Fundamentals

sT-Embed Training

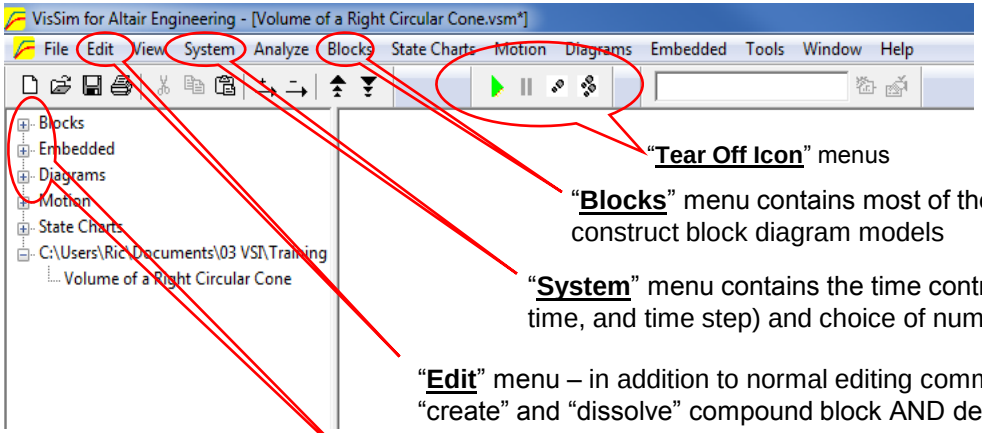
Ric Kolk
Altair Engineering
rkolk@altair.com

Topics:

- sT Embed workspace and terminology
- Simulation Setup & Control
  - Setup & Control options
  - Command line Startup
- Basic Model Blocks & Features
  - Compound Blocks
  - Model Levels
  - Signals, Wires, Parameters, & Pins
  - Pin addition & removal
  - Variables & scoping
  - Model Navigation
  - Compound Block Connector Labels
  - Wire Routing & Block Direction
  - Data Types
  - Scalar, Vector, Matrix Signals
  - Simulation Control
  - sT Embed built in variables
  - Simulation Setup
  - Signal Producers, Signal Consumers
  - Compound Block Enabled Execution
  - Adding a Compound Block to the sT Embed Menu
  - Compound Block Dialog Constants
- Plot Block & Features
  - Types of plots Interactive coordinate readout
  - FFT Plots
  - Interactive coordinate readout
  - Panning, zooming
  - Saving data to file
- Import & Export Blocks
- Signal recording & Playback
- Advanced Model Blocks & Features
  - Gain
  - Expression
  - Limit, Max, Min
  - Dialog Tables
  - Maps (1, 2, & 3 Dimensional)
  - Delayed Switch
  - Startup Script File
  - If – Then – Else using Merge and Compound Block with Enabled Execution

Window, Workspace, Key Menu's, Models, & Simulation

The sT Embed window consists of a menu bar containing tiers of dropdown commands, blocks, and examples and a workspace. Many of the menu bar command and blocks are also available in "tear off" icon bars (located under the menu bar) and in the tree structured "Menu & Browser window" to the left of the workspace.



"**Tear Off Icon**" menus

"**Blocks**" menu contains most of the building blocks you will need to construct block diagram models

"**System**" menu contains the time controls for simulating (start time, end time, and time step) and choice of numerical methods

"**Edit**" menu – in addition to normal editing command, this menu contains commands to "create" and "dissolve" compound block AND define user "preferences"

"**Menu & Diagram Browser**" – for model navigation & access to menu commands

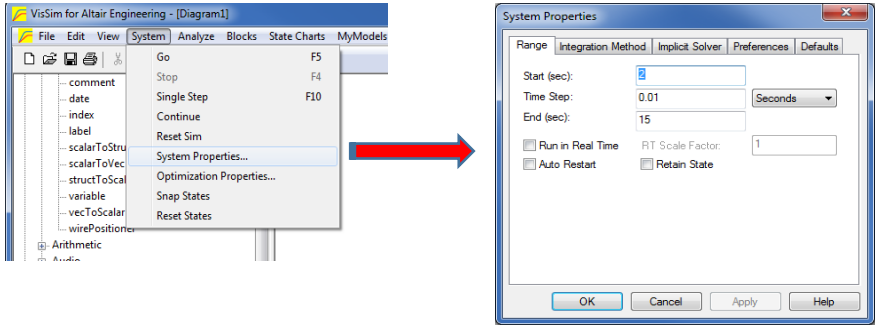
View the sT Embed Workspace

Simulation Setup and Control

Simulation Setup - Basics

sT Embed is designed to solve implicit, explicit, differential, & discrete equations that normally have time as their independent variable.

The simulation setup parameters are accessed through the “System/System Properties/Range” dialog box.



For most simulations, we are interested in setting three basic parameters:

Start (sec): Simulation Start Time, seconds, normally set = 0.

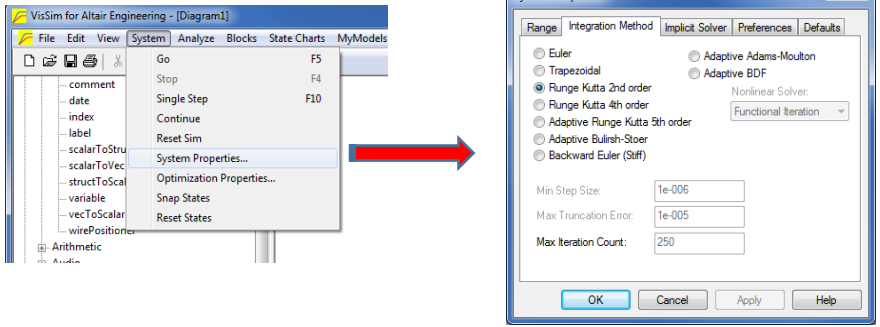
Time Step: Simulation Update Time, selectable units, normally in seconds. Time Step defines how often your sT Embed model is executed.

End (sec): Simulation End Time, seconds

Simulation Setup - Basics

Simulation Setup – Integrator Selection

sT Embed supports 9 integration methods available through the “System/System Properties/Integration Method” dialog box.

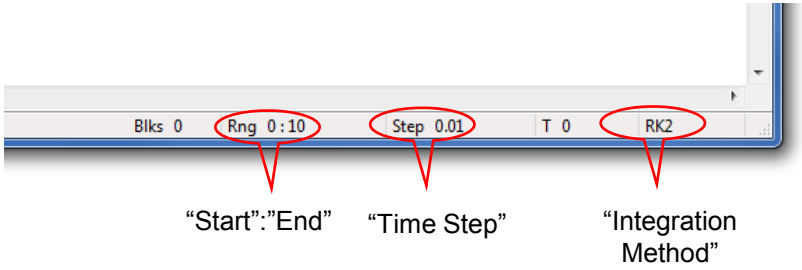


The first four integrators are fixed time step methods and the remaining are variable step methods. sT Embed defaults new models to use the “Runge Kutta 2nd order” integration method. This method provides a good compromise between speed of execution and accuracy of response.

You can easily see your selections “Start”, “Time Step”, “End”, and “Integration Method” in the lower right corner of the sT Embed Workspace Window.

Blks is the number of blocks in your model.

T is the elapsed simulated time (sec).



[Simulation Setup - Integrators]

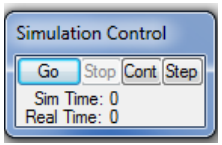
Simulation Control

Four commands to control the Simulation;

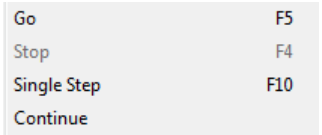
Start: Simulation starts at “Start (sec)” and executes until “End (sec)”
Stop: Stops the simulation, retains all memory values.
Single step: Executes the simulation for one “Time Step”, memory values are retained between single steps
Continue: Executes the simulation from the current time until “End (sec)”

There are three ways to access the commands:

Using the **Control Panel** located in “View/ Control Panel” menu.



Using the **Control Modes** located in “System” menu.



Using the **Control Mode Icons** located in toolbar.

Single Step Simulation Control Example

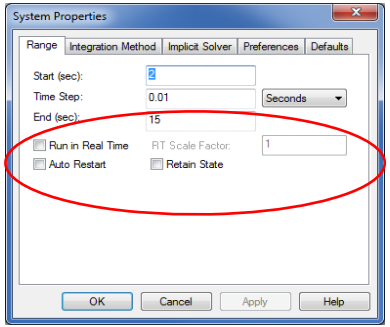
Simulation Configuration

sT Embed can be configured using three options available in the “System Properties/ Range” menu

Run in Real Time: Causes simulation execution to synchronize with clock time

RT Scale Factor: Used with “Run in Real Time” to control simulation execution speed. For example, setting “RT Scale Factor” = 10 will cause the simulation to run at 10x Real time

Run in Real Time Example



Auto Restart: Causes the simulation to continuously restart when “End (sec)” is reached

Auto-Restart Example

Retain State: Used with “Auto Restart” to restart with previous run memory values

Retain State Example

Command Line Startup

Customize sT Embed Startup with the following procedure:

1. From the DESKTOP, Right-click the **sT Embed** icon.

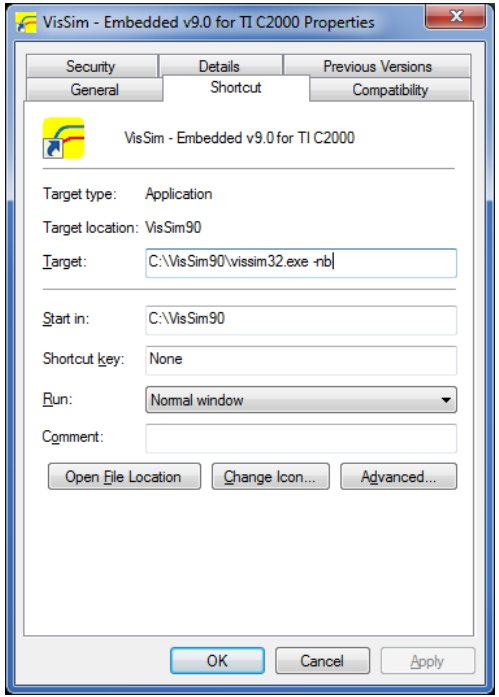
2. Click the **Shortcuts** tab and enter one or more of the following arguments in the “Properties/Shortcut/Target” box.

Use this argument	To
block-diagram-name	Start VisSim and open the specified block diagram.
-i [block-diagram-name]	Start VisSim as an icon and optionally open a block diagram.
-nb [block-diagram-name]	Start VisSim without the start-up banner and optionally open a block diagram.
-ne	Suppress the simulation completion dialog box.
-r block-diagram-name	Run a simulation read in from the specified block diagram upon start up.
-re block-diagram-name	Run a simulation read in from the specified block diagram and exit VisSim upon completion.

NOTE: If you enter more than one argument, separate them with spaces. If a model name is included in the argument list, **it must be specified last.**

Command Line Startup Example

Create a shortcut that launches sT Embed without the startup banner.



Basic Model Blocks and Features

EMBED™
solidThinking

Compound Blocks

A sT Embed model may be created on one level or many hierarchical sub levels

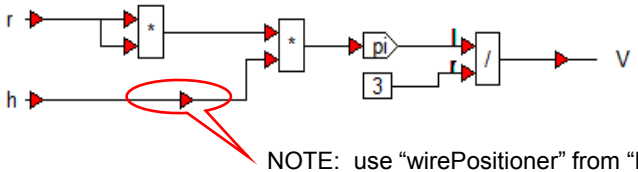
sT Embed **compound blocks** are used to encapsulate diagrams or parts of models into sub-level blocks. Compound blocks save screen space, hide unnecessary detail, can be easily replicated, and preserve calculations from accidental change.

After selecting the blocks to be encapsulated in a compound block, there are two ways to create a compound block

1. "Edit/Create Compound Block…" menu
2. Right Button Click, select "Create Compound"

Example:
The volume or a right circular cone is defined as: $V = \pi r^2 \frac{h}{3}$

This equation has two inputs, r and h, and one output, V, and may be written in block diagram form as:

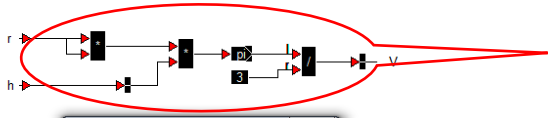


NOTE: use "wirePositioner" from "Blocks/Annotation" to position wire corners

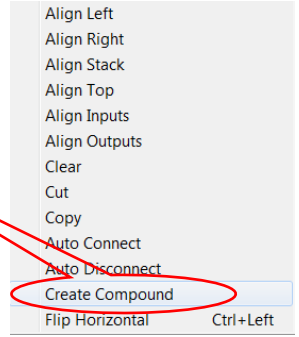
Right Circular Cone Equation

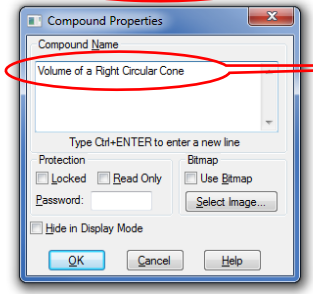
Compound Block Creation

Using “Edit/Create Compound Block…” the equation is captured in a compound block named “Volume of a Right Circular Cone”

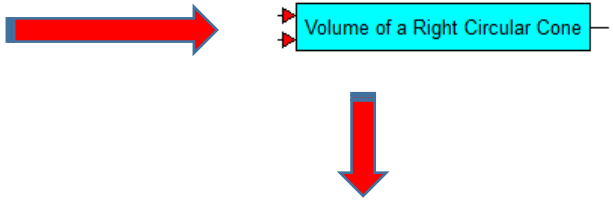


Step 1: “left button + drag” to select compound block contents, then right mouse to display shortcut menu, select “Create Compound”



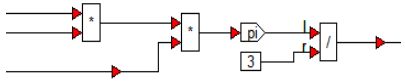


Step 2: Define compound block name, click “OK”



Our block diagram now has 2 levels:

Sub Level:



Associated Commands:
Flip Horizontal: rotate a compound block by 180 degrees (“Edit/Flip Horizontal”)
Dissolve Compound Block: undo the compound block creation (“Edit/Dissolve Compound Block)

[Create Compound Block Example]

Signals, Parameters, Wires, & Pins

Signals are real data that vary with respect to continuous time while **Parameters** are constant.

Signals travel on **wires** or via **variables** (described shortly)

Wire connections to any block (compound, sT Embed, …) are called **Pins** and defined as follows;

Input Pins;
Pin 1 (upper),
Pin 2 (lower)

Output Pin;
Pin 1

Pins are numbered from top down in ascending sequence [1,2,3…]

Pins only exist on the left and/or right sides of a block.

All Input pins lie on one side of a block and all output pins lie on the other side.

“Right Click” on a sT Embed block or a Compound Block will expose the **Block Parameters** OR the next sublevel of the compound block if it is a multilevel structure.

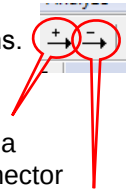
Compound Block Pins Example

Pin Addition & Removal

Compound Block Pins can be added two ways:

1. Automatically: From within a compound block, sT Embed will automatically add input and output pins such that all variables and outputs have pin connections.

2. Manually: Using either the “Add Connector…” or “Remove Connector…” commands located in the “Edit” menu or using the “toolbar” icons.



Adding an Input or Output Pin Manually:

1. Selecting “Add Connector…” from either the “Edit” menu or the toolbar icon.

2. The mouse cursor will change to ->+, place the cursor on either the left or right side of the compound block, “left click” as many times as needed to add inputs or outputs.

NOTE: connectors are always added from the pin on the lower side of the compound block edge.

Removing an Input or Output Pin Manually:

1. Selecting “Remove Connector…” from either the “Edit” menu or the toolbar icon.

2. The mouse cursor will change to ->-, place the cursor on either the left or right side of the compound block, “left click” as many times as needed to remove inputs or outputs.

NOTE: connectors are always removed from the pin on the lower side of the compound block edge.

Pin Addition and Removal Example

In addition to compound blocks, many of the blocks in the sT Embed library can have pins added to expose internal parameters.

Variables & Scope

sT Embed **variables** (“Blocks/Annotation/variable”) are used to create “wireless” connections for improved readability and conservation of screenspace. sT Embed variables are always connected to the pins of compound or other sT Embed blocks.

sT Embed **variables** may be scoped in any of three ways:

1. **Diagram Scope**: The variable is available at any level of the block diagram (also called globally scoped variables) Ex: “xDot”

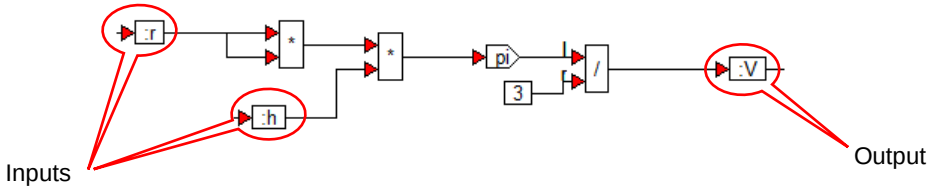
2. **Level (Local) Scope:** The variable is available ONLY at the level in which it was defined. Precede the variable name with a colon : for level scope. Ex: “:xDot”

3. **Definition Scope**: The variable is available at the level in which it was defined AND all sublevels. Precede the variable name with a double colon :: for definition scope. Ex: “::xDot”

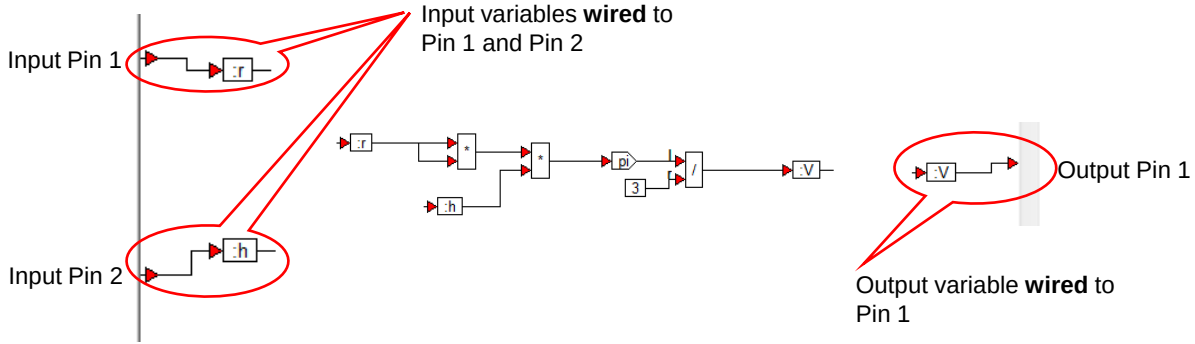
Variable Scope Example

Variable & Pins

1. Local Scope Variables are added for the input and output signals in the “Volume of a Right Circular Cone” compound block calculation:



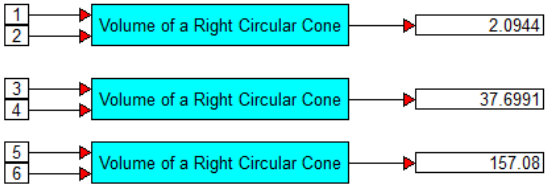
2. Wire the Local scope input and output variables by assigning them to the internal compound block pins;



[Locally scoped variables for Compound Block Example]

Compound Block Duplication

Replication of compound blocks with level scope variables preserves the compound block contents, for example, using the sT Embed copy command, “shift + ctrl + left button”, two additional copies of the “Volume of a Right Circular Cone” compound block are made. When supplied with different input signals, each copy produces a unique output.



Compound blocks intended to be replicated can be more efficiently designed if;

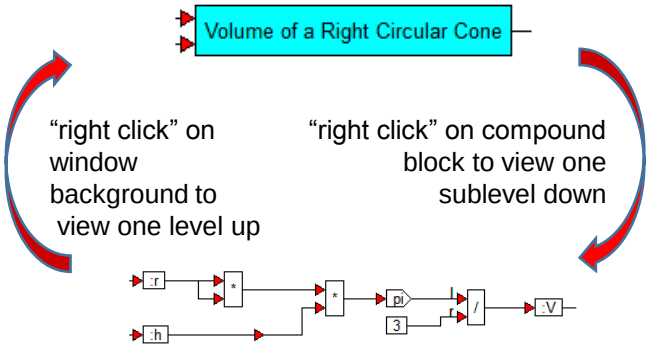
- Signals are communicated through the input and output pins
- Parameters are defined as internal variables with scoping that

Compound Block Duplication Example

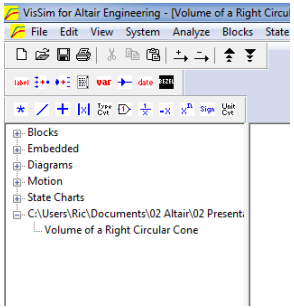
Compound Block Navigation

Compound Blocks and their sublevels may be navigated in either of two ways:

Mouse Navigation



“Menu & Diagram Browser” Navigation



The compound block name and its sublevels will appear in the “Menu & Diagram Browser” window on the left of the workspace.

In this example, click on the “Volume of a Right Circular Cone” to view the contents of the sublevel.

Compound Block Navigation Example

Multilevel Compound Blocks & Navigation

sT Embed supports multilevel compound blocks with no software limitations on the number of levels. For example, using a copy of the “Volume of a Right Circular Cone” compound block, a diagram with four levels of hierarchy is created and presented below;

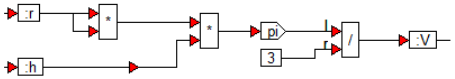
Our compound block now has 4 levels:

Top Level:

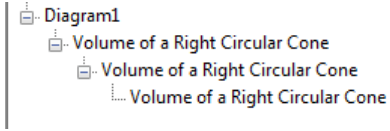
Sub Level 1:

Sub Level 2:

Sub Level 3:



“Menu & Diagram Browser” Navigation



The four levels of the compound block will appear in the “Menu & Diagram Browser” window, where;

Top Level = Diagram1
Sub Level 1 = Volume of a Right Circular Cone
Sub Level 2 = Volume of a Right Circular Cone
Sub Level 3 = Volume of a Right Circular Cone

“Left click” on any Level to view the contents of that level.

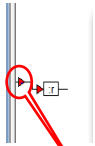
Create a multilevel Compound Block

Compound Block Connector Labels

For improved readability, compound block pins may be assigned names. Names may consist of digits and characters, special characters (other than alphabet characters) should not be used as they carry special meaning .

Pin names can be added either from within the compound block or at the top level of the compound block.

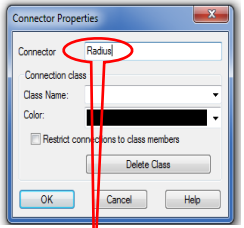
Adding a Pin name from within a compound block



Double left click on the red internal pin.

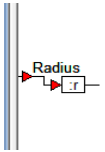
Adding a Pin name from the top level of a compound block

Double left click on the red pin.



Enter a Pin name

Connector name will appear inside the compound block

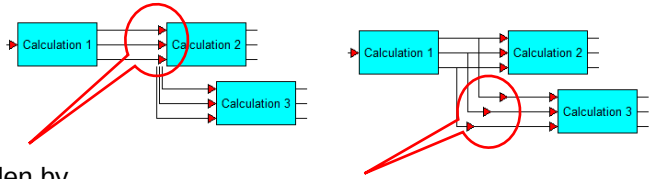


And ON the compound block top level

Adding Pin Labels to Compound Block Example

Wire routing & Block direction

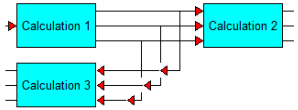
sT Embed will automatically choose the best wire path when connections are made between input and output pins. Sometimes it is necessary to manually define the wiring path. In these situations, the "**wirePositioner**" block in the ("Blocks/Annotation") menu is used.



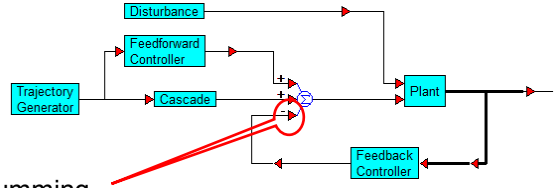
Wires partially hidden by "Calculation 2" compound block

"wirePositioner" blocks

sT Embed blocks are normally oriented to accept input signals on their left side and produce output signals on their right side. For feedback systems, use "ctrl + left arrow" to rotate any block by 180 degrees.



Example Feedback System: Illustrates the use of the "wirePositioner" block, block directions, and mixing of scalar and matrix signals in a block diagram. Note the use of "Add Connector…" to add an input to the summing junction block.



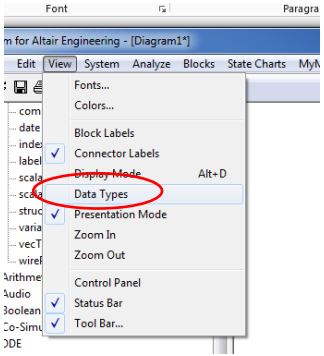
"ctrl – right click" to toggle the summing junction sign from + to – and back.

Wire Routing Example

Data Types

sT Embed automatically interprets data types and performs calculations without the need for any manual conversion.

You can view the datatype associated with each wire by selecting “Data Types” in the “View” menu.



sT Embed uses the following colors to display data types

Color	Data Type
Yellow	Scaled Integer
Light red	Float
Red	Double
Light magenta	Structure
Magenta	Matrix COMPLEX
Dark magenta	Matrix double or COMPLEX
Purple	String
Light green	Char
Green	Short, integer, or enum ComboItem
Dark green	Long
Light blue	Unsigned char
Blue	Unsigned short
Dark blue	Unsigned long or void

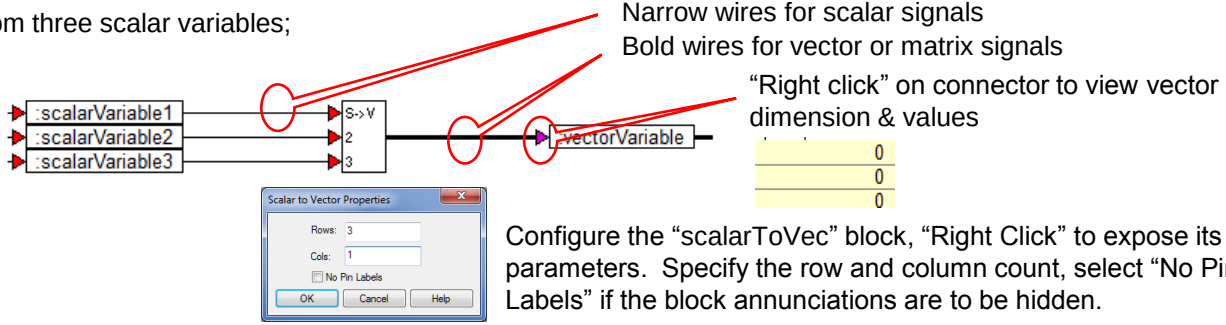
Data Type Example

NOTE For plots, meters, and stripChart blocks, the input connector tabs reflect a color for the signal trace and not the data type.

Scalar to Vector & Matrix Conversion

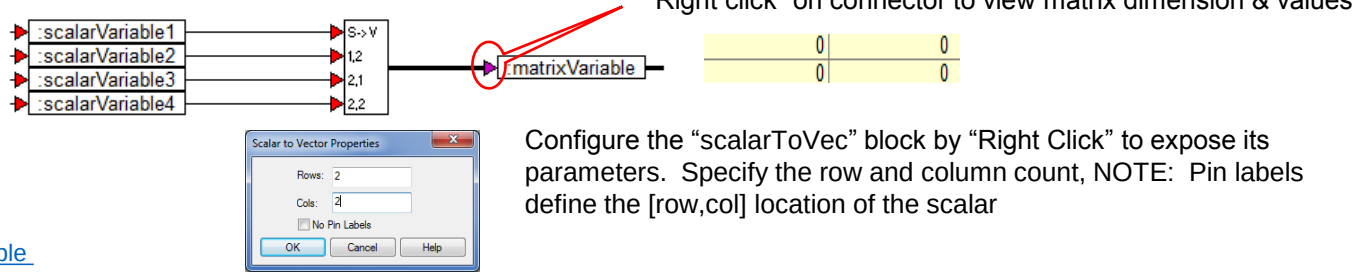
Scalar signals may be grouped into either vectors or matrices for conservation of screenspace, improved readability, or for matrix calculations (“Blocks/Matrix Operation”). The sT Embed the “scalarToVec” block (“Blocks/Annotation”) is used to create vector or matrix signals and the “vecToScalar” block (“Blocks/Annotation”) is used to create scalar signals from either vectors or matrices.

Create a 3x1 vector signal from three scalar variables;



Configure the “scalarToVec” block, “Right Click” to expose its parameters. Specify the row and column count, select “No Pin Labels” if the block annunciations are to be hidden.

Create a 2x2 matrix signal from four scalars:

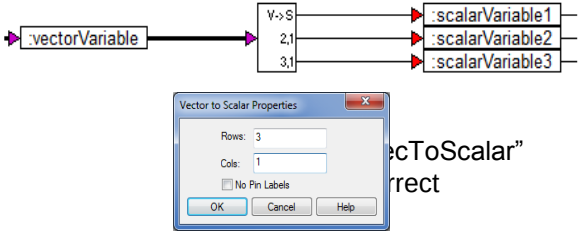


Configure the “scalarToVec” block by “Right Click” to expose its parameters. Specify the row and column count, NOTE: Pin labels define the [row,col] location of the scalar

Scalar Vector Conversion Example

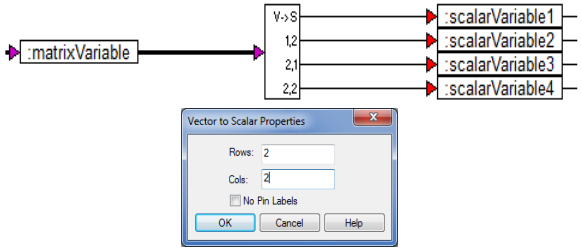
Vector & Matrix to Scalar Conversion

Create three scalar signals from a 3x1 vector signal;



ecToScalar"
rrect

Create four scalar signals from a 2x2 matrix signal;

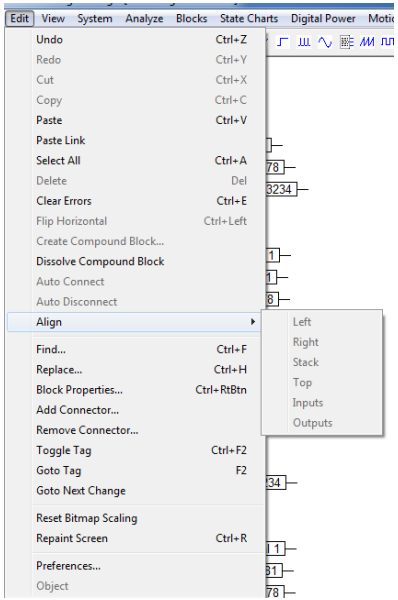


Scalar Vector Scalar Round Trip Conversion Example

Alignment Tool

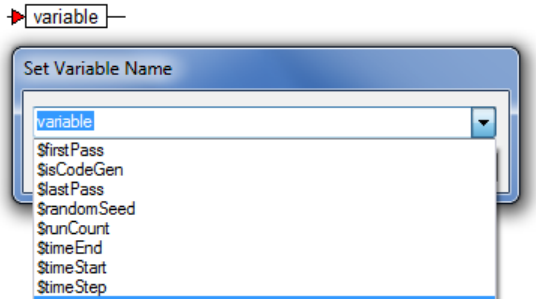
The block alignment tool is located in two locations:
1. “Edit/Align” menu.
2. Right Button Click on a selected collection of blocks

The following 6 commands are supported:
Left: Left justify
Right: Right justify
Stack: Stack all blocks vertically with no space between them
Top: Place all blocks on top of one another
Inputs: Align the “input” signals within a compound block with their connectors
Outputs: Align the “output” signals within a compound block with their connectors



Built in “Read Only” Variables

sT Embed provides 8 built-in variables that output either a **pulse** or a **value** for special simulation events or settings. sT Embed built-in variables begin with **$** and are accessed through the “**variable**” (“Blocks/Annotation”) block, dropdown menu.



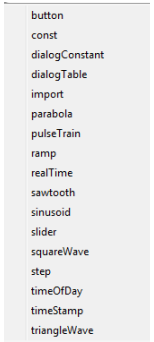
$firstPass: Outputs a pulse when the simulation starts.
$lastPass: Outputs a pulse at the last “Time Step” of the simulation.
$randomSeed: Outputs the random seed used by the random number generator.
$runCount: Outputs the simulation run number when “Auto Restart” is selected, begins with “1”
$timeEnd: Outputs the simulation “End (sec)” value.
$timeStart: Outputs the simulation “Start (sec)” value.
$timeStep: Outputs the simulation “Time Step” value.
.

Built In Variable Example

Signal Producers & Consumers

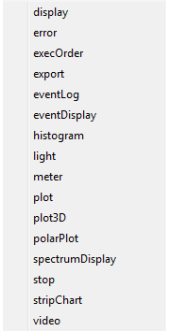
sT Embed provides two block menus to create signals (“Blocks/Signal Producer”) and to process signals (“Blocks/ Signal Consumer”);

Signal Producer blocks



Signal Producer blocks are used to create signals for use in a block diagram. Most can be used to create scalar OR matrix signals. Signal Producer blocks have output Pins only.

Signal Consumer blocks



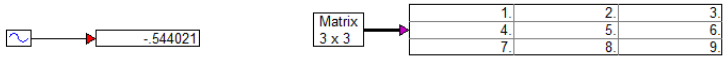
Signal Consumer blocks are used to display signal characteristics. Most are designed to receive scalar signals. Signal Consumer blocks have input Pins only.

Commonly Used Signal Producers

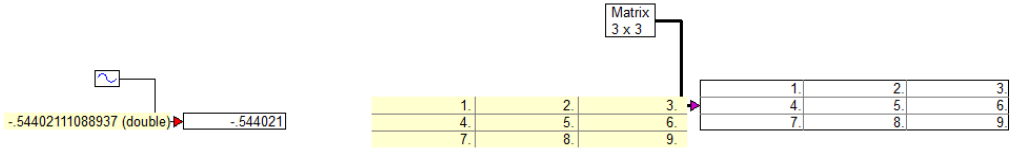
Commonly Used Signal Consumers

Wire Connector Data Display

Data can be displayed dynamically by adding a “display” block (“blocks/menu/signal consumer”) to any wire, scalar or vector.



Alternatively, data that passes through a signal connector can be displayed dynamically by placing the mouse over the connector and holding the mouse right button.



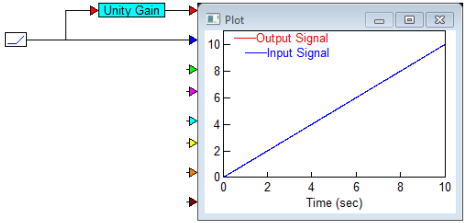
Display data does not have to be numeric, the following example displays an expression. The expression is must be created in a “const” block and must be enclosed in double quotes:

Compound Block Enabled Execution

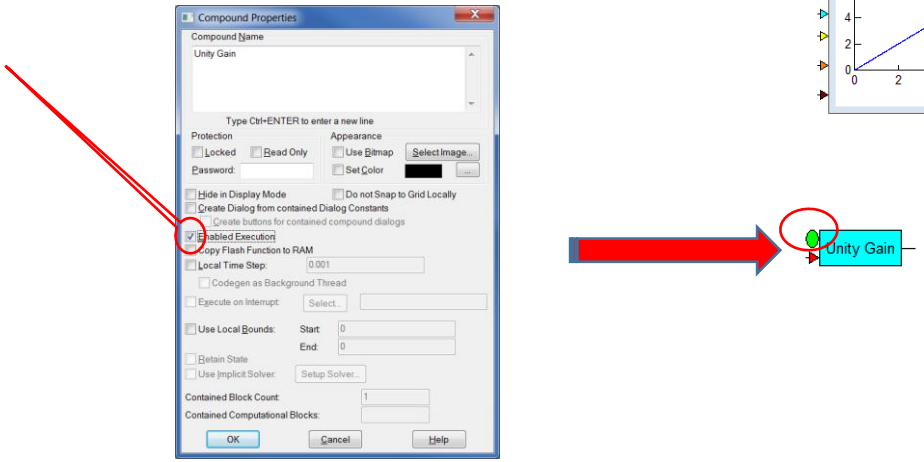
Normally, a sT Embed compound block will produce output at each “Time Step” of the simulation. In some situations it is necessary to alter the times at which a compound block’s output is produced. In these situations, the “**Enabled Execution**” option of the “Compound Properties” is checked (to access “Compound Properties” for any compound block, enter “ctrl + right button” on the compound block).

To illustrate, we’ll look at a compound block, named “Unity Gain”, that produces an output signal equal to it’s input signal times ONE. The simulation is configured with “Start (sec)” = 0, “Time Step” = ..01, and “End (sec)” = 10.

A unit “ramp” is applied to the compound block and both the input and output signals are plotted in a “plot” block (right).



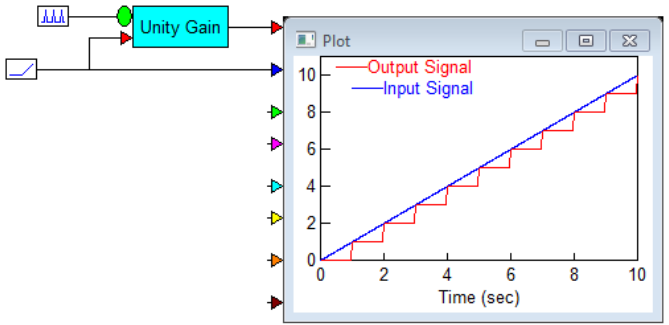
The “Enabled Execution” option of the “Unity Gain” compound block is checked, and a round green pin connector (named the “Enabled Execution” pin) appears on the input side of the compound block (right).



Compound Block Enabled Execution & External Trigger

A “pulseTrain” block (“Blocks/Signal Producer”) is configured with a “Time between pulses” set to 1 (units are in seconds). The “pulseTrain” is connected to the “Enabled Execution” pin and the simulation is executed (right).

The “Unity Gain” compound block is executed when the “Enabled Execution” pin input goes “high” or “1” and the compound block holds its previous value while the pin value is “low” or “0”. Since the “pulseTrain” was configured with a 1 second interval, the Output Signal is updated 1x/second every second as shown.



In addition to compound blocks, many of the blocks in the sT Embed library have the “Enabled Execution” pin function, however, it is named “External Trigger” in the block properties.

Compound Block Enabled Execution Example

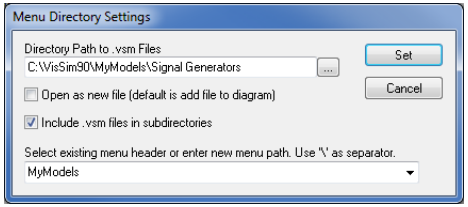
Adding a Model to the sT Embed Menu

Frequently used models can be added to the sT Embed toolbar menu for easy access.

For example, let's create a compound block named "Repeating Time", that produces a special signal & place it in the menu "MyModels/Signal Generators"

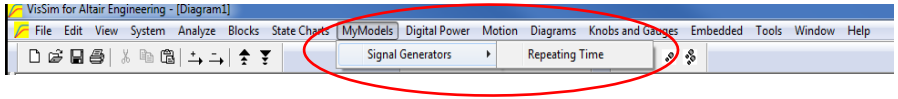
1. Create a "MyModels" folder in sT Embed90 with subfolder "Signal Generators"

2. Launch sT Embed EMBEDDED
   a. "Edit/Preferences/Menu Directories", double click "…"
   b. Browse to "MyModels/Signal Generators" folder
   c. Uncheck "Open as new file"
   d. Check "Include .vsm files in subdirectories"
   e. Enter "MyModels" for the menu header
   f. Click "set", "ok"



3. Save the "Repeating Time" sT Embed model in the "MyModels/Signal Generators" folder

4. Close & relaunch sT Embed EMBEDDED, the "MyModels" folder will be included in the sT Embed toolbar with a subfolder "Signal Generators" containing the "Repeating Time" model



Adding Dialog Constants to a Compound Block

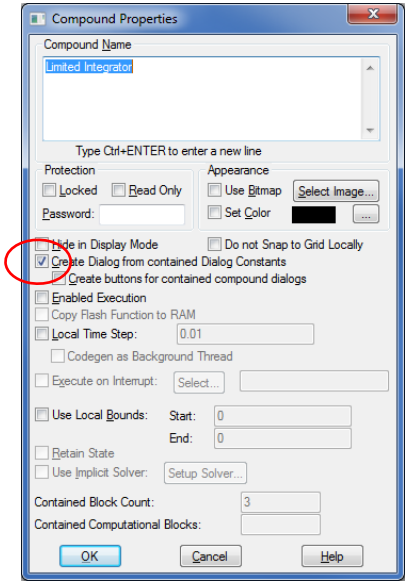
Variables within a compound block can be limited, defaulted, and presented through a “Dialog Window” when you Right Click on a compound block.

1. Identify the variables, select a “dialogConstant” block (“Blocks/Signal Producers”).

2. Enter the name you want to appear in the in the “Dialog Window” in the “name” field of the “dialogConstant” block.

3. Attach the “dialogConstant” block to the expression or variable needed

4. Create a compound block that encapsulates the “dialogConstants” and check the “Create Dialog from contained Dialog Constants”



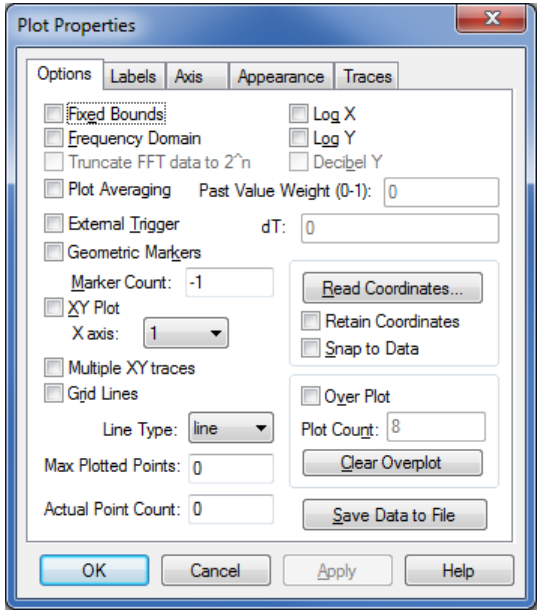
Dialog Constant Example

Plot Block and Features

Plot Block

Four types of plotting are supported:

- Time History Plotting: Plots up to 8 signal time histories simultaneously or in their subplot
- XY Plotting: Use the “XY Plot” option and specify the “X axis” signal.
- FFT Plotting: Use the “Frequency Domain” option to replot a time history plot as an FFT Plot.
- Vector Plotting: Using “External Trigger” option, the plot block can plot a vector of data versus it’s sample number (this is covered in “2. Signal Buffering & Vector Plotting”



Plot Block - Time History Example

Plot Block - XY Plotting Example

Plot Block - FFT

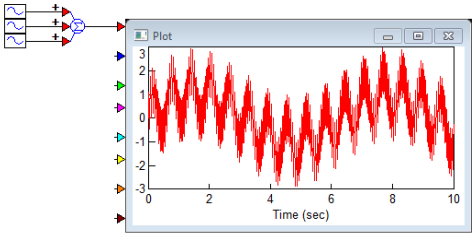
A signal is normally displayed as a time history in a sT Embed “plot” block. The frequency content of the signal can be displayed in the same “plot” block by selecting the “**Frequency Domain**” option located in the “plot/Options” tab.

This “Frequency Domain” option calculates and displays the Fast Fourier Transform (FFT) of the time domain signal. Signal power is displayed on the “y-axis” and frequency (in hertz) on the “x-axis”. The frequency range is dependent on the “Time Step” value used in the simulation. The **frequency range** begins at 0 Hz and ends at 2/”Time Step” value (where “Time Step” is in units of “seconds”).

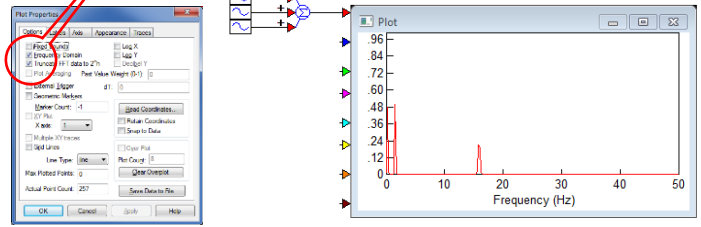
The sT Embed FFT algorithm requires that the signal time history point count is an integer power of 2. The “**Truncate FFT Data to 2^n**” option, when checked, will truncate the time history down to the nearest power of 2. If not checked, the time history is padded with zeros to round the point count up to the nearest power of 2.

The x and/or y axis scaling may be converted to “Log10” and db (10*Log10) by selecting the “**Log X**”, “**Log Y**”, or “**Decibel Y**” options in the “plot/Options” tab.

A complex waveform time history (below) simulated with “Time Step” = .01 seconds is shown below.



The “Frequency Domain” option reveals the signal consists of three fundamental frequencies.

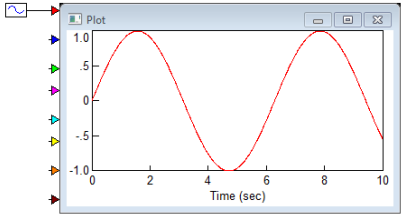


[Plot Block - FFT Example]

Plot Block – Interactive Coordinate Difference Readout

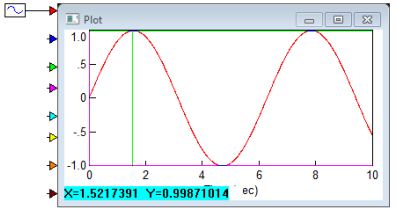
In addition to interactive coordinate readout, the “plot” block “Read Coordinate” feature can be used to display the **difference between an initial crosshair readout and the current crosshair readout.**

A “sinusoid” (“Blocks/ Signal Producer”) time history is displayed in a “plot” block (right).

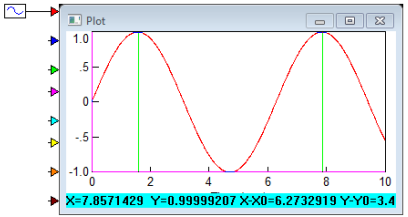


Using the Coordinate Difference Readout feature of the “plot” block, the period of the sine wave can be measured;

1. Create the Initial crosshair readout: “right+click” on the “plot” block, “Options” tab, check the “Snap to Data” option and then click “Read Coordinates”, then “OK”. Place the crosshair at a peak value of the sinusoid, click the left mouse button.



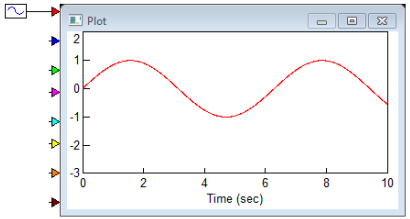
2. “right+click” on the “plot” block, “Options” tab, check the “Retain Coordinates” option and then click “Read Coordinates”. Place the crosshair at an adjacent peak value of the sinusoid, the initial crosshair readout values are displayed in the lower left corner of the “plot” block followed by the difference between the current and initial readout values. In this case the sinusoid has a period of 6.2732919 seconds.



Plot Block - Interactive Coordinate Readout

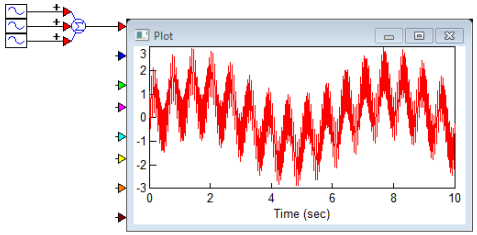
Plot Block – Bounds, Pan, & Zoom

The “plot” block axis ranges are automatically rescaled dynamically by sT Embed as the simulation evolves. Axis can be manually adjusted by checking the “**Fixed Bounds**” option under the “plot/Options” tab and specifying the “Y Upper Bound”, “Y Lower Bound”, “X Upper Bound”, or “X Lower Bound” located under the “plot/Axis” tab. For example, the Y axis of the sinusoid plot (previous) is manually set to the range [-3, 2] (right).

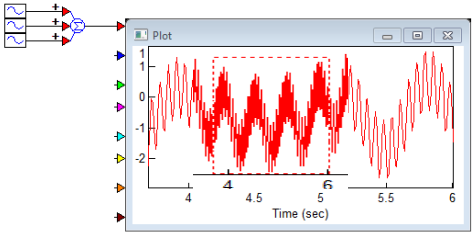


Plot **zooming** and **panning** are additional important features of the “plot” block.

Beginning with a complex waveform time history (below)



A **zoom** rectangle is selected by placing the mouse on the plot window, click and hold the “ctrl + left button” at one vertex of the rectangle, drag the mouse to the opposite vertex, release, and the zoomed rectangle will fill the “plot” block.



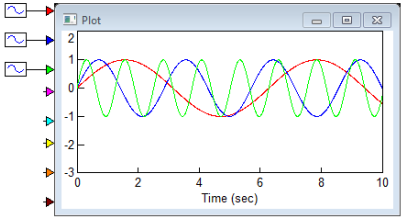
The zooming operation may be repeated as many times as needed, crosshairs can be used to readout coordinates, use “ctrl + right button” to **pan** back to the original plot

Plot Block Pan and Zoom Example

Plot Block – Saving Data to File

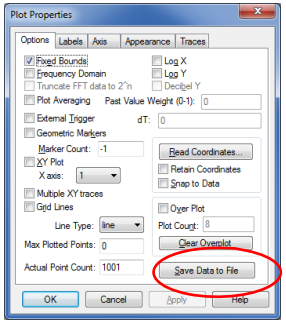
Time history signals displayed in the “plot” block can be saved to files of various types including ASCII “.txt”, “.dat”, and “.csv”.

In the block diagram (right), the “plot” block presents the time histories of the three signals attached to its pins.

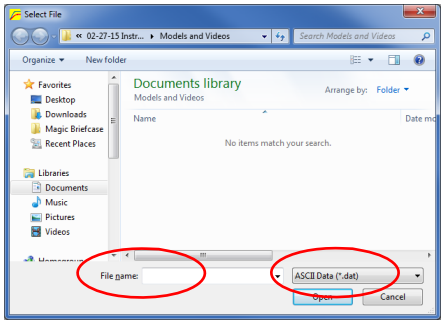


To **save the time history signals to a file**, the following two steps are performed;

Step 1. “right click” on the “plot” block to expose it’s properties, then, under the “Options” tab, click “Save Data to File”.



Step 2. A standard Windows “Select File” window appears, enter the file name, then “Open”



Plot Block Save Data To File Example

Plot Block – Interpreting a Saved Data File

Navigating to the saved file location and opening the file using notepad or wordpad, the following features are noted:

[Start (sec), Time Step, End (sec)] values set in “System/System Properties”

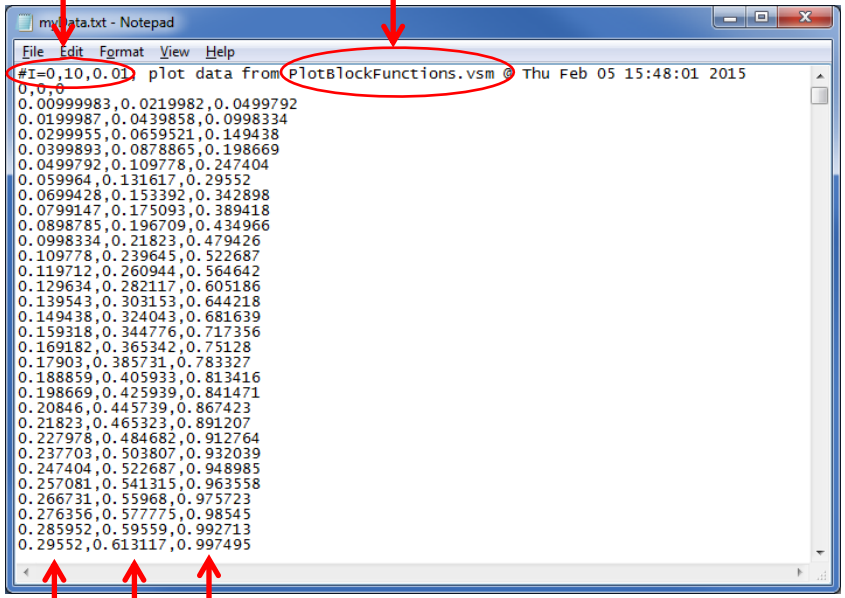
sT Embed file name that produced the data

Line 1: File Header Line

Line 2: signal values at time = “Start (sec)”

Lines 3 to (n-1): Time increases for each descending row by the “Time Step value”

Line n: signal values at time = “End (sec)”



Pin 1 time history data

Pin 2 time history data

Pin 3 time history data

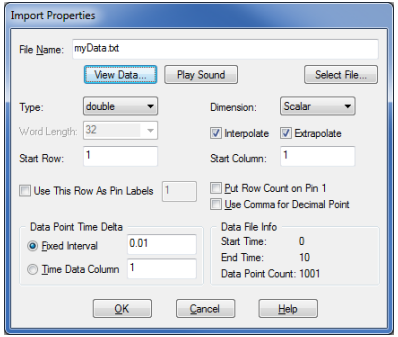
Import and Export Blocks

EMBED™ solidThinking

Import Block

Data files can be imported into any sT Embed diagram using the “**import**” block located in the (“Blocks/Signal Producer”) menu (right).

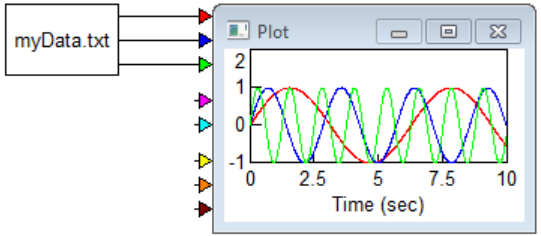
The “import” block is used to import the “plot” data saved in the previous example from the file “myData.txt” into a new sT Embed diagram. Access the “import” block properties with “ctrl + right click” and select the “myData.txt” file (right).



The “import” block detected that this data file was produced by sT Embed and has set the “Data Point Time Data” to “Fixed Interval” = .01 seconds (the “Time Step” that was used to generate the data.

Interpolate and Extrapolate are also selected in case the “Time Step” in the new sT Embed diagram differs from the “Time Step” used to collect the data.

Connecting the “import” output pins to a “plot” block and simulating results in the time histories that were originally recorded from the “plot” block in the previous example. (right).

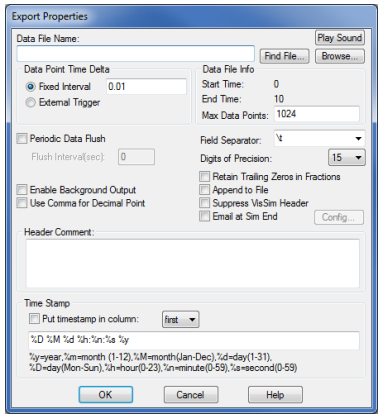


Export Block

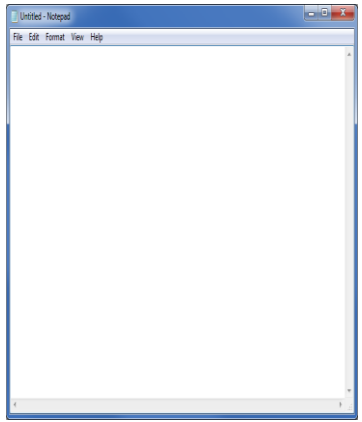
Data files, consisting of signal time histories produced from a sT Embed simulation, can be exported into a file using the "**export**" block located in the ("Blocks/Signal Consumer") menu (right).

The "export" block is configured to accept signal data in **three** basic steps;

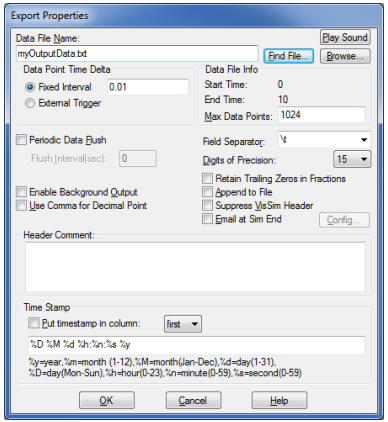
Step 1. "right click" on the "export" block to expose it's properties



Step 2. click "Browse", when "notepad" appears, save it with the export filename and type, close "notepad"



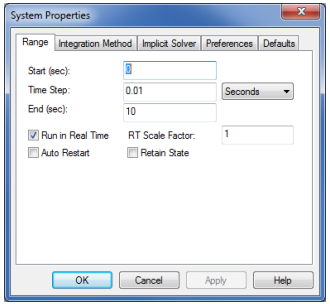
Step 3. click "Find File…" and select the file created in Step 2.



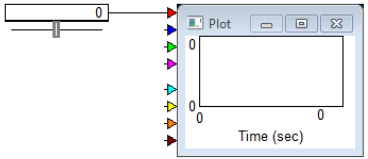
Example – Signal Recording & Playback

The interactive nature of sT Embed provides many unique features. This example illustrates how a user generated signal can be recorded in a file and later used for play back using an **import** block. The signal is generated by a sT Embed "**slider**" ("Blocks/Signal Producer") and plotted in a "plot" block configured for **two subplots** and **overplot.**

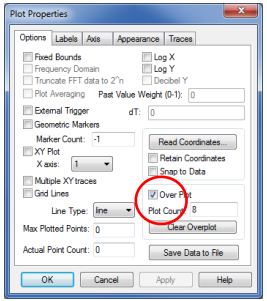
Step 1: the simulation is configured to run with [Start (sec), Time Step, End (sec)] = [0, .01, 10] and the "**Run in Real Time**" option is selected (this forces the simulation to run synchronized with "clock time" giving the user enough time to create a signal using the "slider".



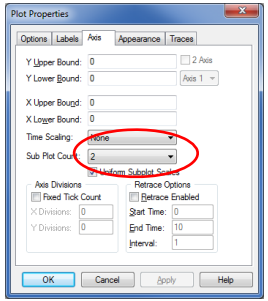
Step 2: A "slider" and "plot" blocks are added and connected as follows;



Step 3: The "plot" block is configured with the "Over Plot" option selected ("Plot Properties/Options") tab.



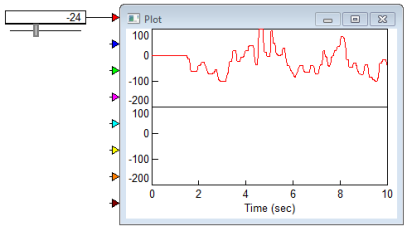
Step 4: The "plot" block is configured for two subplots in the ("Plot Properties/Axis") tab.



Note: When checked, "Uniform Subplot Scales" causes all subplots to use the same axis scaling. If independent scaling is desired, uncheck this option.

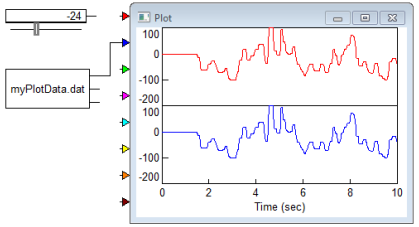
Example – Signal Recording & Playback

Step5: With the simulation running, move the “slider” control randomly, observe the signal time history in the upper subplot of the “plot” block.



Step6: Save the time history to a file using the “Save Data to File” option in the “plot” block “.

Step7: Add an “import” block to the diagram, configure the “import” block to read the file named in Step 6. Connect pin 1 of the “import” block to pin 2 of the “plot” block. DISCONNECT pin 1 of the “plot” block, otherwise an additional signal equal to the present “slider” value will be added. Run the simulation



The “export” block could have been used in place of the “plot” block to capture the “slider” signal to a file. Aside from the block, the procedure is the same.

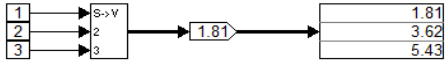
Import Block Playback Example

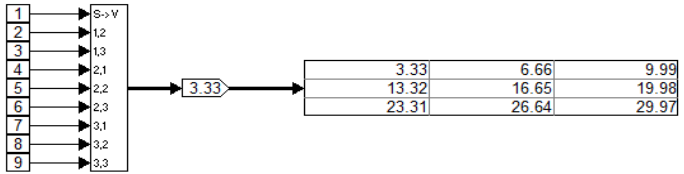
Advanced Model Blocks and Features

Gain Block

The “gain” block (“Blocks/Arithmetic”) is used to apply a gain to a signal, vector, or matrix.

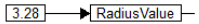
Numerical Gain Example

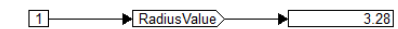


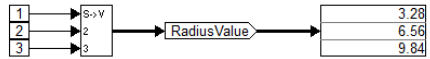


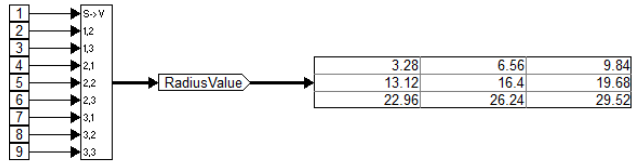
The “gain” value may be either numeric or a variable. (NOTE: the gain block is evaluated once at the start of the simulation, if the variable used for the gain changes dynamically, it will NOT change in the “gain” block). NOTE: In this situation, use the “*” block and a “variable” block to capture the gain value.

Variable Gain Example



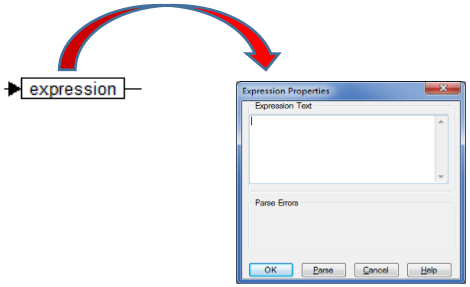






Expression Block (1/2)

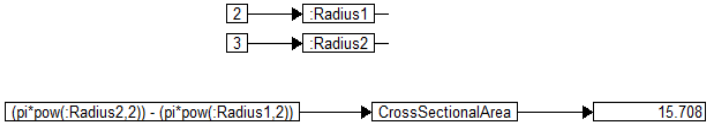
The “expression” block (“Blocks”) allows you to enter C expressions using constants, sT Embed Variables, or input Pin values. Pin values are accessed using the format “$n” where n = Pin number.



The expression block is evaluated at each time step. After you enter the expression, click Parse to check for errors.

NOTES:
- The expression block only accepts scalar data
- The expression block produces only one output
- An expression (in C) is code that evaluates to a value
- Expressions are not terminated with ;
- Expressions may have multiple lines

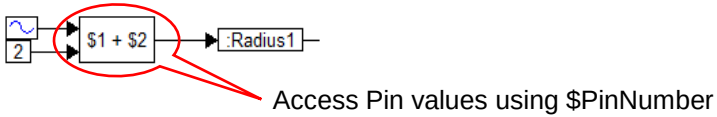
Example using sT Embed static variables: In this example the cross sectional material area is calculated for a pipe defined with an inner radius (:Radius1) and an output radius (:Radius2). Both radii are fixed.



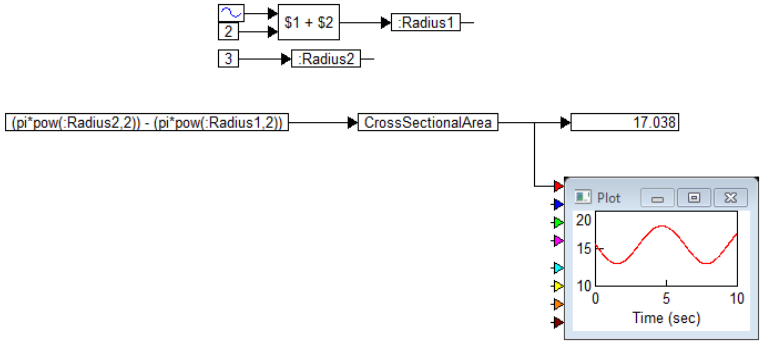
Expression using Variables

Expression Block (2/2)

In addition to sT Embed Variables, an expression block may also use input Pin values for its calculation. In this model the expression block is being used to sum Pins 1 and 2:



Example using sT Embed dynamic variables: In this example, :Radius1 is dynamically varied during the simulation:

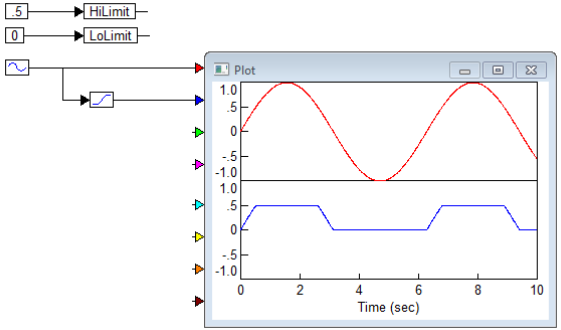


Expression using Dynamic Variables

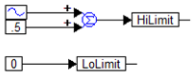
Limit, Max, & Min Blocks

The “Limit” block (“Blocks/Nonlinear”) provides signal limiting using Limit values that remain constant for the duration of the simulation.

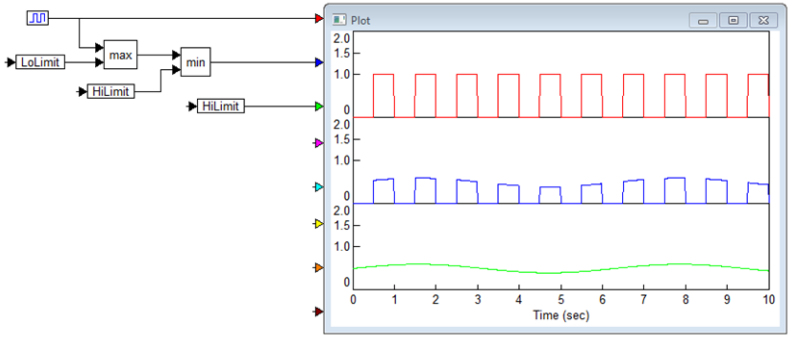
Limit Example



For dynamic limits, use the “Max” and “Min” blocks (“Blocks/Nonlinear”)

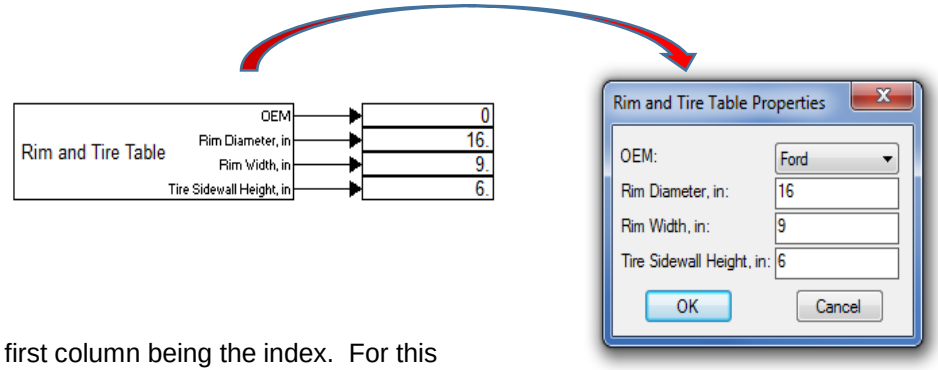


Min Max Limit Example



Dialog Table Block (1/2)

The “Dialog Table” block (“Blocks/Signal Producers”) provides an interface to tabular data that is indexed by the first column.



Steps:

1. In EXCEL, define the table with the first column being the index. For this example, the following table was constructed.

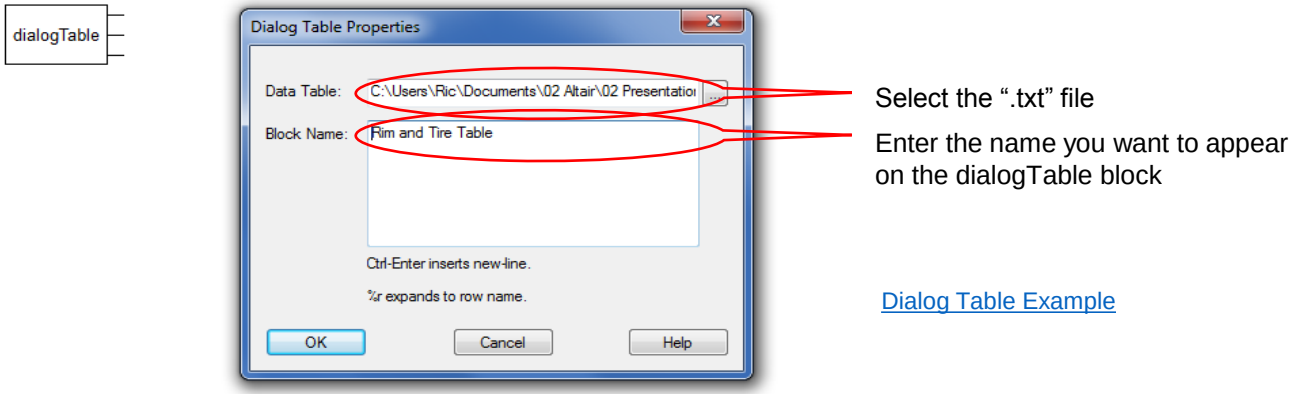
OEM	Rim Diameter, in	Rim Width, in	Tire Sidewall Height, in
Ford	16	9	6
GMC	17	10	7
Dodge	16.5	11.5	5
Toyota	22	14	6.5
Nissan	12	7.5	3.3

EXCEL table

Dialog Table Block (2/2)

2. In EXCEL, save the table file in “Text (Tab delimited) (*.txt)” format.

3. In sT Embed, place a “Dialog Table” block on the screen (“Blocks/Signal Producers”), right mouse to select the “.txt” file from step 2.



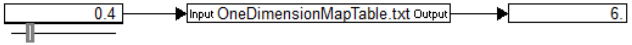
Values from the “dialogTable” output Pins can be used as parameters in your sT Embed model

Map Block (1/7)

The “map” block (“Blocks/Nonlinear”) provides a table lookup function that can be configured for tabular data using 1, 2, or 3 independent variables. The independent variable(s) must be monotonically increasing or decreasing.

Map Block: One Independent Variable example:
A one-dimensional map file has one independent variable and can have from 1 to 16 dependent variable outputs. A one-dimensional matrix is limited to 8,000 rows. Lines that are prefaced with a semi-colon (;) are treated as comments. A one output map example is presented below.

Steps:



1. In EXCEL, define the map with the first column being the independent (input) variable and the second column being the output variable. If you want connector labels, include them in the first row.. For this example, the following map was constructed in EXCEL:

Input	Output
0.1	2
0.2	4
0.3	5
0.4	6
0.5	7

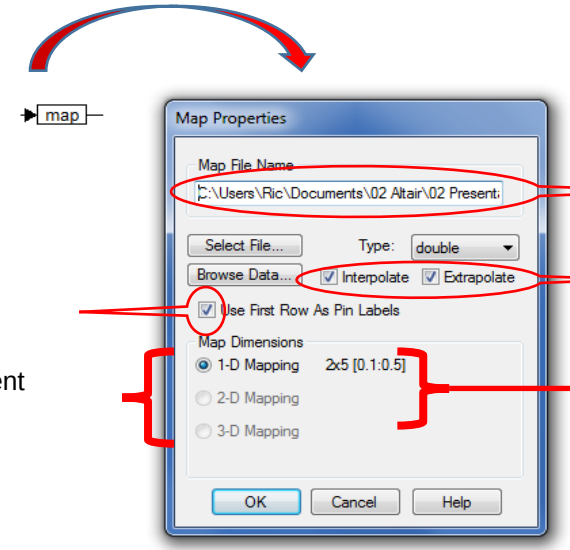
EXCEL One D Map File

First column is the first independent variable (Pin 1)

2. In EXCEL, save the map file in “Text (Tab delimited) (*.txt)” format.

Map Block (2/7)

3. In sT Embed, place a “map” block on the screen (“Blocks/Nonlinear”), right mouse to select the “.txt” file from step 2 and configure as follows:



Select the “.txt” file

Check if labels are desired

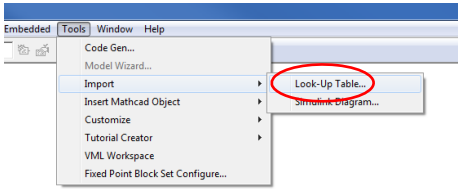
Select for linear interpolation and extrapolation

Select # independent variables

2x5 is the map size (col, row)

(0.1:0.5) is the range of the independent variable

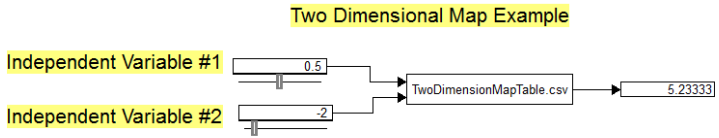
NOTE: sT Embed also provides a “Lookup Table Wizard” (right). Assuming that tabular data is available, the wizard automatically guides you through the setup procedure.



[One Dimensional Map Example]

Map Block (3/7)

Map Block: Two Independent Variable example:
A two-dimensional map file has two independent variables and one dependent variable output. The first row contains the first independent variable; and the first column (excluding the column member in row 1) contains the second independent variable. The position (1,1) must be left blank. A two-dimensional matrix is limited to 90 rows by 90 columns. Lines that are prefaced with a semi-colon (;) are treated as comments.



Steps:

1. In EXCEL, define the map with the first column being the independent (input) variable and the second column being the output variable. If you want connector labels, it is easiest to add them as "connector labels" on the "map" block after it is loaded into your sT Embed model. For this example, the following map was constructed in EXCEL:

	0.1	0.2	0.3	0.4	0.5
-3	1	2	3	4	5
0	1.2	2.3	3.8	3.3	5.7
3	0.9	1.1	1.3	1.21	2.1
7	-6	0	1.2	2.2	3
10	0.89	2	3.2	4.1	3

First row is the first independent variable (Pin 1)

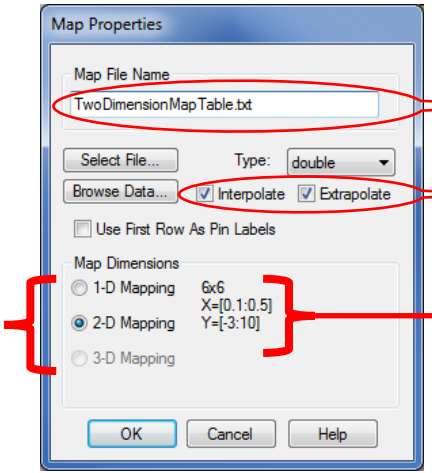
First column is the second independent variable (Pin 2)

2. In EXCEL, save the map file in "Text (Tab delimited) (*.txt)" format.

EXCEL Two D Map File

Map Block (4/7)

3. In sT Embed, place a “map” block on the screen (“Blocks/Nonlinear”), right mouse to select the “.txt” file from step 2 and configure as follows:



Select the “.txt” file

Select for linear interpolation and extrapolation

Select # independent variables

6x6 is the map size (col, row)

X=(0.1:0.5) is the range of independent variable #1

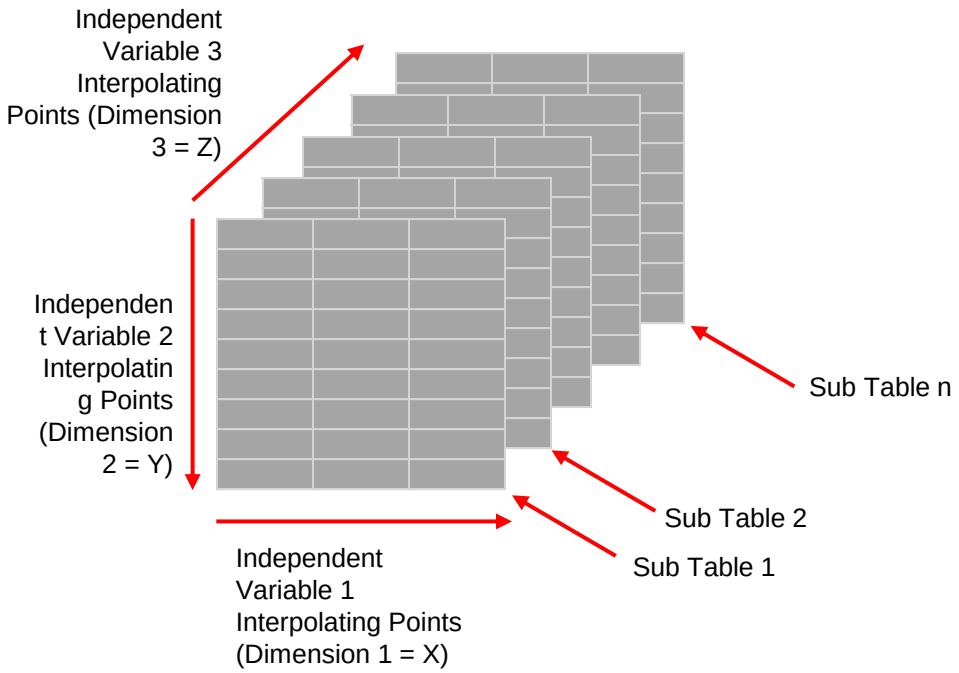
Y = (-3:10) is the range of independent variable #2

Two Dimensional Map Example

Map Block (5/7)

Map Block: Three Independent Variable example: A three-dimensional map file has three independent variables and one dependent variable output.

It is convenient to think of the 3D table data in the following format



Map Block (6/7)

The first seven lines in the 3D map data file must contain the following information:

Line 1: #3D
Line 2: Size of dimension 1
Line 3: Interpolation points for dimension 1
Line 4: Size of dimension 2
Line 5: Interpolation points for dimension 2
Line 6: Size of dimension 3
Line 7: Interpolation points for dimension 3

Lines 8 and beyond contain the Sub Table 1, Sub Table 2, …

NOTES:
1. Use ";" to comment a line
2. #3D MUST be the first line (no comments before this line are allowed)

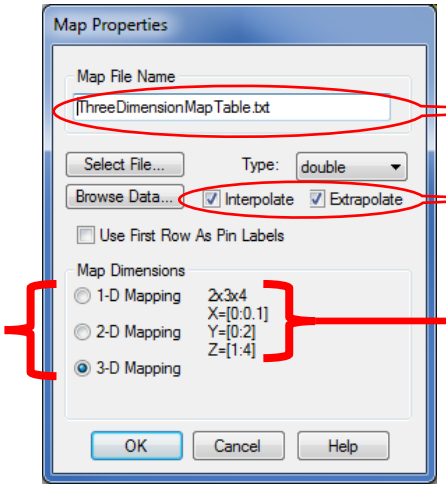
Steps:

1. In EXCEL, define the map using the 7 line header format described above.
2. In EXCEL, save the map file in "Text (Tab delimited) (*.txt)" format.

EXCEL Three D Map File

Map Block (7/7)

3. In sT Embed, place a “map” block on the screen (“Blocks/Nonlinear”), right mouse to select the “.txt” file from step 2 and configure as follows:



Select the “.txt” file

Select for linear interpolation and extrapolation

Select # independent variables

2x3x4 is the map size (col, row,depth)

X=(0:0.1) is the range of independent variable #1

Y = (0:2) is the range of independent variable #2

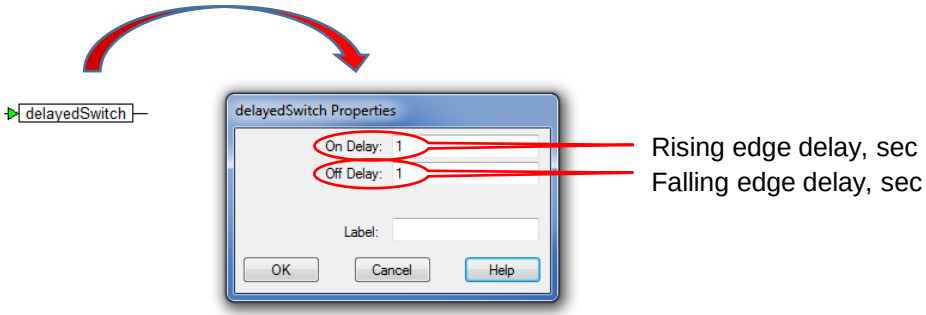
Z = (1:4) is the range of independent variable #3

Three Dimensional Map Example

Delayed Switch

The “delayedSwitch” (“Blocks/Nonlinear”) accepts a binary signal and produces a binary signal that can be configured to;

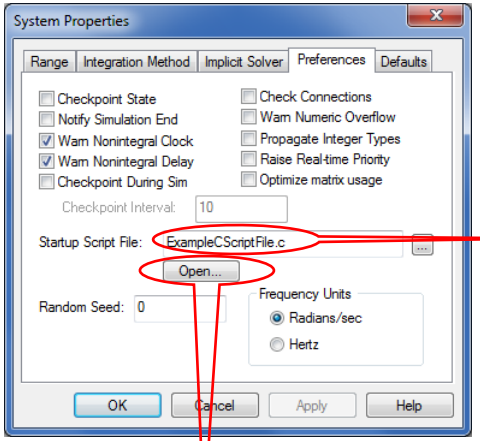
- Delay its rising edge
- Delay its falling edge
- Delay both its rising and falling edge



Delayed Switch Example

Startup Script File (1/2)

The “Startup Script File” located under (“System/System Properties/Preferences”) lets you initialize sT Embed “const” and “gain” blocks used in your sT Embed model. The start-up script is a C file that contains names and C expressions.

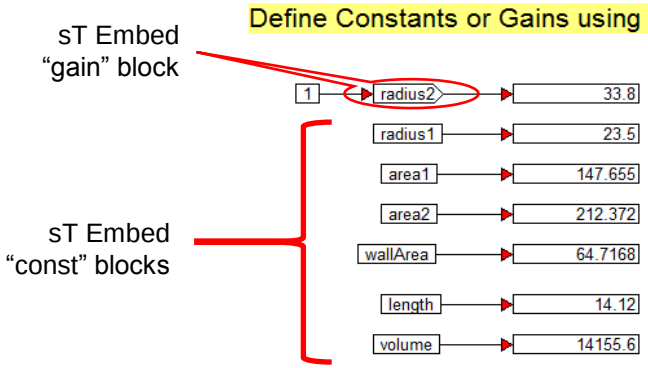


Startup script file name, create using notepad and save as “.c” file

Once the file has been created you can view and edit it using this button.

Startup Script File (2/2)

Example:




```
// Example C Script File

// Declarations
double wallArea, length, volume,
radius1, radius2, area1, area2;

// Radius Values
radius1=23.5;
radius2=33.8;

//Areas
area1= 2*pi*radius1;
area2= 2*pi*radius2;

// Wall area
wallArea = area2 - area1;

if (wallArea <0)
        wallArea = 0;

// Pipe Length
length = 14.12;

//Pipe Vowume
volume = WallArea*length;
```

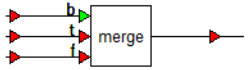
Startup Script File Example

If-Then-Else Function using Merge

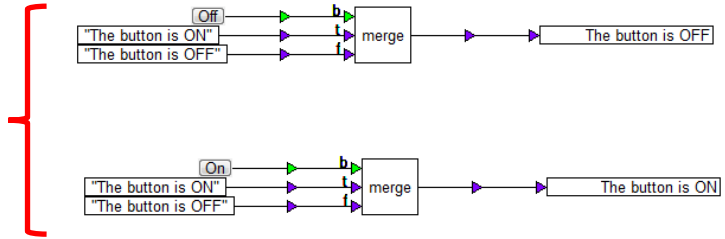
The “If-Then-Else” function may be programmed using either a “merge” block (“Blocks/Nonlinear”) or using the “enabled execution” feature of a compound block.

The “merge” block (right) requires three inputs;

- A Boolean selector signal “b” (either 1 or 0)
- A “t” input set to the desired output value if “b” = “1”
- A “f” input set to the desired output value if “b” = “0”

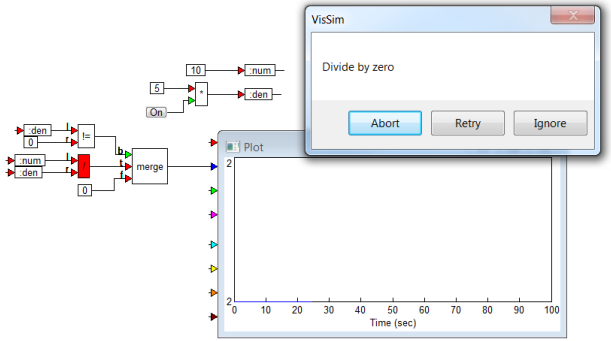


An example using the “merge” block to output a button status value is presented to the right.



When using the “merge”, both the “t” and “f” inputs are calculated before the output selection is made. Sometimes this can cause problems.

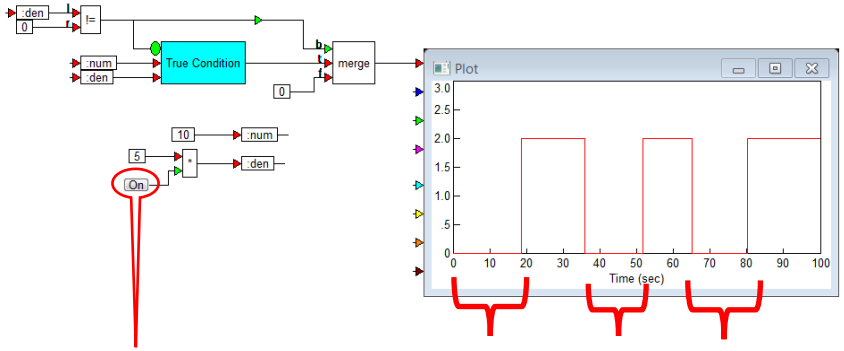
In the example shown to the right, we attempt to use the merge to prevent a “divide by 0”. Because both the “t” and “f” inputs to the merge are computed before the decision is made for output, a “Divide by zero” message is displayed.



If-Then-Else Function using Enabled Execution Compound Block

The divide by zero example is repeated using an “enabled execution” compound block. When using this approach, the “Divide by Zero” message no longer appears.

The “True Condition” compound block contains the “num/den” expression and the “enabled execution” option of the compound block is checked.



Clicking to “Off” sets the “den” = 0

Output is set = 0 when “Off” is selected

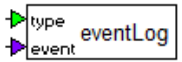
IfThenElse using Enabled Execution Compound Block Example

Event Log and Event Display (1/3)

The “eventLog” and “eventDisplay” blocks (“Blocks/Signal Consumers”) provide the ability to record and display events stored in the sT Embed system log while executing your model.

The “eventLog” block is used to record events. The “eventLog” accepts two inputs;

- type
- event

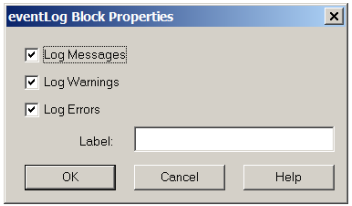


Event “types”, applied to the upper pin, are described by the enumerated list (right);

"type" value	Action
0	No event is recorded
1	Message is recorded
2	Warning is recorded
3	Error is recorded

The “event” is applied to the lower pin. The “event” can be a alphanumeric string.

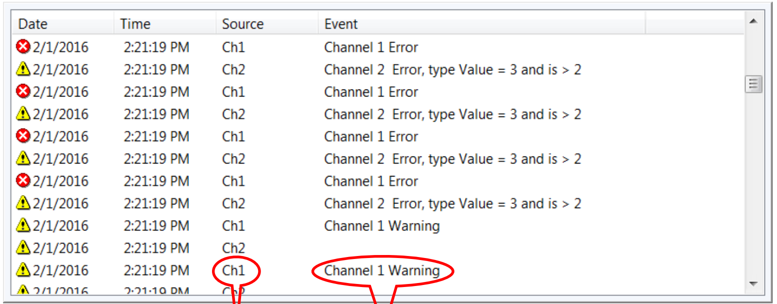
The “eventLog” block is configured with a “Label” and the “type” of events to be logged.



Once the “eventLog” block is configured, it’s “Label” and event (if it occurs), are displayed in the “eventDisplay” block.

Event Log and Event Display (2/3)

The “eventDisplay” block (right) displays the events stored in the sT Embed system log.



Date	Time	Source	Event
2/1/2016	2:21:19 PM	Ch1	Channel 1 Error
2/1/2016	2:21:19 PM	Ch2	Channel 2 Error, type Value = 3 and is > 2
2/1/2016	2:21:19 PM	Ch1	Channel 1 Error
2/1/2016	2:21:19 PM	Ch2	Channel 2 Error, type Value = 3 and is > 2
2/1/2016	2:21:19 PM	Ch1	Channel 1 Error
2/1/2016	2:21:19 PM	Ch2	Channel 2 Error, type Value = 3 and is > 2
2/1/2016	2:21:19 PM	Ch1	Channel 1 Error
2/1/2016	2:21:19 PM	Ch2	Channel 2 Error, type Value = 3 and is > 2
2/1/2016	2:21:19 PM	Ch1	Channel 1 Warning
2/1/2016	2:21:19 PM	Ch2	
2/1/2016	2:21:19 PM	Ch1	Channel 1 Warning
2/1/2016	2:21:19 PM	Ch2	

Events are Date and Time stamped. To the left of the “Date” column, the type of event is displayed as an icon:

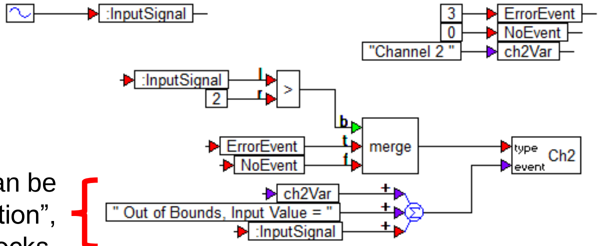
- Information event
- Warning event
- Error event

The “Source” column displays the “Label” field from the “eventLog” block

The “Event” column displays the “event” that was applied to pin 2 of the “eventLog” block.

Example:

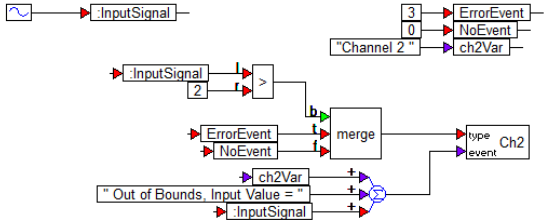
Use the “eventLog” block to record the error message “Channel 2 Out of Bounds, Input Value = xyz” whenever the “InputSignal” is > 2.
NOTE: “xyz” is the value of the “InputSignal”



Note how a string expression can be constructed using a “summing junction”, “variable” and “const” blocks.

Event Log and Event Display (3/3)

Results:



Date	Time	Source	Event
2/1/2016	2:41:45 PM	Ch2	Channel 2 Out of Bounds, Input Value = 3.38772
2/1/2016	2:41:45 PM	Ch2	Channel 2 Out of Bounds, Input Value = 3.45859
2/1/2016	2:41:45 PM	Ch2	Channel 2 Out of Bounds, Input Value = 3.4949
2/1/2016	2:41:45 PM	Ch2	Channel 2 Out of Bounds, Input Value = 3.49629
2/1/2016	2:41:45 PM	Ch2	Channel 2 Out of Bounds, Input Value = 3.46275
2/1/2016	2:41:45 PM	Ch2	Channel 2 Out of Bounds, Input Value = 3.39461
2/1/2016	2:41:45 PM	Ch2	Channel 2 Out of Bounds, Input Value = 3.29256
2/1/2016	2:41:45 PM	Ch2	Channel 2 Out of Bounds, Input Value = 3.1576
2/1/2016	2:41:45 PM	Ch2	Channel 2 Out of Bounds, Input Value = 2.9911
2/1/2016	2:41:45 PM	Ch2	Channel 2 Out of Bounds, Input Value = 2.7947
2/1/2016	2:41:45 PM	Ch2	Channel 2 Out of Bounds, Input Value = 2.57039
2/1/2016	2:41:45 PM	Ch2	Channel 2 Out of Bounds, Input Value = 2.32039
2/1/2016	2:41:45 PM	Ch2	Channel 2 Out of Bounds, Input Value = 2.04721

eventLog and eventDisplay Example

End of Section

EMBED™
solidThinking