit time
- No current flows on outer surface
- If a fast pulse is sent in the left side at $t=t_0$ and the right side is an open circuit, then $V_R[t_0+\tau]=2*V_L[t_0]$, $V_L[t_0+2*\tau]=V_L[t_0]$
- If right is a short circuit, $V_L[t_0+2*\tau]=-V_L[t_0]$

Transmission line codes (Such as Stella) work by calculating the waves going into each transmission-line end at each time step (V2) by forming equations around the waves coming out of each end of the transmission line (V1) and then solving how all the waves interact at each junction in terms of reflection and transmission coefficients:

© 2023 Raymond Allen

Transmission-Line Model

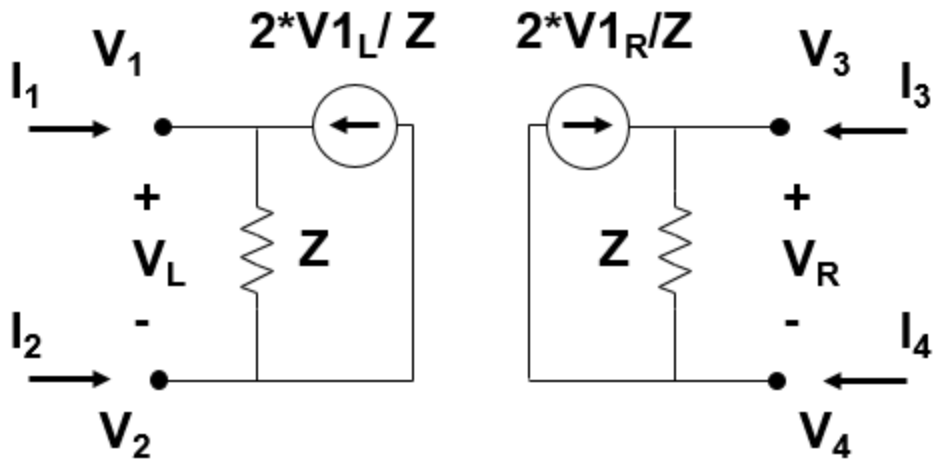


- V1 are the waves exiting the line
- V2 are the waves entering the line
- $V1[t] = V2[t-\tau]$
- $R = (Z_L-Z) / (Z_L+Z)$ = Reflection Coef.
- $T = 1 + R$ = Transmission Coef.
- M = portion of T_i from connected ends, i
- $V2 = R*V1 + \sum_{i=2..n} V1_i*M_i$ $\quad (M_1=R)$
- $V = V1 + V2$
- $I = (V2-V1) / Z = (V-2*V1) / Z$

In CASTLE, each end of a transmission line is treated as an isolated circuit. We do calculate the same V1 wave at each time step, as in transmission line codes, but we apply this to the value of a current source:

© 2023 Raymond Allen

CASTLE Model



- V1[t] = V2[t-τ]
  - $V1_R[t] = V2_L[t-\tau]$
  - $V1_L[t] = V2_R[t-\tau]$
- I = (V-2*V1) / Z
  - $I_1 = -I_2 = (V_1 - V_2)/ Z - 2*V1_L/ Z$
  - $I_3 = -I_4 = (V_3 - V_4)/Z - 2*V1_R/ Z$
- Solve **GV=I**
- V2 = V – V1
  - $V2_L = V_L-V1_L = V_1-V_2-V1_L$
  - $V2_R = V_R-V1_R = V_3-V_4-V1_R$

© 2023 Raymond Allen

This page is intentionally left blank.
Remove this text from the manual
template if you want it completely blank.

Component Reference

7 Component Reference

Look here for more detailed information on the more complex components.

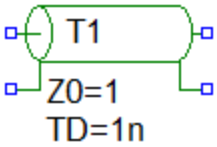
Note: Two terminal devices all rotate around their positive node. The direction of positive current is into the positive node and out the negative node. Transmission lines rotate around the positive node on the left side.

Transmission Line 66
Transmission Line Inductor 67
Transmission Line Capacitor 68
Voltage Source 68
Variable Resistor 72
Switch 73
Variable Capacitor 76
Variable Inductor 78
Voltage Probe 79
Current Probe 79
Transformer 79
Semiconductor Diode 80
Simple Diode 81
SubCircuit 82
Lossy TL 83
Lossy TL Transformer 85
Lossless TL Transformer 85
Lossy TL Resistance 85
Railgun 86
Rod Pinch Diode 87
Pinched Beam Diode 90
Cylindrical Diode 92
Planar Diode 95
Ekdahl Diode 96
Fuse 106
Imploding Load 106
Saturable Core 109
MOV 111
Sandia Switch 97
Sandia Tom Martin Switch 99
Sandia Resistive Wall Model 101
Sandia Plasma Opening Switch 103

7.1 Transmission Line

The simple transmission line is at the heart of CASTLE, making it easy to use was a key motivator.

© 2023 Raymond Allen



The two parameters of a transmission line are the characteristic impedance, Z0, in Ohms, and the time delay, TD, in seconds.

CASTLE makes unique use of the self-isolated ends of transmission lines to speed simulation.
A large circuit with no transmission lines can be sped up by adding transmission lines that let CASTLE divide up the circuit into smaller circuits.

Unlike other schematic entry codes, you do not need to add a ground to a simple connection between two transmission lines. CASTLE will assume that for simple connections of two transmission lines, that the outer conductors are ground.

All transmission line type elements in CASTLE rotate round their left, positive node (center conductor).

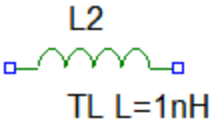
The convention for current polarity is the same as Spice with positive flowing into positive node and out of negative node.

Note that TL waves are available as outputs. V1 is wave leaving left or right ends (L or R suffix) and V2 is wave entering each end.

Like Spice the initial condition can be a voltage or voltage and current with comma between them.

7.2 Transmission Line Inductor

The transmission-line inductor was added for comparison to the transmission-line code, Bertha. In Bertha, you can use a single time step transmission line as an inductor and specify only the inductance instead of time delay and impedance, for convenience. So, this was added to CASTLE for direct comparison of circuits.



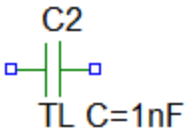
In the schematic editor the only difference between regular inductor and TL inductor is the addition of "TL " to the value. A inductor type can be changed by simply adding or removing "TL " to the start of the value.

© 2023 Raymond Allen

The impedance of the transmission line that is created for simulation is determined by the relationship: TD = L / Z0 , where TD is time delay in seconds, L is impedance in Henries, and Z0 is characteristic impedance in Ohms.

7.3 Transmission Line Capacitor

The transmission-line capacitor was added for comparison to the transmission-line code, Bertha. In Bertha, you can use a single time step transmission line as an capacitor and specify only the capacitance instead of time delay and impedance, for convenience. So, this was added to CASTLE for direct comparison of circuits.



In the schematic editor the only difference between regular capacitor and TL capacitor is the addition of "TL " to the value. A capacitor type can be changed by simply adding or removing "TL " to the start of the value.

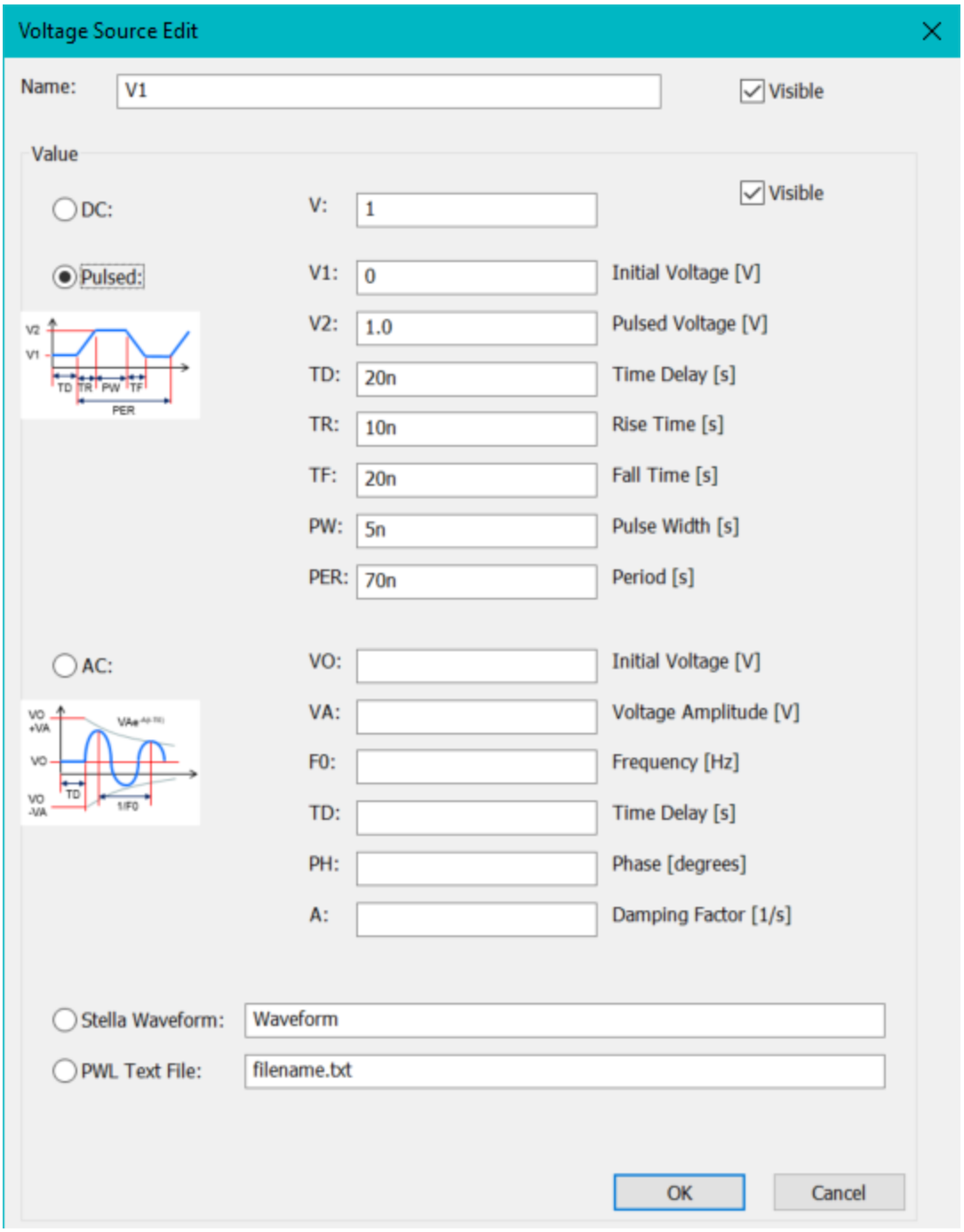
There is a slight difference between the CASTLE and Bertha implementation. In Bertha, one end of a TL capacitor is connected to the circuit and the other end is terminated with a high value resistor. In CASTLE, the inner conductors are connected to each other and the circuit, becoming the positive node and the outer conductors are both connected and become the negative node. The length of the CASTLE TL capacitor could be one time step, like Bertha, but to get exact agreement between them, the CASTLE TL is two time steps long.

The impedance of the two time-step transmission line that is created for simulation is determined by the relationship: Z0 = TD / C , where TD is time delay in seconds (two time steps), C is capacitance in Farads, and Z0 is characteristic impedance in Ohms.

7.4 Voltage Source

A constant or variable voltage source.

© 2023 Raymond Allen



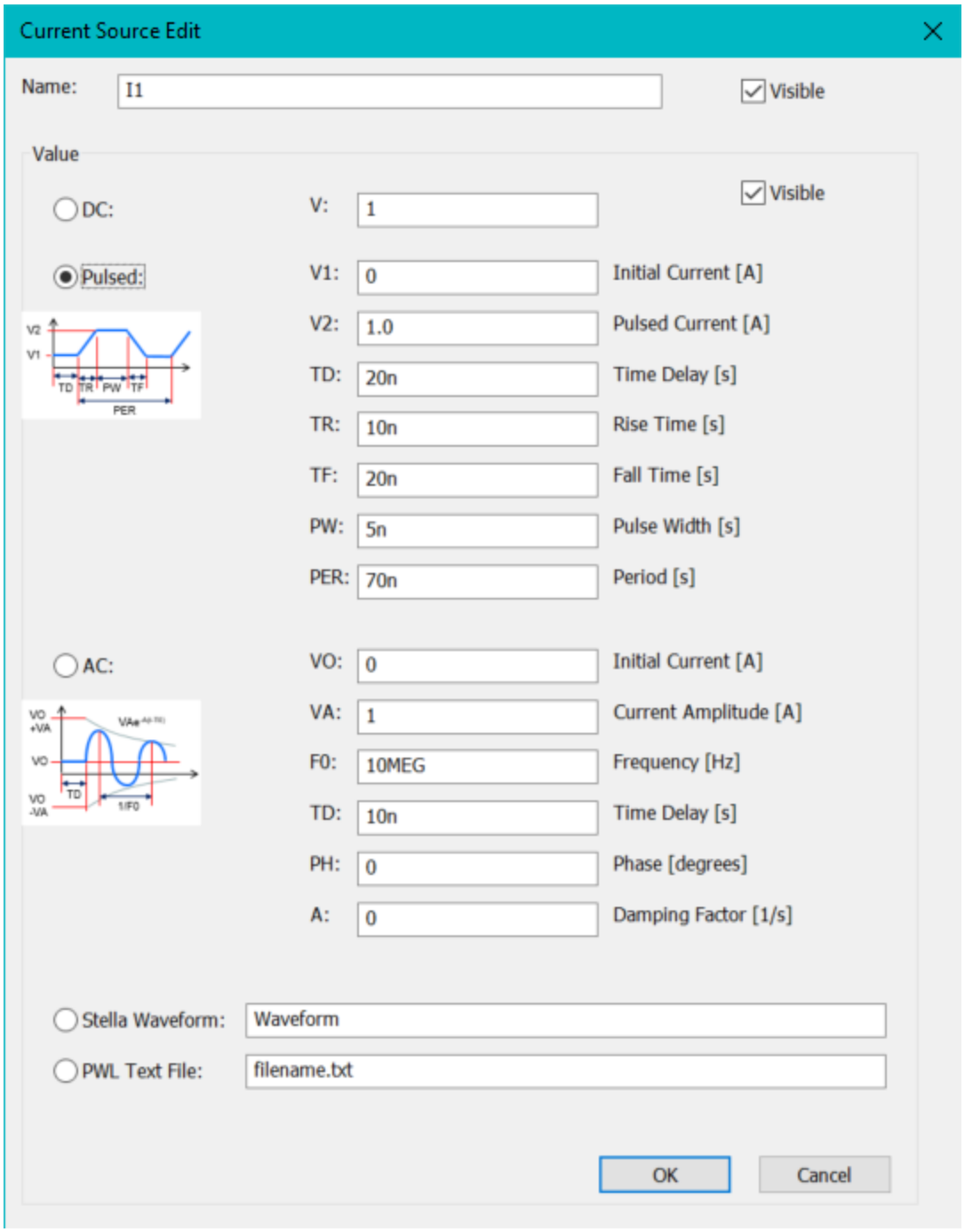
DC: Used to specify a constant voltage source
Pulsed: Used to specify a pulsed voltage source in the style of SPICE
AC: Used to specify an AC voltage source in the style of SPICE.
Stella Waveform: A Stella waveform can be used to specify the voltage profile.
PWL Text File: Uses a Piece-Wise linear text file (similar to LTSpice) for voltage profile. Each line of text has one time, a space, and then a voltage.

© 2023 Raymond Allen

7.5 Current Source

A constant or variable voltage source. The inputs are basically a copy of the Voltage Source 68.

© 2023 Raymond Allen

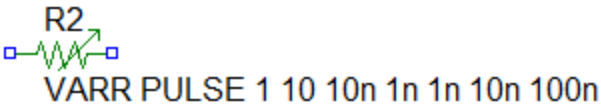


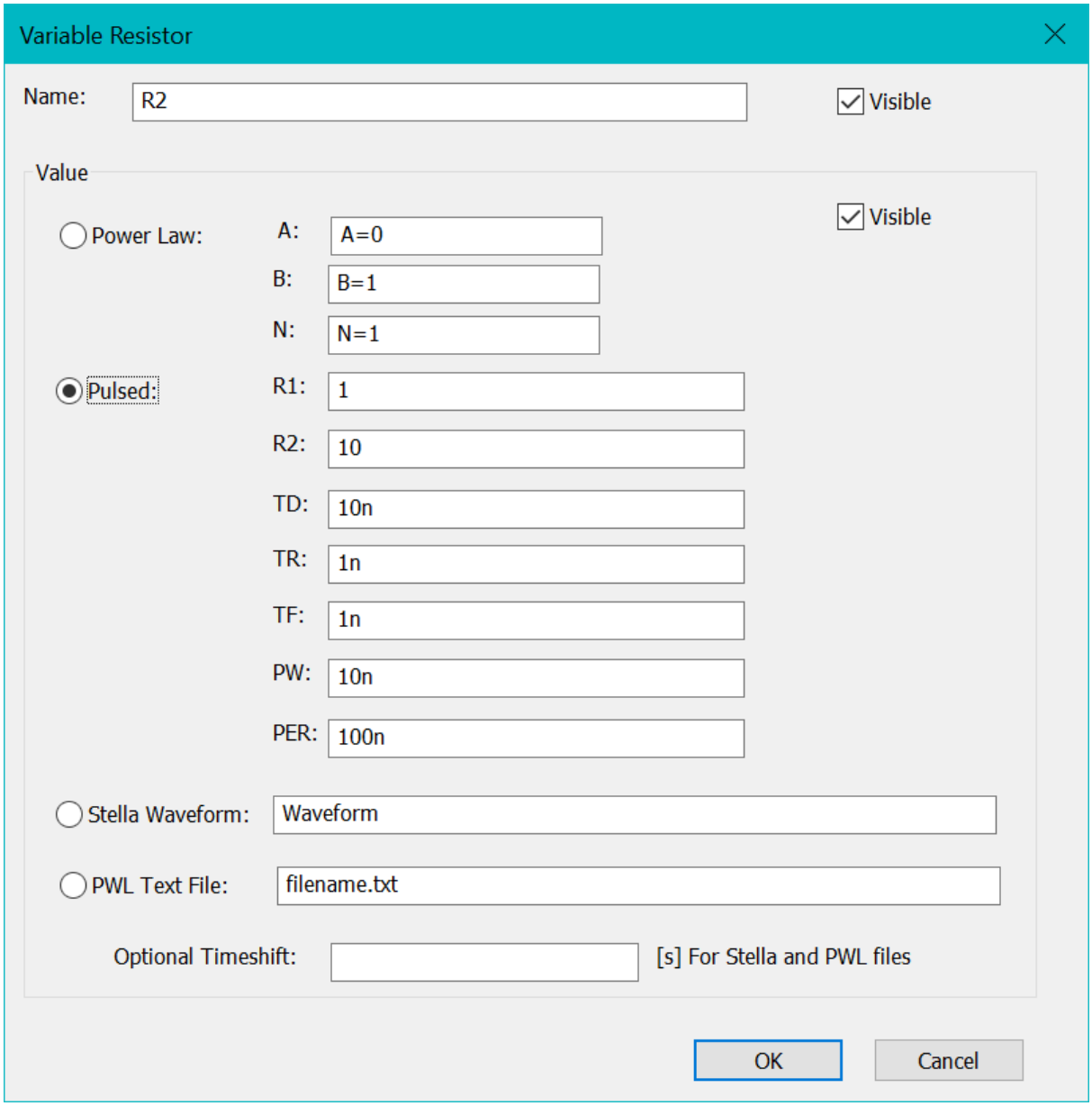
DC: Used to specify a constant current source
Pulsed: Used to specify a pulsed current source in the style of SPICE
AC: Used to specify an AC current source in the style of SPICE.
Stella Waveform: A Stella waveform can be used to specify the current profile.
PWL Text File: Uses a Piece-Wise linear text file (similar to LTSpice) for current profile. Each line of text has one time, a space, and then a current .

© 2023 Raymond Allen

7.6 Variable Resistor

A resistor that changes value during simulation with a few options for how this is done.

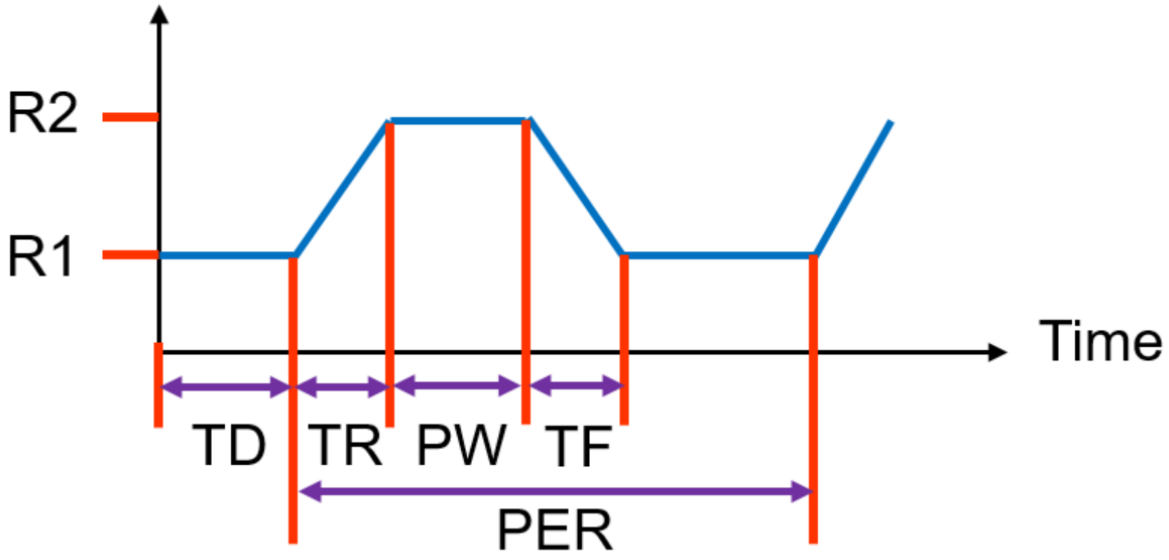




© 2023 Raymond Allen

Power Law: Resistance at current time step = A + B*V^N. V is presently the voltage from previous time step, but reiteration could be added in the future so that it would be V from last iteration on current time step.

Pulsed: Vary resistance between two values in the same way a SPICE pulsed voltage source does. See image below for TD=Time Delay, TR= Rise Time, PW= Pulse Width, TF=Fall time, and PER= period.

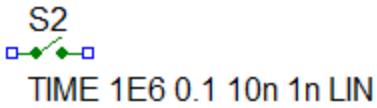


Stella Waveform: Uses a Stella waveform as resistance versus time.

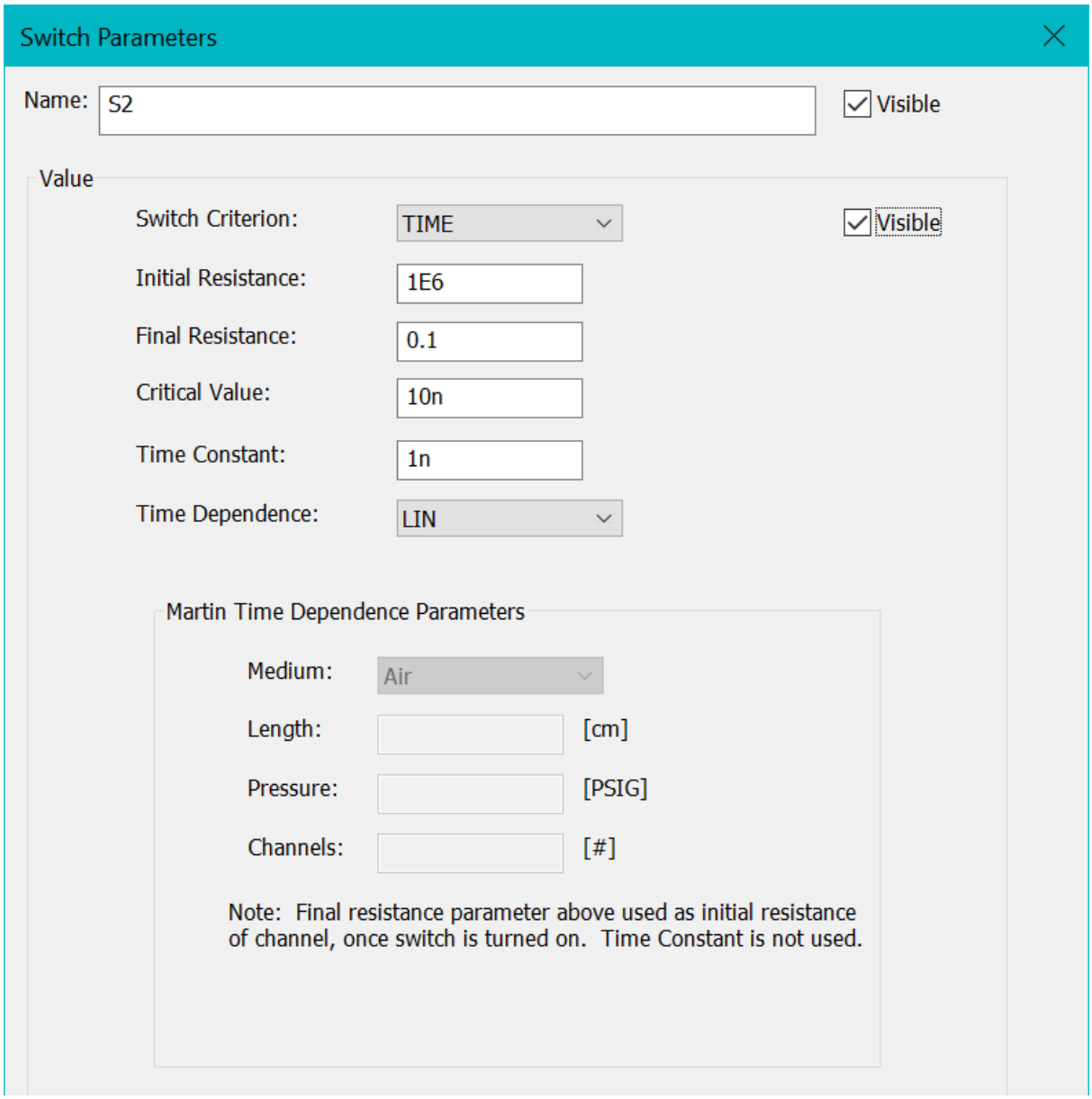
PWL Text File: Uses a Piece-Wise linear text file (similar to LTSpice) for resistance profile. Each line of text has one time, a space, and then a resistance.

7.7 Switch

This is essentially a copy of the Bertha regular switch model combined with the Bertha Tom Martin switch model 75.
Below is a slightly modified copy of the Bertha help for these.



© 2023 Raymond Allen



This subroutine models a simple closing or opening switch. Switching can be specified to occur at a given time, voltage, current, or energy. After the switching time the switch resistance will progress from a given initial value to a given final value, with either an exponential, linear, or T0/T time dependence, or an arbitrary dependence supplied as a waveform. The switching time is available as a scalar output.

At the first timestep, the resistance is set to the initial value supplied here.

Switch Criterion: Choose Time (in ns), Current (in A), Voltage (in V) or Energy (in J). The absolute value is used for voltage and current.

Initial Resistance: The initial switch resistance (Ohms).

© 2023 Raymond Allen

Final Resistance: The final switch resistance (Ohms).

Critical Value: This is the value of the above criterion at which switching occurs (seconds)

Time Constant: This is the time constant (seconds) used in the switching time dependence below.

Time Dependence: Choose exponential, linear, Tau/t, or Martin.

Time dependences are defined as follows, where fact is defined as (time - switching time) / Tau:

Exp Resistance = FR + (IR - FR) * exp(-fact)

Lin Resistance = FR + (IR - FR) * max((1 - fact), 0.)

t/tau Resistance = max(1 / fact, FR)

Tanh Resistance = FR + 0.5 * (IR - FR) * (1 - tanh(fact))

The Tanh time dependence is different because it projects backward in time to -infinity. When the Time crierion is used with this dependence, the resistance is adjusted from the beginning to have this dependence. But for the other criteria, this is not possible because the switching time is not known in advance. In that case, the resistance will jump from IR to IR + (FR - IR) / 2 at the switching time. Therefore, Tanh time dependence may not be useful unless the switching criterion is Time.

For the Trace time dependence, the resistance is given by IR before the time of switching (as it is with the other options). The Trace waveform defines the resistance for subsequent times. The zero time of the Trace waveform corresponds to the switching time. As a convenience, two scale factors may be used. The amplitude of the supplied resistance is multiplied by the fourth (FR) argument, unless that argument is zero. The timescale of the supplied resistance is multiplied by the Tau argument unless that argument is zero. For example, if Tau is set to 2.0, then the resistance change will be "stretched out" by a factor of two.

Future: Sw time The name of the field where the switching time (also in ns) is to be stored.

For Martin Dependence:

This subroutine models a the resistive phase of a closing switch using a model developed by Tom Martin at SNL (T. H. Martin, J. F. Seaman, and D. O. Jobe, Proc of the 9th Int. Pulsed Power Conference, pp.463-470 (1993)). This model

© 2023 Raymond Allen

uses work by Braginskii to calculate the resistivity of the expanding ionization channel(s) in the switch. The resistance drops as the channels expand and heat. One interesting consequence of this model is that the resistive-phase resistance drops rather slowly with the number of channels (as n^-.33), because of the reduced heating and expansion per channel that results. Note that the absolute value of the current is used.

Medium This subroutine has coefficients for either Air, SF6, H2O, or Oil switch media.

Medium: Choose Air, SF6, H20, or Oil media.

Length: The switch gap in cm

PSIG The switch (gauge) pressure in PSI.

chan The number of switching channels

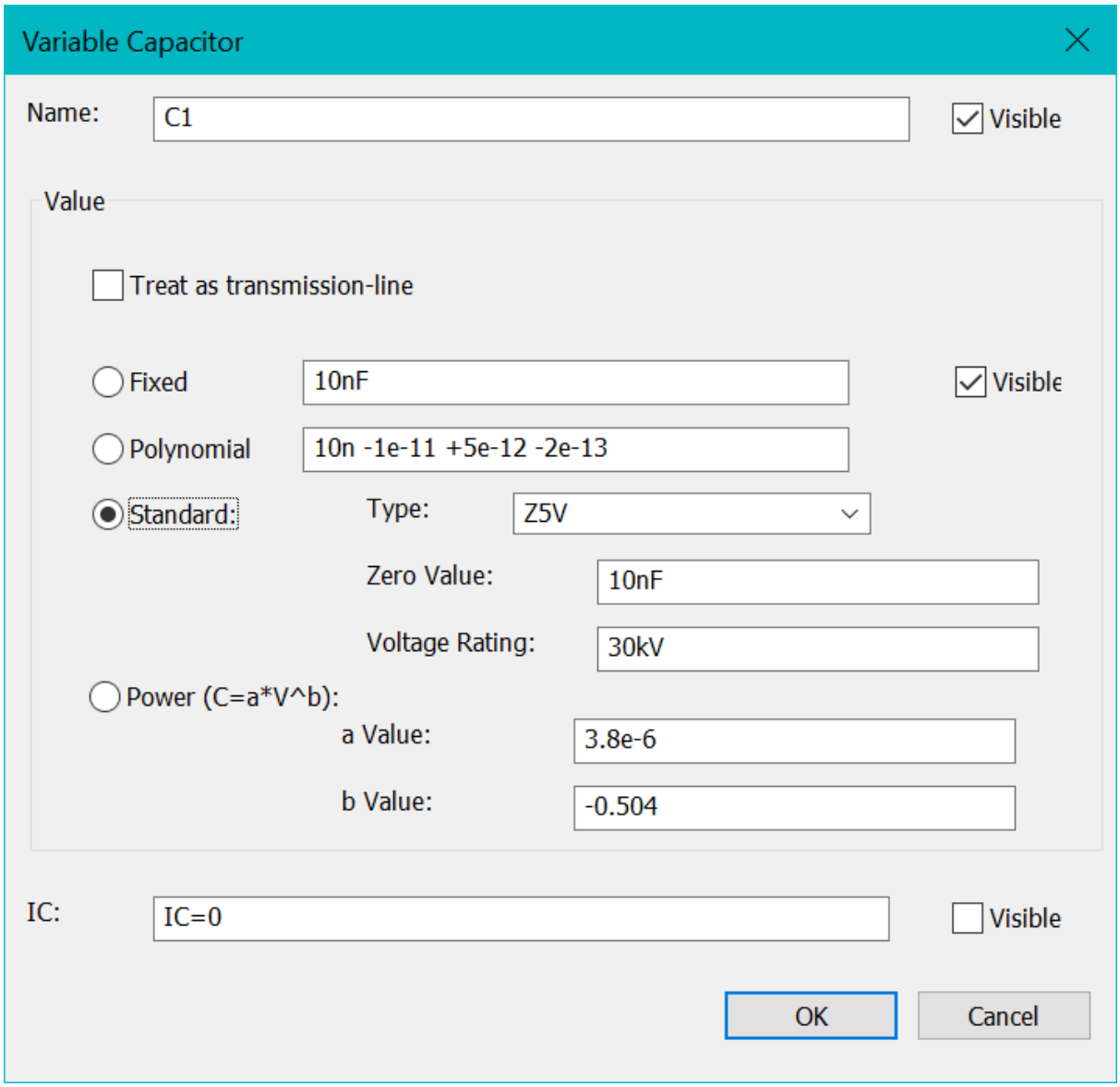
Future: Sw time The name of the optional scalar output containing the switching time

Note that there has been a significant amount of research since the original paper was published (for example, Sandia Report SAND2005-6994), and the calculations in this subroutine probably do not represent the current state of the art.

7.8 Variable Capacitor

A capacitor that changes value with voltage. Can optionally be implemented as a transmission line capacitor.

© 2023 Raymond Allen



Fixed: Just a fixed value (for testing)

Polynomial: Uses 4 values so that C= (((Poly3*V)+Poly2)*V+Poly1) *V+m_dPoly0, where V is voltage from previous time step.

Standard: Calculates a reduction in capacitance with voltage base on data sheet data for a few common types of capacitors.

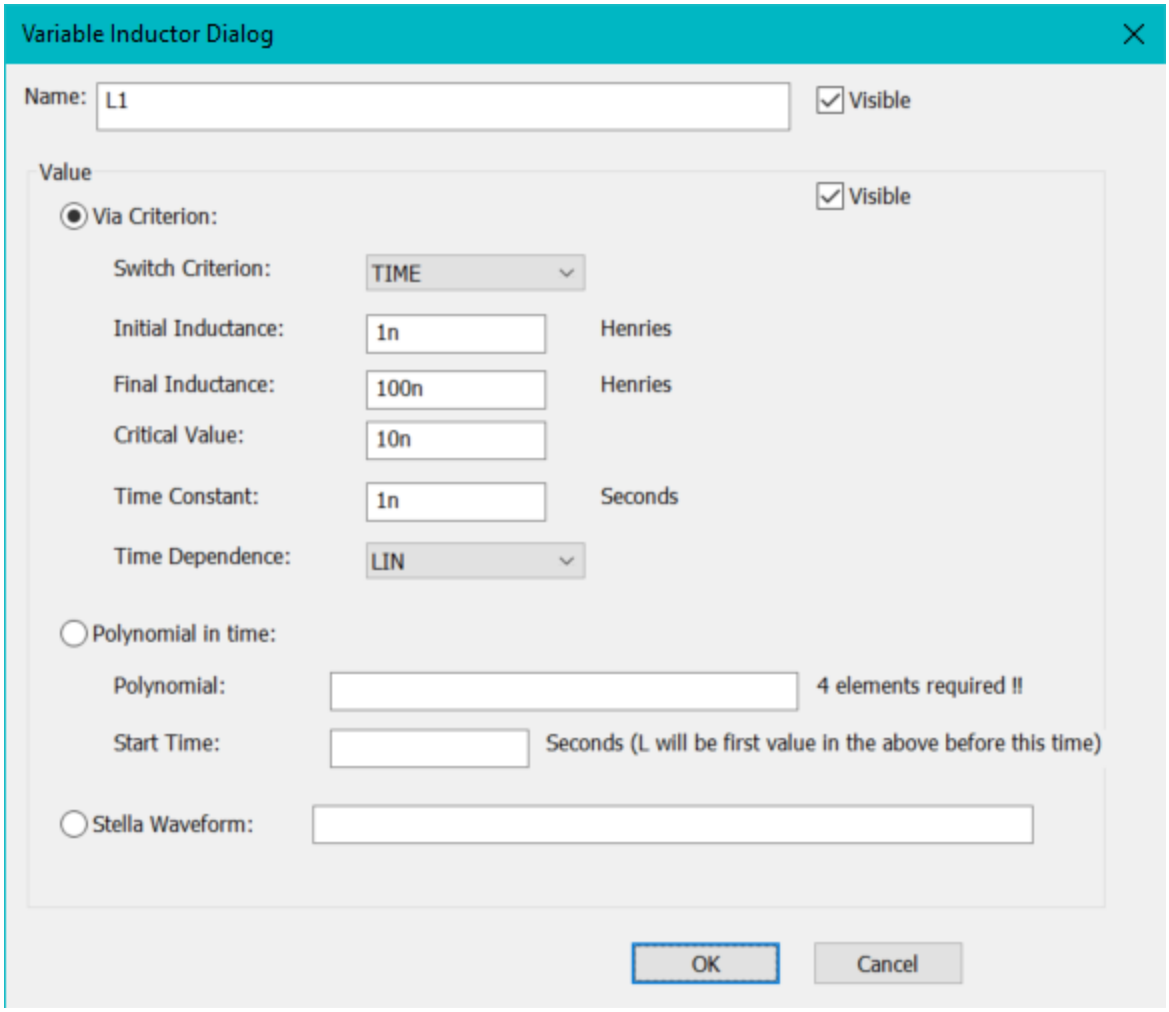
Power: C = a*V^b, where V is voltage from previous time step. This can be used to match a variable capacitor in LTSpice when Q is quadratic in x.

IC is initial condition, i.e., initial voltage.

© 2023 Raymond Allen

7.9 Variable Inductor

A variable inductance. An inductor whose inductance can vary in a few ways. The variable inductor is implemented as a transmission line inductor (one time step long) with a variable impedance.



Via Criterion: This like the switch model 73 but changing L instead of R.

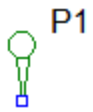
Polynomial in Time: After start time, L = (((Poly3*t) + Poly2)*t + Poly1)*t + Poly0, where t is the simulation time - start time in seconds. Before the start time, L=Poly0.

© 2023 Raymond Allen

Stella Waveform: You can use a Stella waveform to define the inductance history.

7.10 Voltage Probe

The voltage probe provides a convenient way to name a node in the simulation and more easily see the corresponding voltage waveform output. Voltage probes are usually not needed, however, because the voltage across all parts is available after simulation.

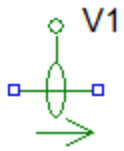


The nodes in the CASTLE generated netlist are usually numbers preceded by the letter "N". But, when a probe is connected to the circuit, the corresponding node is renamed to the name of the probe, allowing it to be more easily selected for plotting.

Probably not a good idea to have two probes or any bubble connectors connected to the same wire in a circuit.

7.11 Current Probe

A current probe allows one to more easily plot the current in a simulation. Current probes are usually not needed, however, as the current through all elements is available after simulation.

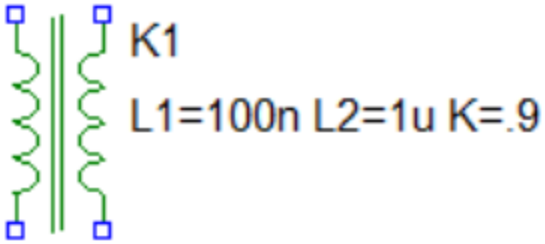


The current probe is implemented as a voltage source set for zero volts. So, the part name starts with the letter "V", which may not seem intuitive.

7.12 Transformer

The CASTLE transformer model is equivalent to what can be done in Spice, but with a simpler implementation. In Spice, one would create two inductors and then use a K control statement to link them. In CASTLE, the two inductors and the K are combined in one element.

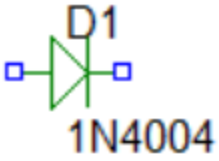
© 2023 Raymond Allen



The variables that define the element are L1, the inductance of the left inductor (Henries), L2, the inductance of the right inductor (Henries) and K, the coupling coefficient (unitless). The transformer rotates around the positive node of the left inductor. The initial current in the left and right inductors can be specified.

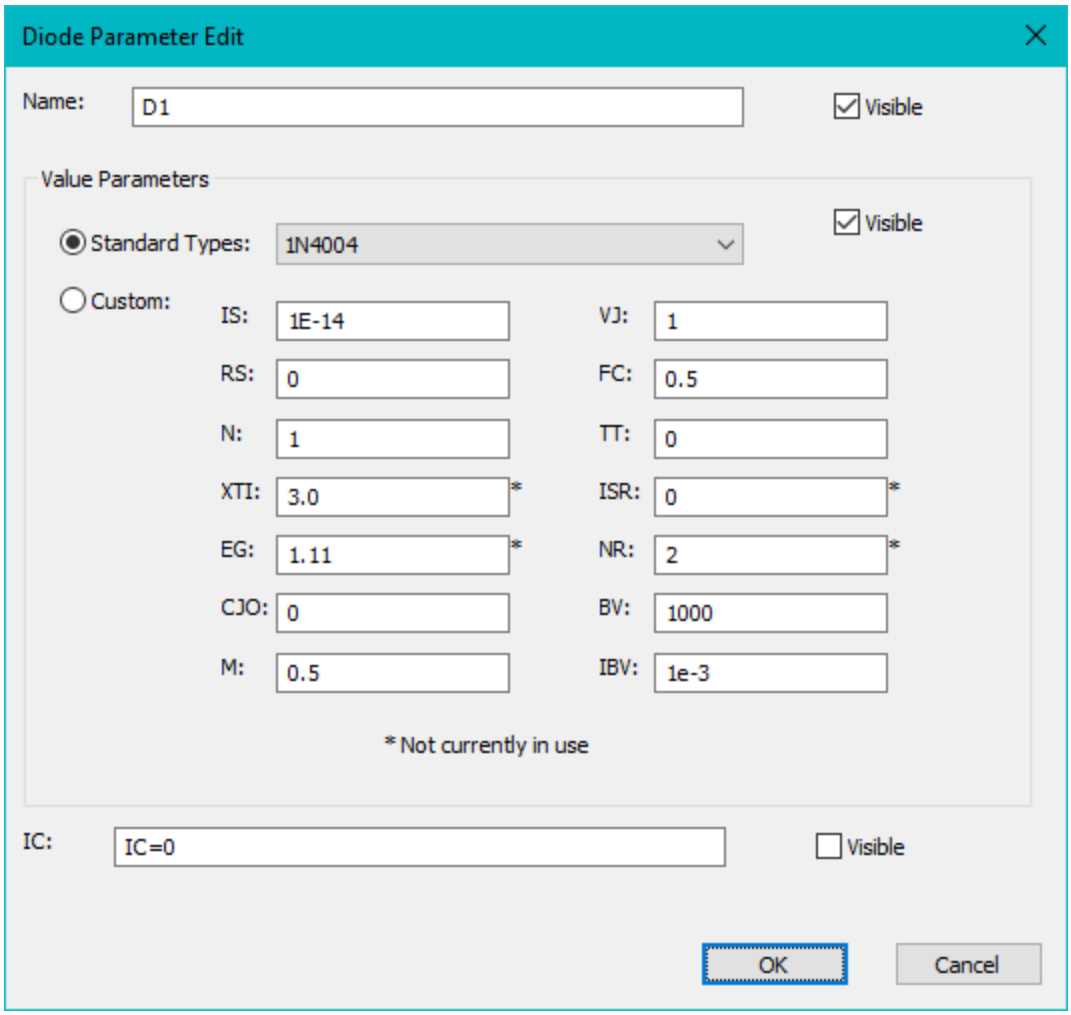
7.13 Semiconductor Diode

The semiconductor diode model is similar in implementation as in Spice.



One can select from a few standard types to set parameters, or enter them under "Custom". These parameters are the same as in Spice, although not all are currently used, as noted. Note that in pulsed power circuits at high voltage, one must use a large value for the Breakdown Voltage, BV, to avoid convergence issues.

© 2023 Raymond Allen

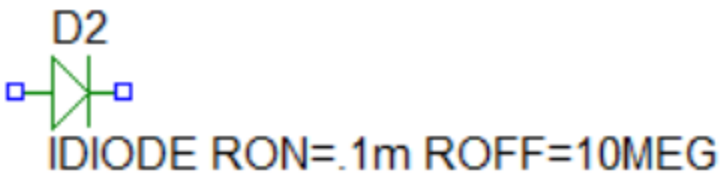


The semiconductor model is one of the more complex models in CASTLE. It is the only one that can reiterate many times while linearizing around the operating point, as does Spice. It is possible for convergence to fail, especially as we have a fixed time step. Instead of aborting the simulation in this case, CASTLE currently caps iterations at around 50 and if convergence fails, it will use the operating point from the previous time step.

7.14 Simple Diode

The simple diode is an attempt to avoid the convergence issues that the Spice diode can have. It has a kind of digital Schmitt trigger where switching only occurs when many of the last time steps indicate that it should. This diode can be used as a last resort if the semiconductor diode model fails.

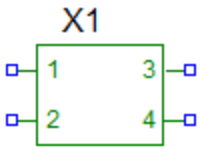
© 2023 Raymond Allen

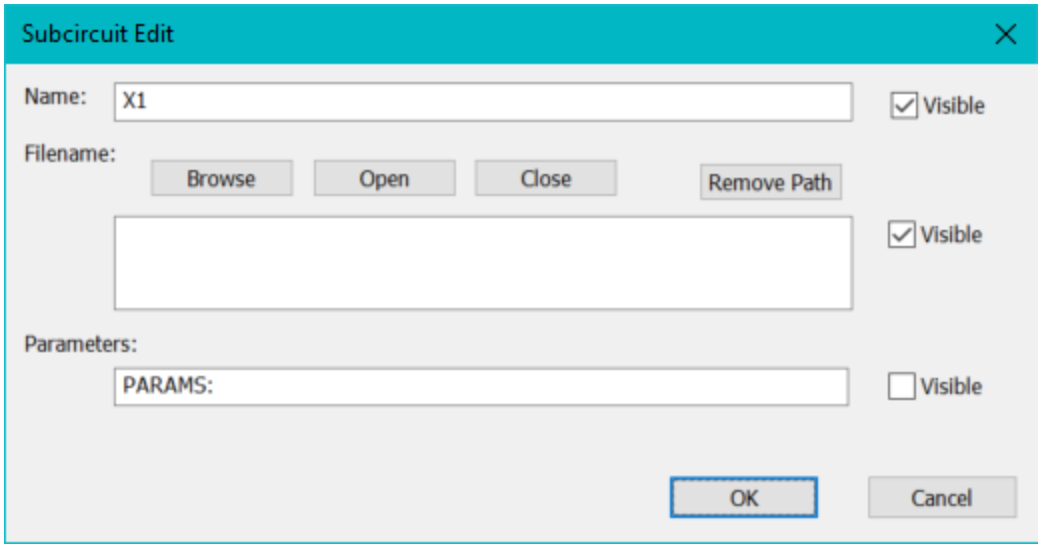


There are only two parameters for the simple diode, RON, the on state resistance in Ohms, and ROFF, the off state resistance in Ohms. This part also rotates around the positive node. The diode begins in the off state and if the diode voltage is positive for several time steps, it will switch in a few time steps to the on state. If the voltage is negative for several time steps, the diode will transition to the off state.

7.15 SubCircuit

The SubCircuit component allows one to include other schematics into a main schematic. This can be very useful where, for example, there are several copies of the same circuit in the main schematic. Using SubCircuits can make a schematic more clear.





Bubble connectors with special names ("sub1", "sub2", "sub3", and "sub4") are used in the subcircuit to connect to the main circuit. Not all four need to be used. Unused nodes can just be left unconnected. It is not necessary to use a node for ground as ground is universal.

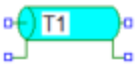
© 2023 Raymond Allen

Use the "Browse" button or just type or paste the name of the subcircuit .cas file in the "Filename" area. It is sometimes useful to remove the path with the "remove path" button, especially if the files are copied somewhere else.

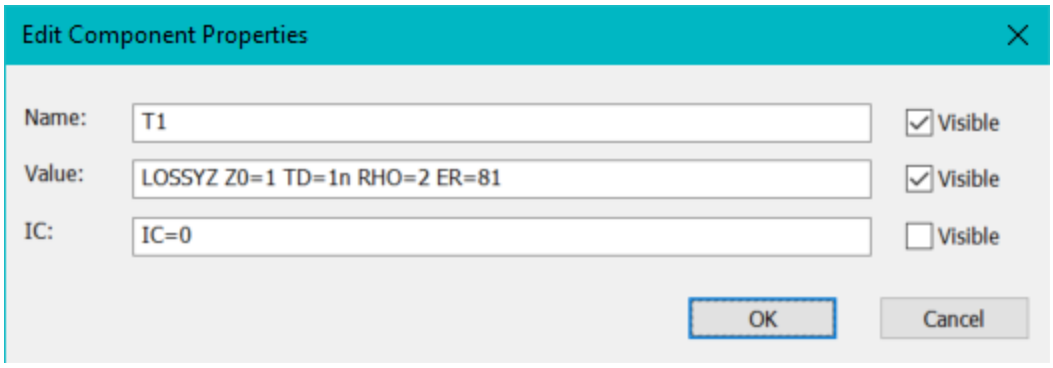
Each copy of the subcircuit in the main circuit can be fed unique parameters using the "Parameters" field. These parameters will be used just as if there were a "PARAMS:" parameter control line 40 in the subcircuit.

7.16 Lossy TL

This is a model for a lossy transmission line. This was implemented to model the effect of the conductivity of water in long pulse setups.



LOSSYZ Z0=1 TD=1n RHO=2 ER=81



This is one of a few models that internally uses transmission-line code methodology to handle the loss. The line is essentially broken up into as many single time step elements as needed and then a shunt resistance is added at every junction.

RHO: Resistivity in MegaOhm-cm

ER: Relative Permittivity (unitless), 81 for water.

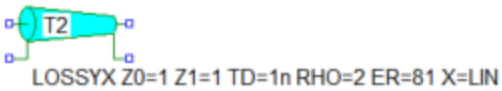
Note that the code uses the relationship between capacitance and resistance to calculate the resistance.
resistance/resistivity = epsilon/capacitance;

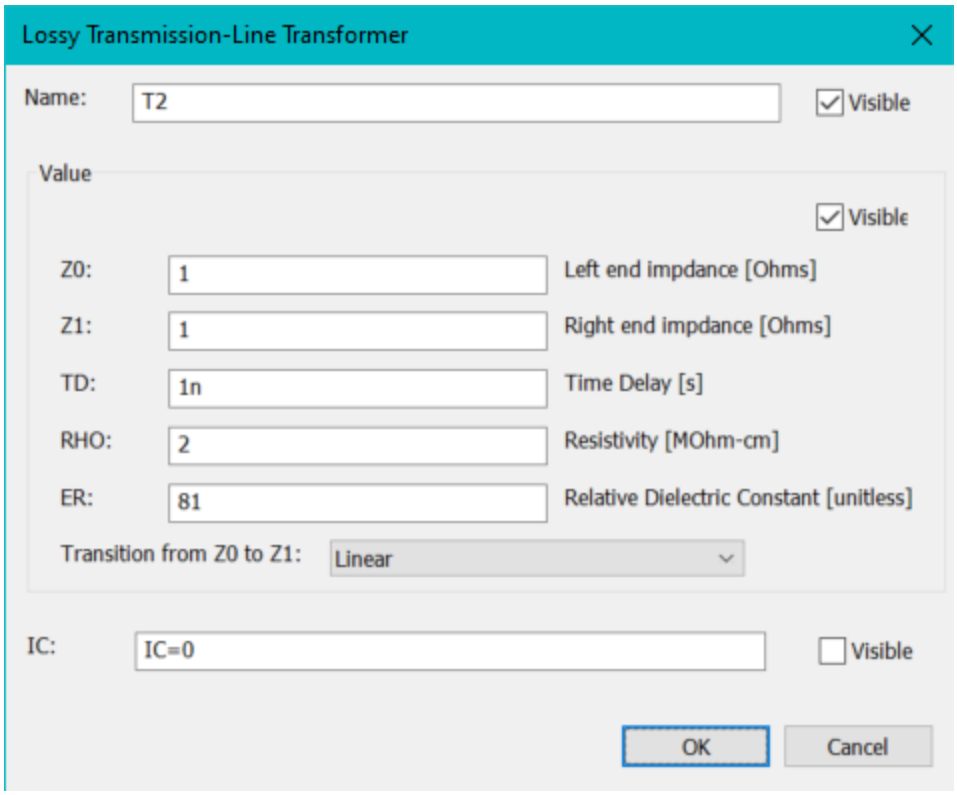
IC= Initial Condition (volts or volts, amps)

7.17 Lossy TL Transformer

This is very similar to the Lossy TL 83 but with the additional option to have the impedance of the line vary from one end to the other.

© 2023 Raymond Allen





This is one of a few models that internally uses transmission-line code methodology to handle the loss and transition. The line is essentially broken up into as many single time step elements as needed and then a shunt resistance is added at every junction and impedance is varied.

RHO: Resistivity in MegaOhm-cm

ER: Relative Permittivity (unitless), 81 for water.

Note that the code uses the relationship between capacitance and resistance to calculate the resistance.
resistance/resistivity = epsilon/capacitance;

The transition can be linear, exponential, or as a coaxial line with either inner or outer conductor varied in radius in a linear way down the line.

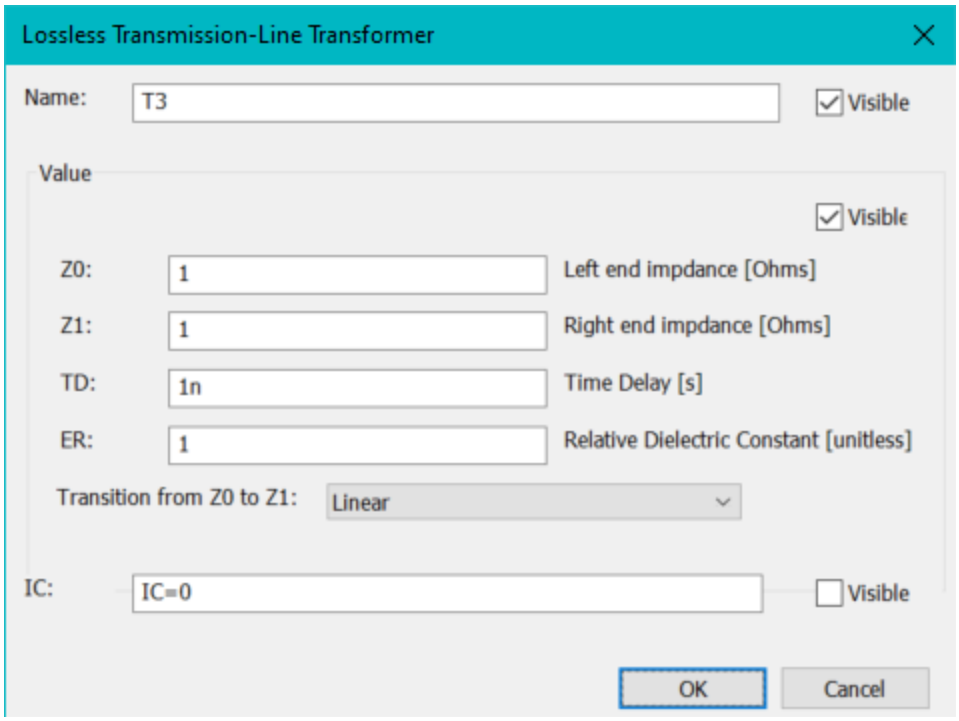
IC= Initial Condition (volts or volts, amps)

© 2023 Raymond Allen

7.18 Lossless TL Transformer

This is very similar to the Lossy TL Transformer, but without any loss.

LOSSLESSX Z0=1 Z1=1 TD=1n ER=1 X=LIN



This is one of a few models that internally uses transmission-line code methodology to handle the transition. The line is essentially broken up into as many single time step elements as needed and impedance is varied.

ER: Relative Permittivity (unitless), 81 for water. This is only used for coaxial transitions.

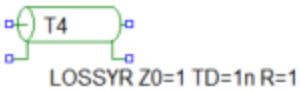
The transition can be linear, exponential, or as a coaxial line with either inner or outer conductor varied in radius in a linear way down the line.

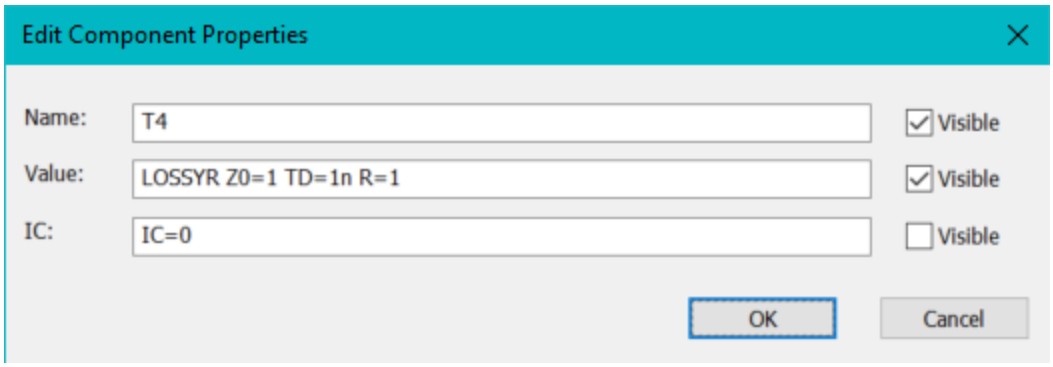
IC= Initial Condition (volts or volts, amps)

7.19 Lossy TL Resistance

This is similar to the Lossy TL model 83, except this is with a series instead of shunt resistance.

© 2023 Raymond Allen





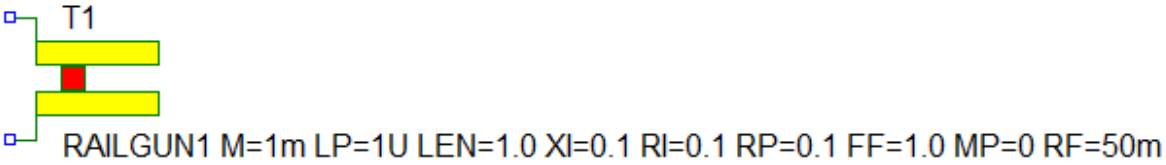
This is one of a few models that internally uses transmission-line code methodology to handle the loss. The line is essentially broken up into as many single time step elements as needed and then a series resistance is added at every junction.

R = the total series resistance (Ohms)

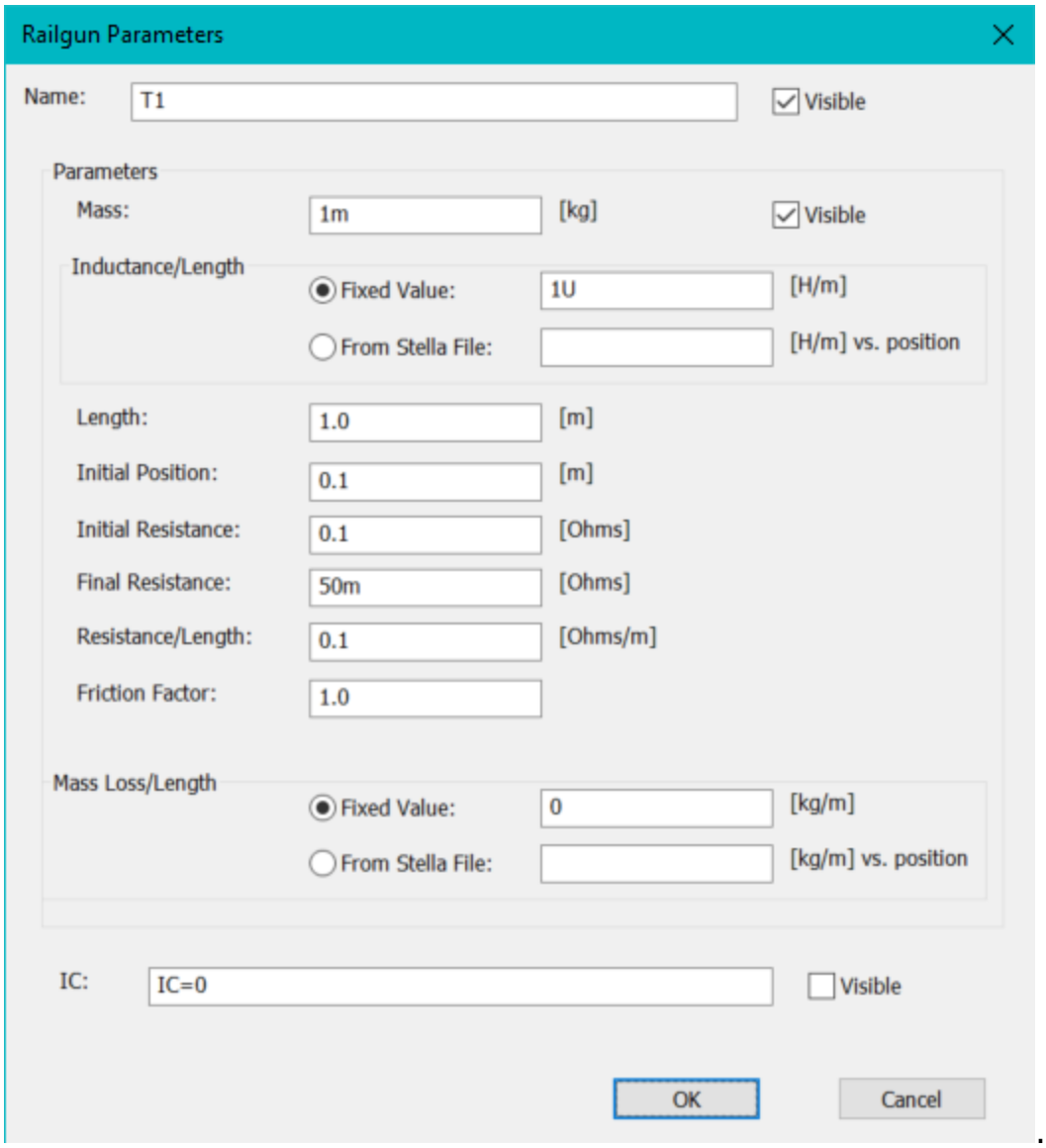
IC= Initial Condition (volts or volts, amps)

7.20 Railgun

Railgun model.



© 2023 Raymond Allen



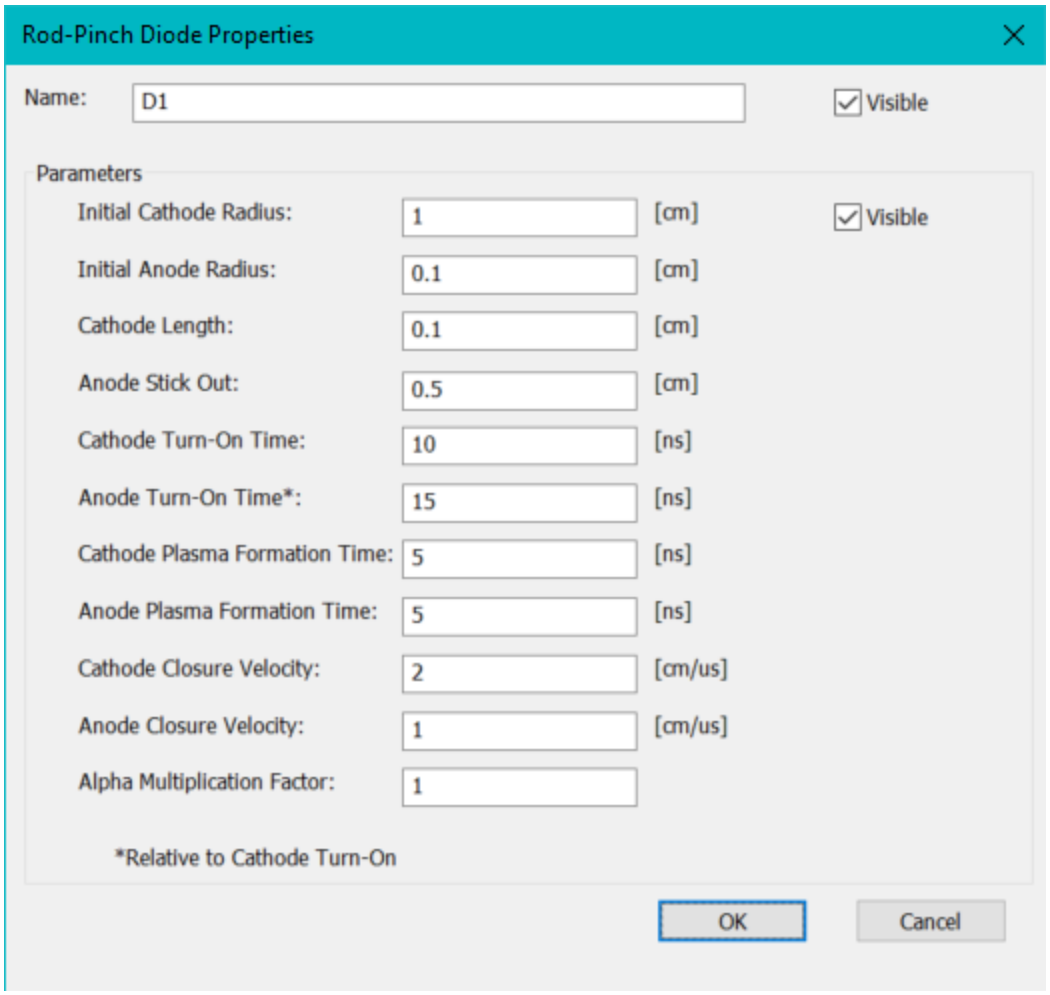
7.21 Rod Pinch Diode

Rod Pinch Diode. Model now has explicit cathode turn on time.

D1

RPDIODE RC=1 RA=0.1 LC=0.1 SO=0.5 TCO=10 TAO=15 TCF=5 TAF=5 VC=2 VA=1 AMF=1

© 2023 Raymond Allen



Code was adapted from Bertha. Here is adapted Bertha notes:

This is based on code written by Steve Swanekamp and is based in part on results of LSP simulations.

rC The cathode radius in cm.

rA The anode radius in cm. Normally the diode will be in positive polarity (i.e., the anode is smaller), but either polarity is allowed. A lookup table supplies the correct polarity-dependent Langmuir-Blodgett coefficients.

C leng The cathode thickness in cm. This affects the space-charge-limited (SCL) current, but not the critical current

A ext The anode stickout in cm. This affects the SCL current - the effective length in the forward direction is taken to be the lesser of the stickout and the radial gap. This also affects the ion-current enhancement during the critical-current phase.

© 2023 Raymond Allen

tC on The cathode turn-on time in ns. All other times are relative to this. This poses a limitation in that this subroutine has no voltage-dependent turn-on. It should probably be updated to include this.

tA on The anode turn-on time in ns, relative to tC on.

tC form The cathode formation time in ns, relative to tC on.

tA form The anode formation time in ns, relative to tA on.

C vel The cathode-plasma closure velocity in cm/microsecond

A vel The anode-plasma closure velocity in cm/microsecond

Al fact The calculated value of alpha is multiplied by this factor, which is typically set to 1.

At time = tC on, plasma expansion begins from the cathode. The calculated SCL current is ramped up from zero to its full value in time tC form. Until time tA on, the ion current and anode-plasma expansion are zero, and the alpha factor is fixed at 1.4 * Al fact. At tA on, anode-plasma expansion begins and the nominal alpha factor is calculated from the electrode radii based on a lookup table. During the anode turn-on time tA form, the alpha factor is transitioned linearly from 1.4 at tA on to the nominal value at tA on + tA form. Finally, the resulting factor is multiplied by Al fact.

The pure-electron SCL current (also with the help of a lookup table) and critical currents are then calculated. The results of simulations are then used to determine the bipolar factor (the increase in electron current due to ions) and ion current fraction as functions of electrode radii and voltage. Both of these are transitioned during the time before complete ion turn-on. Then, the ion-current enhancement resulting pinching is calculated based on the geometry. This gives the final calculated SCL and critical currents. The total current is transitioned smoothly between these as 1/Itotal^3 = 1/Iscl^3 + 1/Icr^3.

Special Outputs:

Future: tcc The name of the scalar output containing the time when the critical current is reached (and thus when strong pinching begins).

IIon The output waveform containing the calculated ion current.

ISCL The output waveform containing the calculated space-charge-limited current.

ICrit The output waveform containing the calculated critical current.

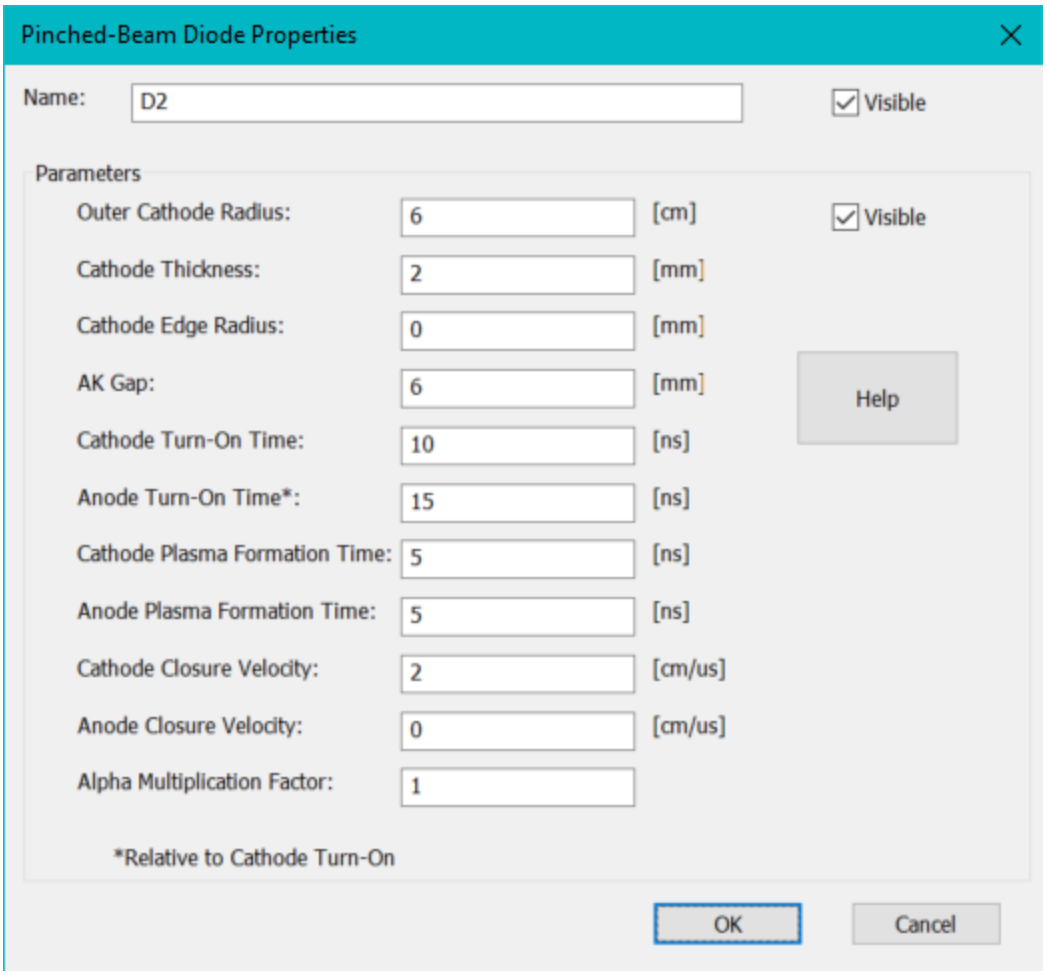
 © 2023 Raymond Allen

Alpha The optional output waveform containing the calculated alpha factor

7.22 Pinched Beam Diode

A pinched-beam diode model. Adapted from Bertha, adapted notes from Bertha below.

D2
PBDIODE OCR=6 CT=2 CER=0 AK=6 TCO=10 TAO=15 TCF=5 TAF=5 VC=2 VA=0 AMF=1



This subroutine models pinched-beam and ring diodes, with a sequential operation as follows:

The model was modified to turn on at a given cathode turn on time rather than voltage as previous version did.

Once the diode turns on, plasma expansion begins from the cathode, reducing the gap.

© 2023 Raymond Allen

Anode plasma formation (and thus ion current) is assumed to start at a specified time (in ns)Once the anode turns on, additional plasma expansion proceeds from the anode, further reducing the gap.

Given the geometry, instantaneous voltage, and gap, the critical and SCL currents are calculated (more on this below). If ion emission has begun, then the SCL current is modified for bipolar flow.

Once anode turnon begins, the ion current is assumed to ramp up linearly to its full value over a specified time. This is included because data, particularly for diodes in the SCL regime, often indicate a gradual transition to full bipolar flow. This model has two such times, depending on whether the diode is operating in SCL or critical-current mode. This gives the user a bit more ability to tweak the model.

The SCL current is calculated using a fit to the voltage dependence that handles relativistic conditions, from Bruce Weber, Dave Mosher, and Paul Ottinger, (IEEE Trans. Plasma Sci., vol. 42, no. 6, pp. 1819, 2014.) This fit also handles the ion current fraction. Edge effects are handled by assuming (1) the beam spreads out at some characteristic angle as it moves to the anode; (2) the beam area on the anode is the relevant area to use in the SCL-current calculation. The justification of these assumptions is not obvious, but LSP simulations confirm the validity of this technique. Thus, the SCL perveance has a term proportional to the area divided by gap squared, plus a term proportional to the circumference divided by the gap.

The critical current is calculated using the standard formula, with an ion-current-dependent enhancement factor alpha. Without ions this is typically unity, and previous work suggests a value of about 1.67 for an ion diode. (The factor of 1.6 is most-commonly used, but Paul Ottinger often uses the value of 60 / 136, or 1.67). The ion-current fraction in a diode operating in the critical-current regime is derived from arguments of space-charge neutrality and path length. The calculated current is enhanced by a factor of two when reflexing is assumed, which is often the case. For a summary of the critical-current formulas, with useful references, see Paul Ottinger's technote 2008-04.

Whether the diode operates in SCL or critical-current phase is determined by which current is lower. However, once the diode transitions to pinched (critical-current) operation, is is assumed to remain that way for the duration. This seems to fit the data better, and only at the very end of the pulse would the SCL current be assumed to fall below the critical current.

For a ring diode, set alpha multiplication factor to two. Set alpha multiplication factor to 1 for regular pinched beam diode.

© 2023 Raymond Allen

Alpha multiplication factor. Internally, the critical current is proportional to an alpha factor. The typical value is 1.6 or 1.67. The code uses 1.6 times the alpha multiplication factor.
So, one can use the alpha multiplication factor if one wants to adjust alpha. This is especially needed for ring diodes, where a multiplication factor of 2 is appropriate.

Special Outputs:

Future: t(Icc) The name of the scalar output containing the time when the critical current is reached (and thus when strong pinching begins).

Icl The name of the output waveform containing the calculated space-charge-limited current.

Icr The name of the output waveform containing the calculated critical current.

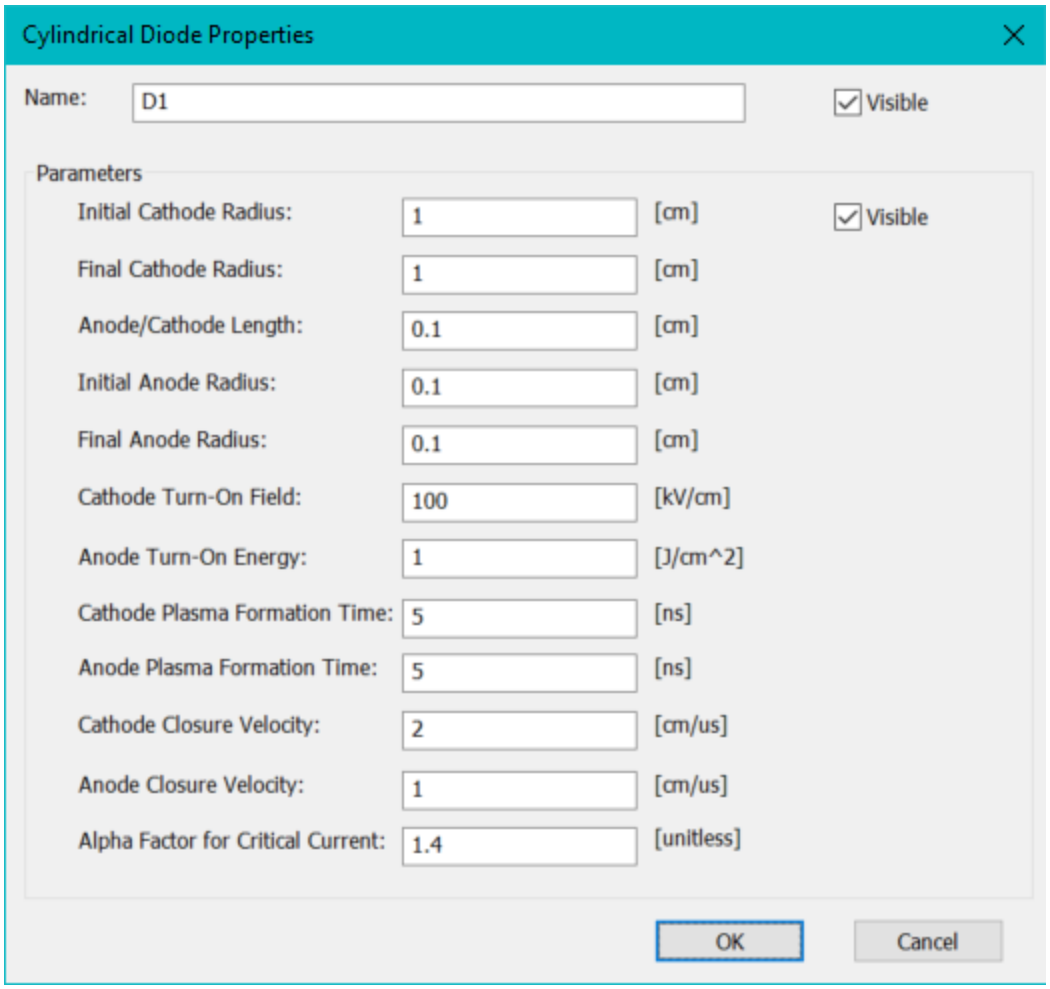
IIon The name of the output waveform containing the calculated ion current.

7.23 Cylindrical Diode

The cylindrical diode models a diode with a geometry like a rod-pinch, but with option for a tapered anode and/or cathode.

D1
CYLDIODE RC=1 RCF=1 RA=0.1 RAF=0.1 LC=0.1 TCF=5 TAF=5 VC=2 VA=1 F=100 E=1 A=1.4

© 2023 Raymond Allen



This is another diode that was adapted from Bertha and adapted help is below.

This subroutine models a cylindrical (rod-pinch-like) diode. It is a similar to the RodPinch subroutine, but allows for conical electrodes. Cathode emission begins at a given electric field threshold, and anode emission begins at a given local deposition of energy. It was created to model diodes used in lower-voltage x-ray generators, which typically employ conical anodes. It is based on code written by Steve Swanekamp, which in turn was guided by the results of LSP simulations.

rCinit Cathode initial (along taper) radius in cm

rCfin Cathode final (along taper) radius in cm

Length Length of the cathode in cm

Eb(kV/cm) Cathode electric field threshold in kV/cm

© 2023 Raymond Allen

tC form Cathode plasma formation time in ns

C vel Cathode closure velocity in cm/usec.

rAinit Anode initial (along taper) radius in cm

rAfin Anode final (along taper) radius in cm

tA form Anode plasma formation time in ns

Ed(J/cm2) Energy density limit for anode plasma creation in J/cm2

A vel Anode closure velocity in cm/us.

The alpha factor here is defined as the initial value. This is different from the Bertha version (and also the CASTLE rod-pinch diode) where it was changed to a multiplication factor.

The diode is represented as a pair of nested cones defined by rCinit, rCfin, rAinit, rAfin, and Length. The cones are modeled as an array of 20 axial segments. Calculation for each segment is performed individually. They turn on individually when the local electric field at the cathode reaches Eb. At this point an alpha factor of 1.4 * Alpha is used. When the energy deposition on the anode reaches Ed, alpha transitions to a value determined from a lookup table (it ranges from 2.1 to 2.7, depending on the aspect ratio), and again multiplied by Alpha. The calculation includes plasma expansion, SCL and critical current, and ion-current enhancement. The currents are then summed among the segments and a single transition from SCL to critical current is calculated (as it is in the RodPinch subroutine).

Special Outputs:

Future: tA on The name of the scalar output containing the time of first anode plasma formation

Future: tcc The name of the scalar output containing the time when the critical current is reached (and thus when strong pinching begins).

IIon The name of the output waveform containing the calculated ion current.

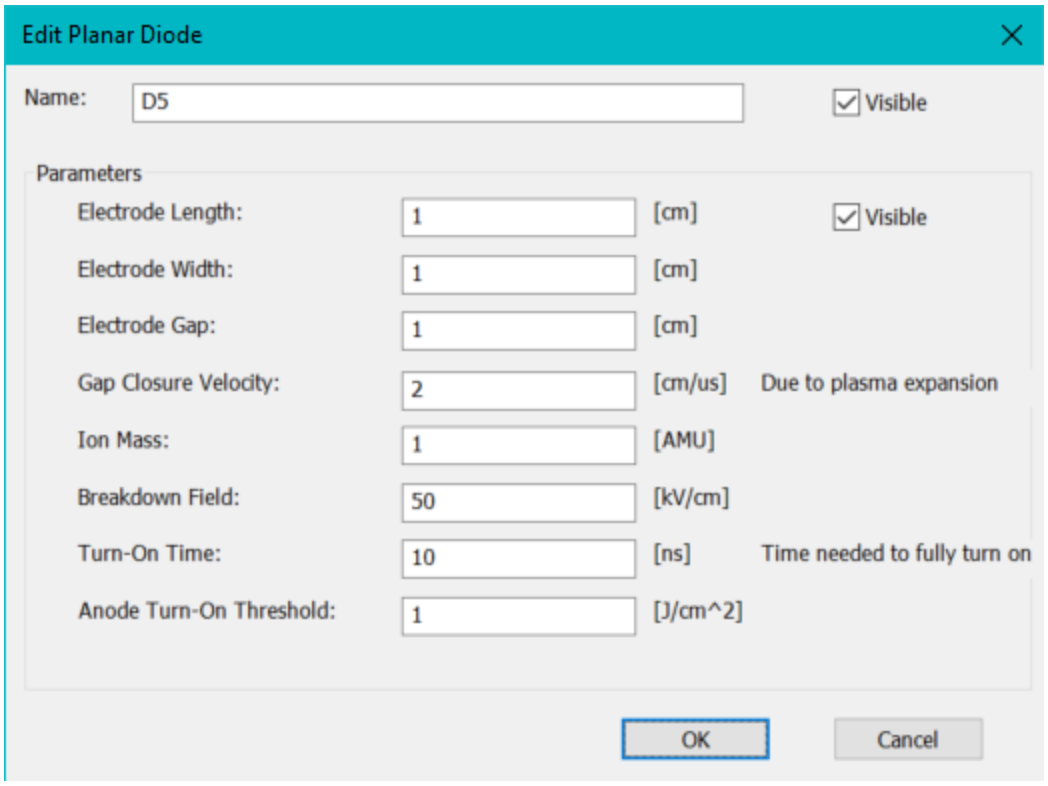
ISCL The name of the output waveform containing the calculated space-charge-limited current.

© 2023 Raymond Allen

ICrit The name of the output waveform containing the calculated critical current.

7.24 Planar Diode

This is a Child's law type space charge limited emission between plates with no edge effects. Relativistic correction only good to around 2 MV.



The code was adapted from Bertha. The help adapted from Bertha is below.

This is an older subroutine that models a planar, space-charge-limited diode. It does not attempt to model edge effects, although these could be approximated by increasing the length and/or width. It was written to help model electron-beam-pumped gas lasers operating at lower currents (or in an axial magnetic field) so that pinching does not occur.

Length The diode length in cm.
Width The diode width in cm. Note that these only appear in the calculation as their product.

© 2023 Raymond Allen

Gap(cm) The diode gap in cm.

ClosVel The effective diode closure velocity in cm/microsecond.

AMU The ion atomic mass (used in calculating the ion component of the diode current).

Eb(kV/cm) The threshold electric field, in kV/cm, for electron emission.

tTO(ns) The turn-on time in ns.

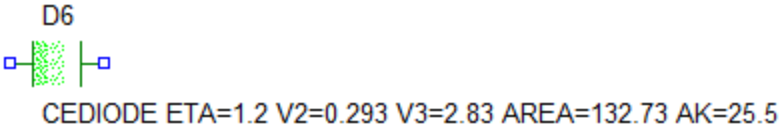
Eion(J/cm2) The specific energy needed to turn on an anode plasma.

The diode does not begin conducting until the breakdown field is reached. At that time the (electron-only) Child-Langmuir (C-L) current is calculated. The diode current increases linearly during the turn-on time tTO from zero to the full calculated value. Once the energy per unit area on the anode exceeds Eion, bipolar flow is assumed.

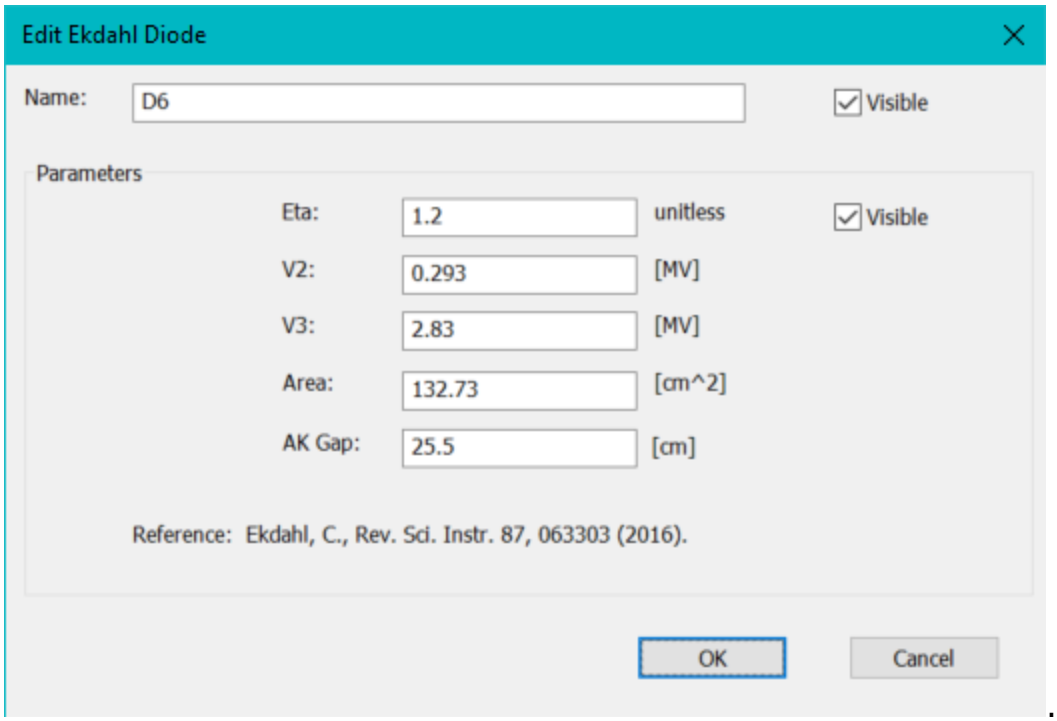
Note: this subroutine uses a crude relativistic correction to the C-L current formula, and would need updating to be used accurately above ~2 MV.

7.25 Ekdahl Diode

This is an approximation for injector diode current used by Carl Ekdahl. See reference below.



© 2023 Raymond Allen

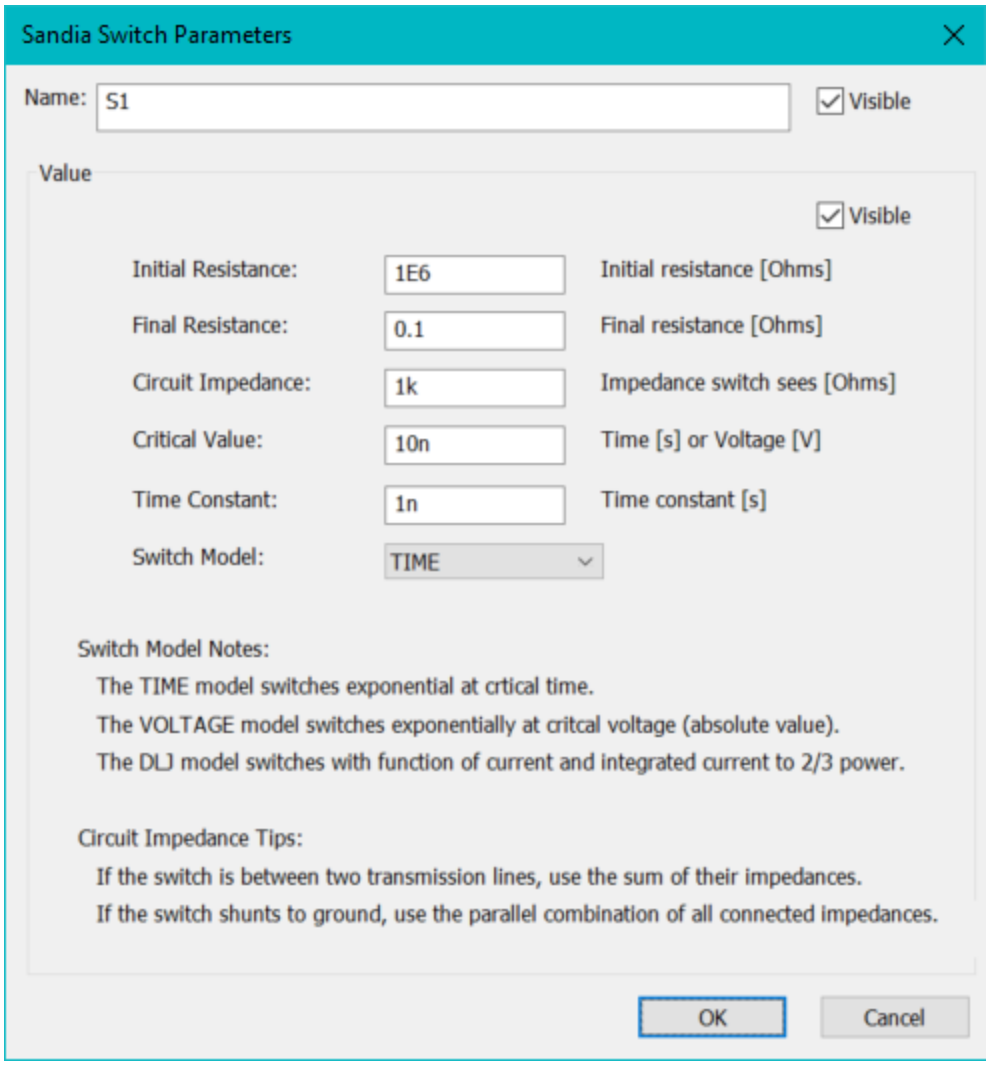


7.26 Sandia Switch

This is the Bertha version of the SNL switch model, adapted for CASTLE.

SNLSW TIME 1E6 0.1 1k 10n 1n

© 2023 Raymond Allen



Below is the Bertha help, adapted for CASTLE

This subroutine is derived from Screamer code and models a simple switch, with two choices for the switching time dependence. It is similar to the Switch subroutine, but uses different models for the switch resistance decay. At the first timestep, the resistance is set to the initial value supplied here, so that the value entered in the junction grid is unused. Three choices are available for the switching model. The first two use the SNL exponential decay model and switch at either a given time or given voltage. The third (DLJ) uses a power-law time dependence developed by Dave Johnson at SNL.

Note: see also the SNLSw2 subroutine, where Zsw and Tau are calculated from the switch parameters directly. That routine is generally preferable, but this one may be useful in checking the effect of Zsw and Tau.

© 2023 Raymond Allen

There is also the SNLSw3 subroutine, which models water switching based on the integral of some power of the electric field.

Ropen The initial switch resistance.

Rclose The final switch resistance.

Zsw The impedance of the rest of the circuit from the view of the switch. The ratio of Zsw to Ropen forms a characteristic time constant

Time The switching time in ns, or voltage in V, depending on the model used. Note that voltage is the absolute value.

Tau The time constant in ns for the first two models, not used for the third

Model (use drop down for CASTLE) Enter a 0 for exponential switching at a specified time, 2 for exponential switching at a specified voltage, and 1 for the DLJ model, switching at a specified time.

Future: Time The scalar output of the switching time. This is only needed for the self-break Model 2.

For the exponential time dependence, the resistance varies as follows, where fact is defined as (time - switching time) / Tau:

Resistance = Zsw * exp(-fact)/ (1.0 - exp(-fact) + Zsw * 1.0 / Ropen) + Rclose

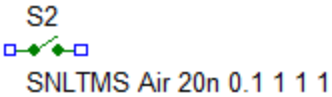
The ratio Zsw/Ropen weakly affects the rapidity of switch closing. The switch closes faster as this ratio drops.

(Note that early versions of Screamer had an error where 1.0/Ropen was replaced by 1.e6.)

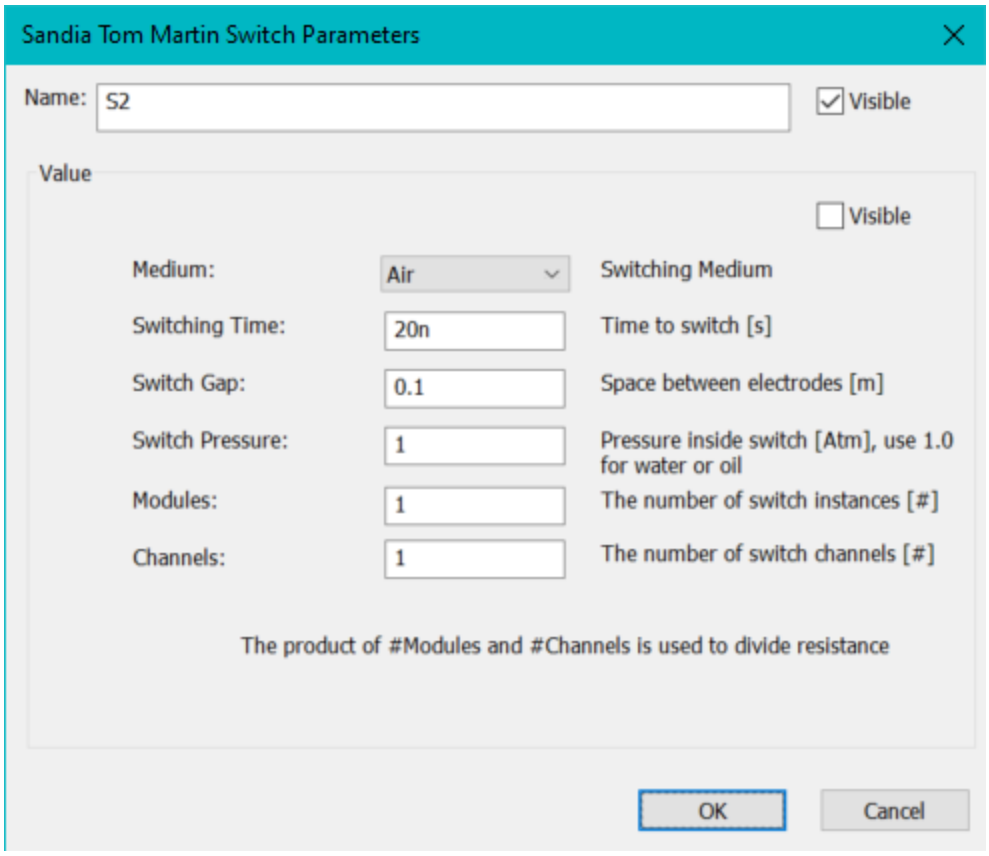
For the DLJ model, the switch resistance is given by R = (charge * charge / current^0.5 / Zsw) ^ 2/3, where charge is the integral of current through the switch.

7.27 SNL Tom Martin Switch

This is the CASTLE implementation of the Bertha SNL Tom Martin switch model.



© 2023 Raymond Allen



The Bertha help on this, adapted for CASTLE, is below.

This subroutine models a the resistive phase of a closing switch using a model developed by Tom Martin at SNL (T. H. Martin, J. F. Seaman, and D. O. Jobe, Proc of the 9th Int. Pulsed Power Conference, pp.463-470 (1993)). This model uses work by Braginskii to calculate the resistivity of the expanding ionization channel(s) in the switch. The resistance drops as the channels expand and heat. One interesting consequence of this model is that the resistive-phase resistance drops rather slowly with the number of channels (as n^-.33), because of the reduced heating and expansion per channel that results. Note that the absolute value of the current is used.

This subroutine is very similar to the SwTMR subroutine, offering two more media but only a triggered option. Eventually, the two should probably be combined and updated.

Medium This subroutine has coefficients for either Air, SF6, H2O, Oil, He, or H2 switch media.

Time The switching time in s

Gap The switch gap in meters.

© 2023 Raymond Allen

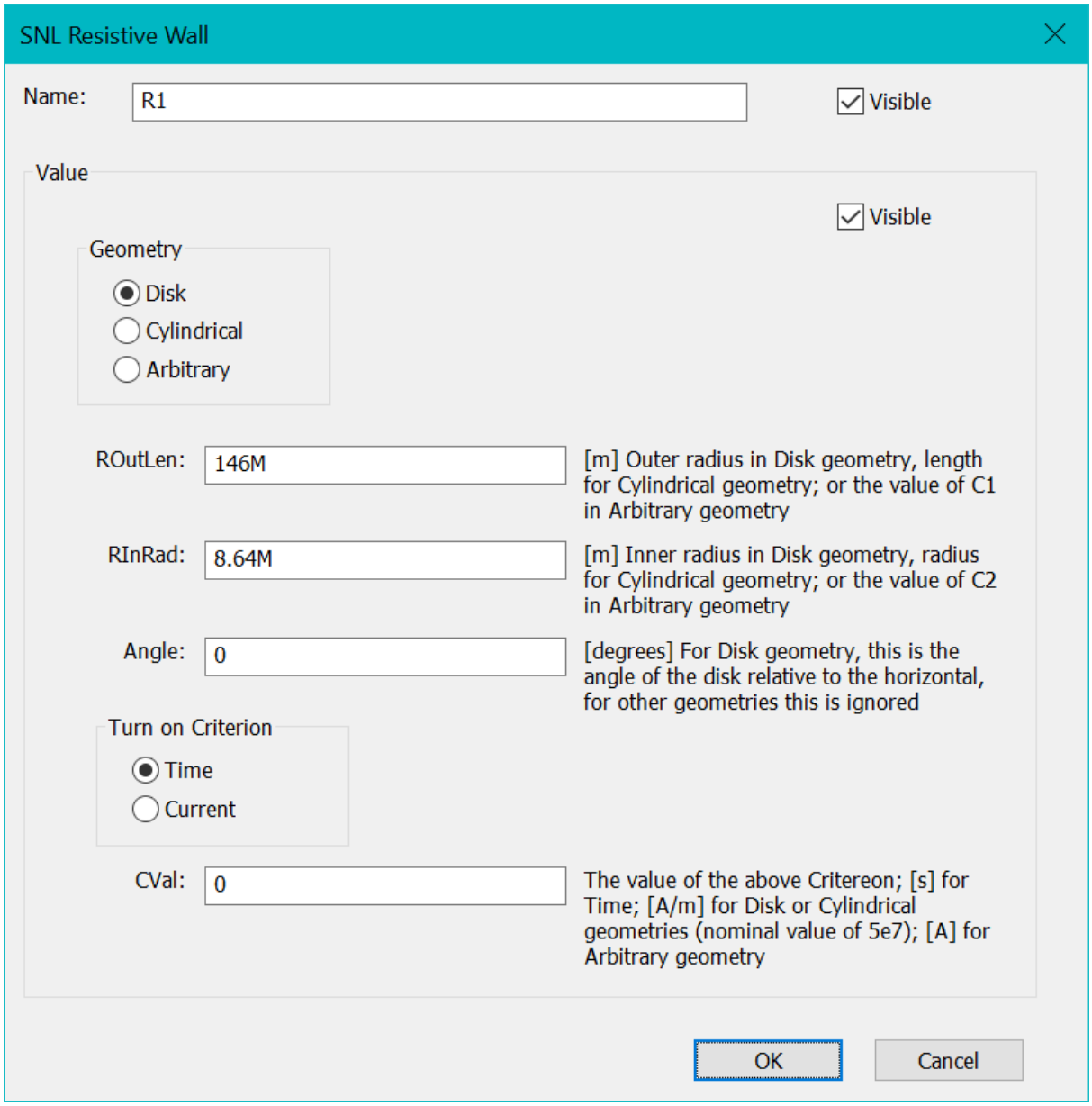
Atm The switch pressure in atmospheres. Use 1 for water or oil.

mods The number of switch modules

chan The number of switch channels. (These two are provided for convenience, although internally they only appear as their product.)

7.28 SNL Resistive Wall

This is another one of several components that is more or less cut and pasted from Bertha. Much of the text below is copied directly from the Bertha Help.



© 2023 Raymond Allen

This subroutine models resistive wall loss in conductors. The model is based on Stygar, Rosenthal, et al., in Phys Rev Spec Topics - Accelerators and Beams 11, 120401 (2008). This paper provides a first-order calculation of energy loss in conductors at high linear current density. Losses due to Ohmic heating, magnetic diffusion, and physical movement of the conductor (work + inductance change) are included. A general equation comprising two terms is presented; for a linearly-rising current in stainless-steel conductors, this takes the form:

R = C1 * time^(-.5) + C2 * I^2 * time^(.25).

The first term includes resistive and diffusion losses; work and inductive losses are included in the second.

The Sandia Resistive Wall model is implemented as a version of a variable resistor:

RESWALL DISK 146m 8.64m 0 TIME 0

Geometry can be either a Disk, Cylindrical, or Arbitrary geometry can be used. In the first two cases, C1 and C2 above are calculated from the dimensions supplied in the following arguments. This is slightly different than the Screamer implementation that can include both a Disk and Cylindrical parts, so two of these models in series may be required for direct comparison.

ROutLen: This is the outer radius for Disk geometry, the length for Cyl geometry (both in meters), or the value of C1 for Arb geometry.

RInRad: This is the inner radius for Disk geometry, the radius for Cyl geometry (both in meters), or the value of C2 for Arb geometry.

Angle For Disk geometry this is the angle of the disk relative to horizontal (in degrees), for other geometries this is ignored.

Turn On Criterion: The criterion for determining when the resistance calculation begins. The code will prevent a singularity for the first term above. The choices are Time and Current.

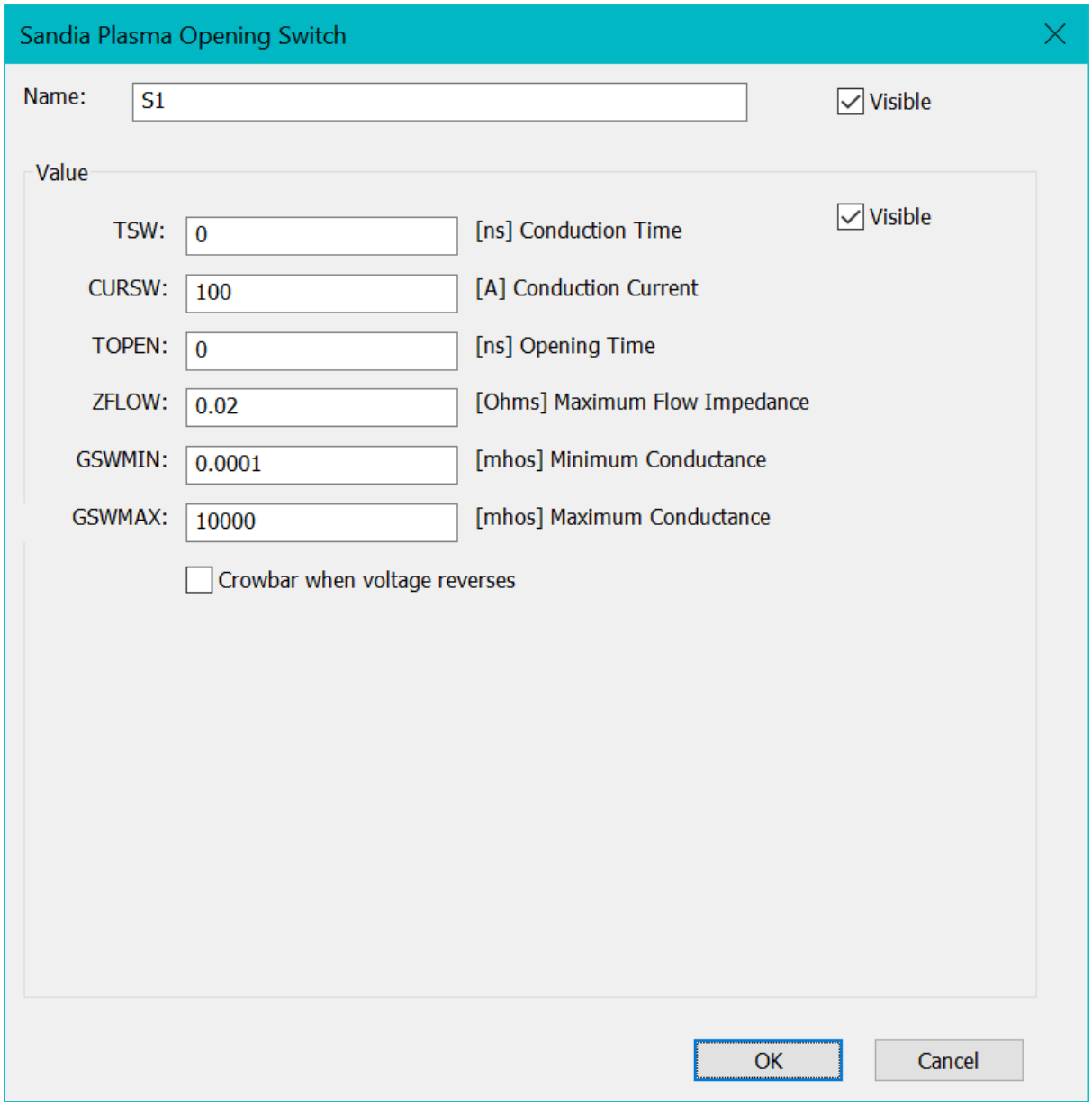
Critical Value: The value of the above criterion. For the Current criterion and Disk or Cyl geometries, this is in Amps/meter, with a nominal value of 5.e7. For the Current criterion and Arb geometry, this is in Amps. For the Time criterion, this time is in seconds.

© 2023 Raymond Allen

Pat Corcoran and Brandon Whitney from L3 Communications were helpful in implementing this routine.

7.29 SNL Plasma Opening Switch

This is another one of several components that is more or less cut and pasted from Bertha. Much of the text below is copied directly from the Bertha Help. One difference between the Bertha and CASTLE implementation is that this version uses absolute value of voltage and all currents.



© 2023 Raymond Allen

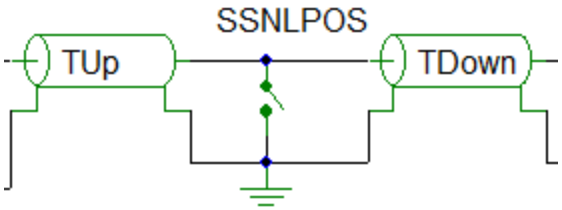
This subroutine implements the SNL plasma-opening-switch (POS) model. It is based on earlier Screamer code. A plasma-opening switch consists of plasma that bridges and shunts conductors in vacuum. Eventually, the plasma is either eroded or pushed away by jXB forces, and a gap develops between the plasma and conductor(s). At this point the switch resembles a short magnetically-insulated transmission line. This subroutine assumes that all vacuum flow in this small line is lost. The ZFlow loss model developed by Cliff Mendel (Phys. Plasmas 3, 4207 (1996), and ref.therein) is then used to calculate the flow, and hence the loss, current. This subroutine does not attempt to model the switch physics. Rather, the flow impedance ZFlow is assumed to increase linearly from the moment of switching to a given maximum value at a given time. This impedance may be thought of as a measure of the switch gap.

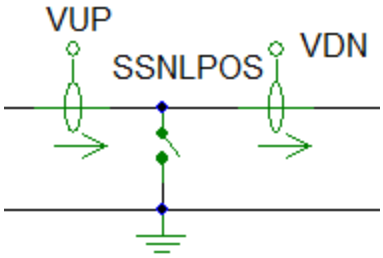
This routine does a minimum of two iterations.

(Future Work: For a similar subroutine with a fixed value of ZFlow, see the ZFlow subroutine.

SNLPOS 0 100 0 0.02 0.0001 10000 0

This routine is implemented as a version of a switch.
Note: Because this routine needs upstream and downstream currents, there is a special limitation on topology.
There must be either a transmission line or a current probe upstream and downstream of the switch. For TLines, the right end of connected TLine gives the upstream current and the left end of a connected TLine gives the downstream current. For current sources, the negative node is used for the upstream current and the positive node is used for downstream current.
Possible topologies are shown below:





TSW: The time threshold for switching. Note that both the current and time thresholds must be reached before switching begins.

CurSW: The current threshold for switching.

© 2023 Raymond Allen

TOpen: The time in ns over which switching occurs. This value is internally limited to a minimum of 0.01 ns.

ZFMax: The maximum flow impedance.

GSMin: The minimum switch conductance in inverse Ohms, limited internally to a minimum of 1 /Megohm.

GSMax: The maximum switch conductance in Inverse Ohms, limited internally to a maximum of 1/Micro-Ohm.

Before switching, the model has GSMax conductance.

Once switching begins, the junction resistance transitions instantly to 1/GSMax. The flow impedance increases linearly from 1/GSMax to ZFMax over a time of TOpen. The junction resistance is determined based on ZFLow and the upstream and downstream currents (see below), but is always limited by 1/GSMin and 1/GSMax.

In calculating the flow loss, the absolute values of the upstream and downstream currents are used. If the downstream current equals or exceeds that upstream, the flow current is set to zero and the switch resistance to 1/GSWMin. Otherwise, the loss is determined from

V = ZFlow * (Ia^2 - Ic^2)^0.5

where the anode current Ia is set to the upstream current and the cathode current Ic is set to the downstream current. The nominal junction resistance is then set to V / (Ia - Ic).

This subroutine offers the option of a crowbar upon voltage reversal. In this case, once the voltage drops below -1 kA, the switch is assumed to be fully closed, the flow current is set to zero, and the junction resistance is set to 1/GSWMax.

Crowbar Whether to crowbar the switch upon voltage reversal.

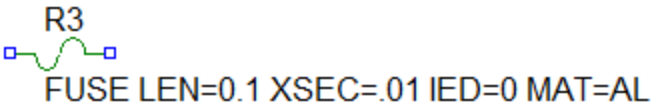
Future Work: Sw time The name of an optional output scalar containing the time when switching begins

Calc ZF The name of an optional output waveform containing the calculated value of the flow current.

© 2023 Raymond Allen

7.30 Fuse

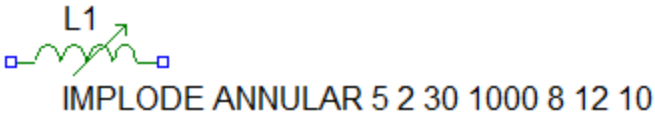
This model is for an aluminum fuse in sand.



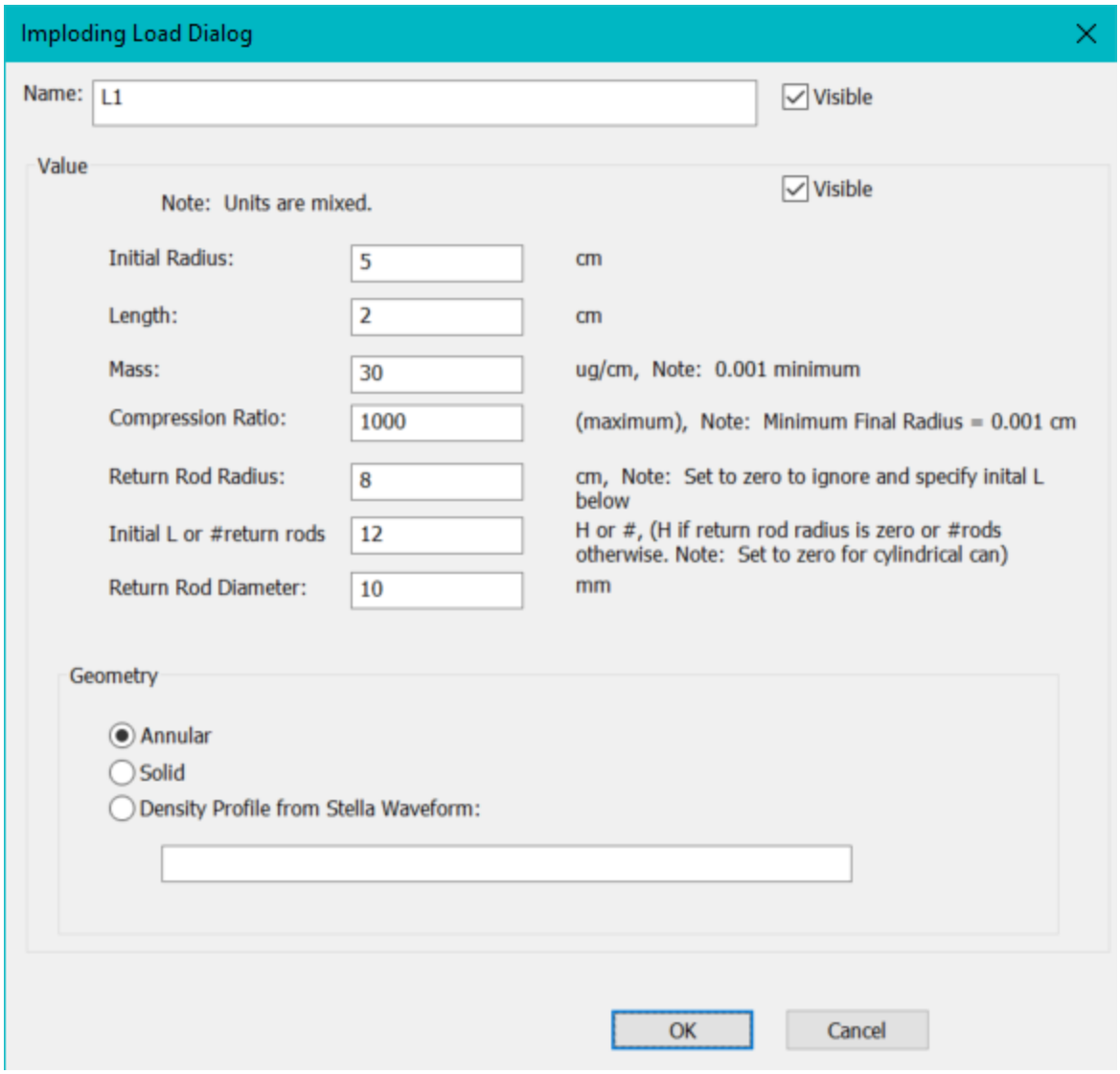
The parameters are "LEN", the fuse length in meters, "XSEC", the cross section in square meters, and "IED", the initial energy density in Joules per cubic meter.

7.31 Imploding Load

Models an imploding load.



© 2023 Raymond Allen



Special outputs include radius (cm), kinetic energy (J/cm), magnetic energy (J/cm), work (J/cm), and mass (g).

Note: This component was adapted from Bertha, and so is the following help:

This subroutine models a current-driven snowplow implosion, typically of a plasma radiation source (PRS). This models a cylindrical implosion. (Future: spherical implosions are modeled using the SphImpl subroutine.) The PRS is modeled by a variable-inductance element. Either annular (thin-shell), solid, or arbitrary initial density profiles can be modeled. The latter option is applicable in particular to modeling multiple thin shells.

Radius The initial radius in cm
Length The length in cm
ug/cm The mass in micrograms/cm. The code forces this to be at least 0.001 micrograms/cm.

© 2023 Raymond Allen

C ratio This is either the final compression ratio (defined as Radius/Rfinal), or the final radius in cm for the case of a user-supplied density profile (see below). The code forces the final radius in either case to be at least 0.001 cm. Three options are provided for the density profile: annular, solid, and user-supplied. The profile is entered as the Geom argument; for the user-defined case (see below), a waveform name should be entered here.
Geom The initial density profile, either Annular, Solid, or the name of a waveform containing the initial density profile.

For a user-defined profile, the density-profile 'waveform' is assumed to be the local volume density as a function of radius (determined by the start and time/pt) in cm. The supplied profile will be normalized to the value of ug/cm entered as the ug/cm argument, so that the absolute value of the input profile is not important. The profile is assumed to be in a stair-step so that multiple thin shells can be easily modeled. At the start of the calculation, the initial mass is calculated by integrating the profile from its outer boundary in to the value of Radius. If this is less than 0.0001 of the total mass, the initial radius will be reduced accordingly.

This routine can also model the inductance of the volume outside the initial radius.
Ret rad The radius of the return structure. If this number is <=0, this part of the calculation will be skipped.
rods The number of current-return rods. Enter 0 to model a cylindrical outer can
Rod dia The diameter of the return rods in mm. This is ignored for a cylinder (# rods = 0).

Future: t impl The name of the scalar output containing the time of implosion
Future: I impl The name of the scalar output containing the current at implosion
Future: KE impl The name of the scalar output containing the kinetic energy at the time of implosion
Future: Vl impl The name of the scalar output containing the velocity (in cm/s) at the time of implosion
Future: Ew impl The name of the scalar output containing the work (in J) at the time of implosion

Special Outputs:
Rad The name of the waveform output containing the radius in cm
L The name of the waveform output containing the inductance in H
KE The name of the waveform output containing the kinetic energy in J
ME The name of the waveform output containing the magnetic energy in J
Ewk The name of the waveform output containing the total work performed in J

 © 2023 Raymond Allen

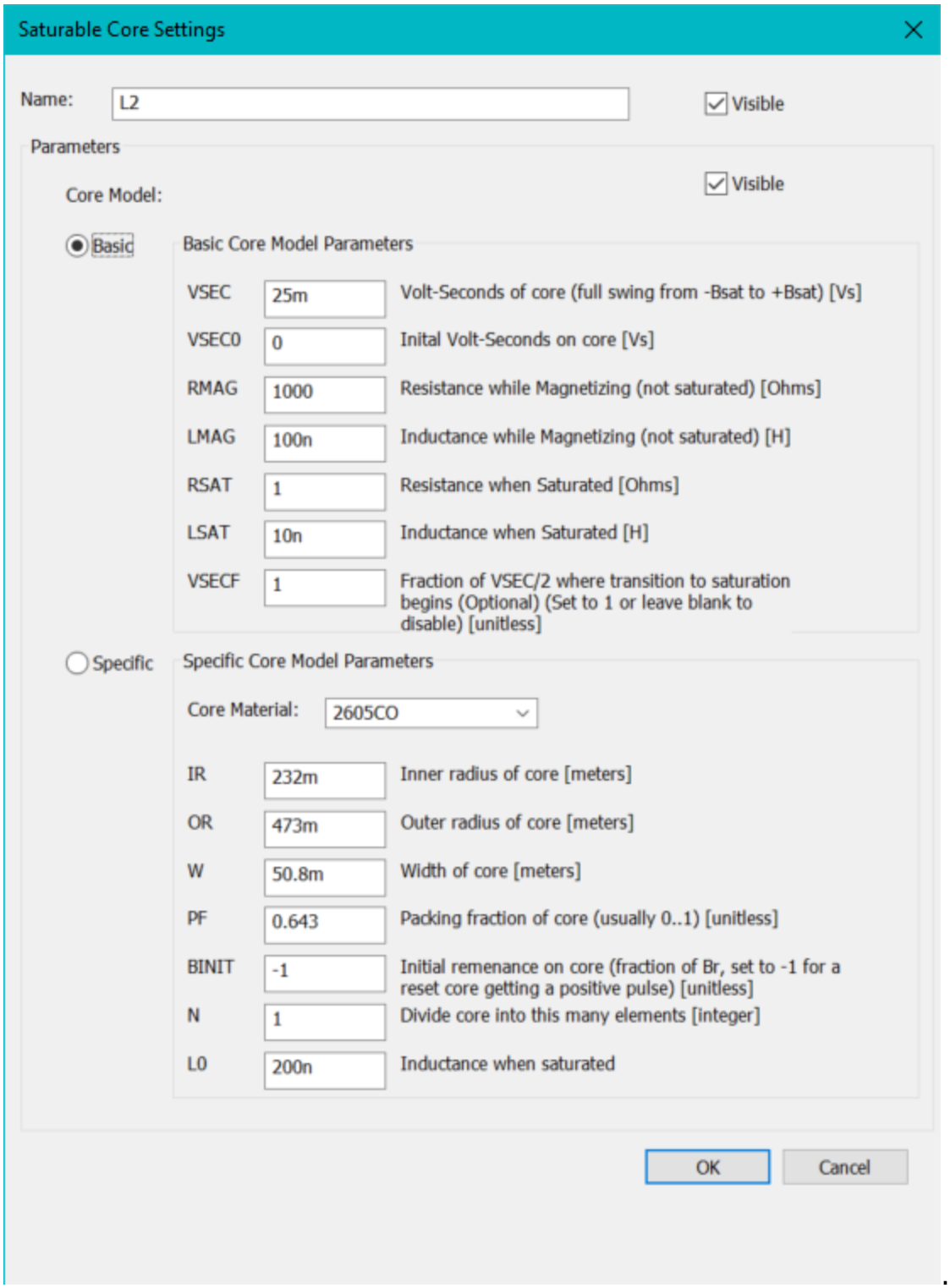
The calculation uses the standard equations for a perfect snowplow (all mass is caught up and moves with the same velocity, no pressure from material inside the snowplow radius) so that momentum is conserved. The kinetic energy is 1/2Mv^2, where M is the accreted mass. The magnetic energy is the instantaneous value of 1/2LI^2. The work is given by 1/2(dL/dt)I^2. The input electrical power (integral of IV) should be equal to Ewk + ME. The difference between Ewk and KE represents the energy dissipated in collisional heating within the imploding shell, and will be finite for solid or variable-density implosions.

7.32 Saturable Core

This is a basic model of a saturable inductive core.

L2
SATCORE MODEL=BASIC VSEC=25m VSEC0=0 RMAG=1000 LMAG=100n RSAT=1 LSAT=10n VSECF=1

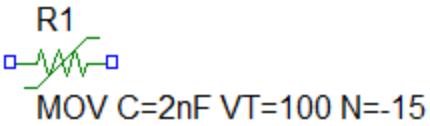
© 2023 Raymond Allen



© 2023 Raymond Allen

7.33 MOV

NOT YET IMPLEMENTED! The MOV (metal-oxide varistor) model exists in the schematic editor, but has not yet been implemented in the simulator and will give an error message if attempted to be used.



© 2023 Raymond Allen

This page is intentionally left blank.
Remove this text from the manual
template if you want it completely blank.

- G -

Grid 24

- K -

Keyboard 28

- M -

Main Window 18

- P -

Page Size 24
Parameters 35

- R -

Ribbon 18
Rotation 24

- S -

Select Tool 24
Shortcut 28

© 2023 Raymond Allen