User's Guide

SystemView®
BY ELANIX

The User's Guide to

Advanced Dynamic System Analysis Software

For
Microsoft Windows

EAGLEWARE
Electronic Design Automation Software
ELANIX

Eagleware-Elanix Corporation
3585 Engineering Drive, Suite 150
Norcross, GA 30092 USA
Phone: +1 (678) 291-0995, Fax: +1 (678) 291-0971
Support e-mail: support@eagleware.com
Web: www.eagleware.com

Document Number SVU-MG1002
Printed in the United States of America.

TABLE OF CONTENTS

SystemView®
BY ELANIX

Chapter 1. Welcome to SystemView

Welcome to SystemView by ELANIX. SystemView is a comprehensive dynamic systems analysis environment for the design and simulation of engineering or scientific systems. From analog or digital signal processing, filter design, control systems, and communication systems to general mathematical systems modeling, SystemView provides a sophisticated analysis engine.

In concert with the Microsoft Windows family of operating systems, SystemView helps you concentrate on the really important functions, designing, simulating and documenting your systems rapidly and efficiently. And taking the next step to hardware implementation is a mouse click away with DSP development options.

Complex systems may be conceived, designed, and tested within SystemView using only a mouse, your eyes, and most importantly, your mind. There are no computer codes or scripts to learn, no opaque syntax errors to battle—nothing to interfere with your concentration on the problem at hand.

SystemView consists of two primary windows: the System window and the Analysis window. Design your system in the System window by selecting tokens from various libraries including Sources, Sinks, Operators and Functions. Specify the parameters for each functional block by double-clicking the token, then entering the parameter values in the user-friendly dialog boxes.

Large systems can be easily simplified in SystemView by defining groups of tokens as a MetaSystem. A MetaSystem allows a single token to represent a complete system or subsystem. MetaSystem connections are handled in the same manner as any other SystemView token. A simple mouse click opens a window showing the complete subsystem contained in the MetaSystem.

SystemView automatically performs system connection checks, and then alerts and visually shows you each token with missing connections, even if the problem is deeply buried within a multilevel MetaSystem. This feature is essential for the efficient diagnosis of your system.

Beyond system design and simulation, SystemView also provides a flexible Analysis window to examine system waveforms. SystemView's unique Sink Calculator provides an extensive suite of block-processing operations that can be performed on the data generated by your simulation. Interactive data zoom; scroll, spectral analysis, scaling and filtering are only a mouse-click away.

SystemView's Dynamic System Probe™ is a powerful tool that speeds your design by providing real-time signal analysis in the time and/or frequency domain. Using the Dynamic System Probe, you may view the output waveform produced by any functional block in real time. As the simulation is executed, simply point and click on any block in your system to instantly view the signal waveform at the output of the selected block. The signal may be viewed in time or frequency via a toggle button, and other controls give the Dynamic System Probe many of the analysis capabilities of a standard laboratory oscilloscope.

Eagleware-Elanix Corporation is committed to maintaining SystemView as the preferred simulation tool for scientists, engineers, and mathematicians in the performance of their discipline. Customer feedback is an essential part of our ongoing development effort, so please send us your comments.

Chapter 2. Installing *SystemView*

SystemView instructions have now been moved to the GENESYS/SystemView Installation Manual.

Chapter 3. Overview of *SystemView*

3.1 The SystemView Concept of Operation

SystemView provides the means for building a visually oriented, dynamic system simulation model. SystemView uses symbolic tokens (functional models) to represent processes, and a time base to represent system sampling characteristics. Complete system or subsystem simulations are designed in SystemView's System window, where tokens are selected from various libraries and connected together in the System Design area. Parameters for these tokens are defined through user-friendly dialog boxes or by direct entry of values in graphical representations of circuit schematics. The user can control the start time, stop time, and the system sample rate.

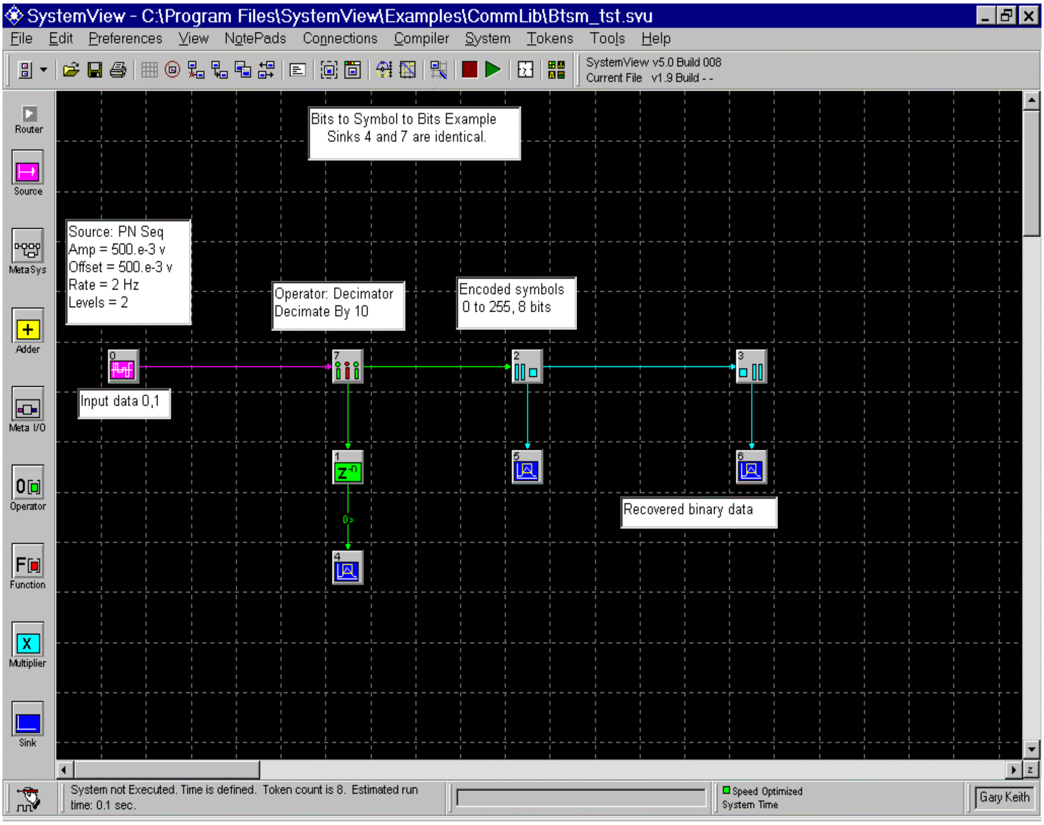
3.2 Starting SystemView

During installation, the SystemView Setup program automatically creates a SystemView shortcut on your desktop, and an icon on the Start menu. SystemView is launched by clicking the Windows Start button and selecting SystemView, or by double-clicking the SystemView Shortcut on your desktop.

Start … SystemView by ELANIX,

3.3 The System Window

The System window, shown in Figure 3.1, consists of a Menu bar, a Toolbar, horizontal and vertical Scroll bars, a Design area, a Message area, and a token Reservoir.


SystemView - C:\Program Files\SystemView\Examples\CommLib\Btsm_tst.svu

SystemView v5.0 Build 008
Current File v1.9 Build - -

Bits to Symbol to Bits Example
Sinks 4 and 7 are identical.

Source: PN Seq
Amp = 500.e-3 v
Offset = 500.e-3 v
Rate = 2 Hz
Levels = 2

Operator: Decimator
Decimate By 10

Encoded symbols
0 to 255, 8 bits

Input data 0,1

Recovered binary data

System not Executed. Time is defined. Token count is 8. Estimated run time: 0.1 sec.

Speed Optimized
System Time

Figure 3.1. The SystemView System Window

3.3.1 System Window Menu Bar

The Menu bar has pull-down menus to show SystemView's functions, ***File, Edit, View, Preferences, NotePads, Connections, Compiler, System, Tokens, Tools,*** and ***Help***.

3.3.2 System Window Toolbar

The Toolbar consists of buttons that execute functions including:

- Taking actions on tokens or groups of tokens in the System window,
- Starting and stopping a simulation
- Providing access to the Analysis window and other utility functions.

Use the toolbar to perform an action on a token or group of tokens, by Clicking the desired button, or Click on the token and click/drag the mouse to outline the desired group of tokens.

Micro help is available for each button on the toolbar. To view micro help, position the mouse pointer over the desired button, and the micro help label for that button will appear. Details about the function of the selected button will appear in the information panel of the System window.

3.3.3 Scroll Bars

The horizontal and vertical Scroll Bars are used to scroll left/right, or up/down within the System window. The arrow keys and the Page Up/Down keys may also be used to scroll. Scrolling is useful when the size of the simulation block diagram is larger than the visible region of the Design area

3.3.4 Design Area

In the System window, the Design Area is defined as the area where the simulation design is performed. In the Design Area, you launch, define and connect tokens to create the block diagram of the design.

3.3.5 Message Area

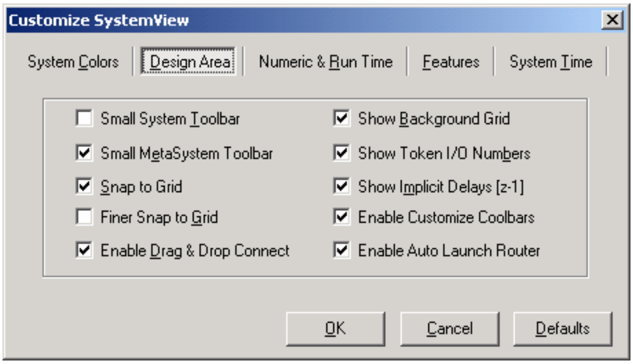
The Message Area at the bottom of the window, displays simulation status information and token descriptions. It is possible to display token parameters in the Message Area, as well as on a pop-up window, by positioning the mouse arrow on the desired token.

3.3.6 Dynamic System Probe Button

The Dynamic System Probe Button is located at the lower left corner of the System Window. This allows access to the Probe Functions, as discussed in Chapter 8.

3.3.7 Auto Router

The Router allows the designer to drop a router while dragging the connection. These routers were placed using the Auto Launch Router feature. The Auto Launch Router feature is controlled through the Preferences Menu | Options | Design Area. This control window is shown in figure 3.2. Note the Enable Auto Launch Router option on the bottom of the left column.



□ Small System Toolbar
☑ Small MetaSystem Toolbar
☑ Snap to Grid
□ Finer Snap to Grid
☑ Enable Drag & Drop Connect
☑ Show Background Grid
☑ Show Token I/O Numbers
☑ Show Implicit Delays [z-1]
☑ Enable Customize Coolbars
☑ Enable Auto Launch Router

Figure 3.2 Auto Launch

The Auto Router features are listed below.

- Drawing a connection in a feedback type configuration results in the placement of a router with the token direction reversed.
- The Preference menu allows the designer to toggle the Auto Launch Router feature.
- A buffer around the “connect to” token eliminates undesired router placement.
- Selecting the right mouse button over the router allows the user to “Snap to Grid” eliminating the jagged connections.

3.3.8 Token Numbers

The user has control of what numbers are associated with tokens. The initial numbering of tokens is obtained through the order of new token placement in the design space. Users can re-number tokens using the Edit | Assign Token Numbers | By Execution Sequence, By Connection, or Using the Mouse, as shown in Figure 3.3.

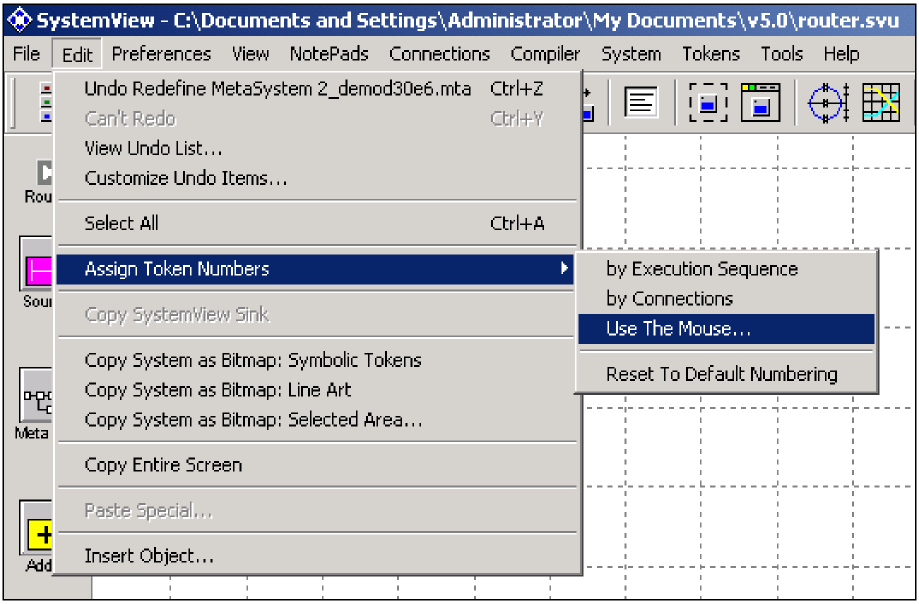
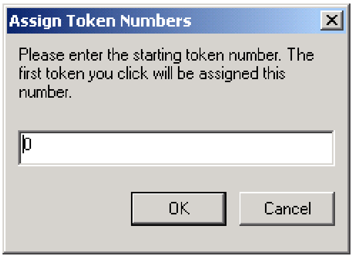


Figure 3.3 Documents and Settings

The option to assign token numbers using the execution sequence will set the token numbers using the compiler method. The compiler has a higher priority for source tokens and an end priority on sink tokens. The option to assign token numbers by connections follows through token connectivity. Therefore, a particular flow path will have consecutive token numbers.

The Mouse option allows the user to graphically select the token numbering sequence. The user is first prompted for the starting token number, note figure 3.4 below.



Assign Token Numbers

Please enter the starting token number. The first token you click will be assigned this number.

0

Figure 3.4 Token Number Selection

The user is not required to go through all tokens in the simulation. Once the user has completed setting the token numbers for the tokens of interest the "shift F" keys will finish the token numbering. This method will swap duplicate token numbers if duplicate numbering exists.

3.3.9 The Token Reservoir

Tokens are the building blocks used in all SystemView simulations. The reservoir, on the left side of the System window, shows generic tokens representing those available.

Library Button
Click this button to toggle between the standard SystemView libraries and the optional SystemView libraries.

Source Token
This token has outputs only, and represents a Source library, (including data from external files) used to create system inputs.

MetaSystem Token
This token allows the adding of hierarchy to a design. The MetaSystem token is used to import a predefined MetaSystem into the current system under design

. Refer to Chapter 7 for details.

Adder Token
This token provides outputs, equal to the sum of its inputs (up to 20 inputs).

MetaSystem I/O Token
This token defines the input and output nodes within a MetaSystem.

Operator Token
This token represents a library of Operators that use the input data as the argument for the operation.

Function Token
This token represents a library of Functions that use the input data as the argument to the function.

Multiplier Token
This token provides outputs, equal to the product of its inputs (up to 20 inputs).

Sink Token
This represents a library of Sinks for collecting, displaying, analyzing, and outputting the data within a system. Sink tokens have inputs only.

3.3.10 Defining Tokens

Tokens may be added, from any SystemView library, to a simulation by dragging a generic token from the Reservoir into the System window. When the new token is double clicked, (or click the right mouse button and select Library) the appropriate token library will appear. Figure 3.5 shows the token library window for the Function Library. Within each library window, the tokens are organized into groups. The Non Linear token group of the Function Library is shown.

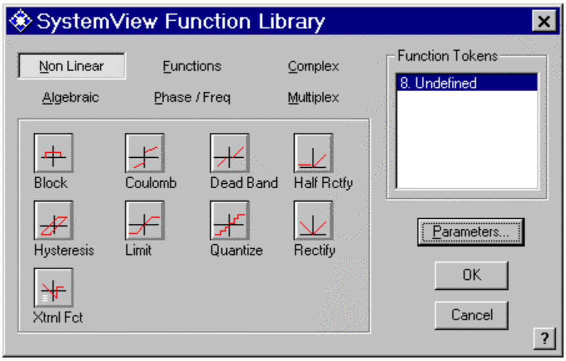
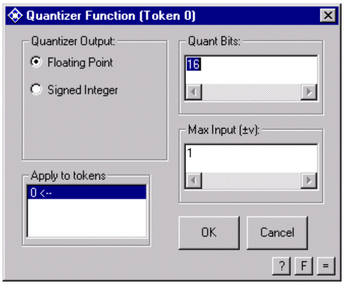


Figure 3.5. The Function Library

From the library window, select the desired token and define its parameters by clicking the Parameters button. A typical parameter window is shown in Figure 3.6, and allows the entry of the specific parameters desired for the token. Note that identical parameters can be assigned to the same type of tokens, by selecting the target tokens in the Apply To window.



Quantizer Output:
● Floating Point
○ Signed Integer

Quant Bits: 16

Max Input (±v): 1

Apply to tokens: 0 <--

Figure 3.6. Typical parameter entry window

Algebraic Parameter Entry

It is possible to perform algebraic operations while entering parameter values. For example, entering the line: *(3*Sin (pi/8))^2* gives *1.318019* as the parameter value.

The hierarchy of operations is as follows:

Functions (e.g., sin (), sinh ())

^ (Exponentiation)

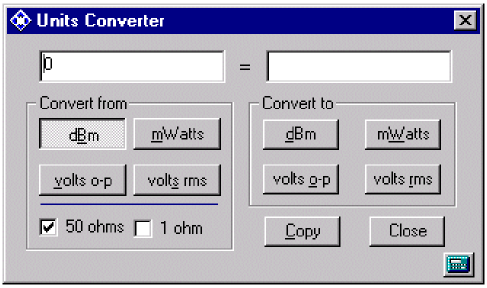
*, /, +, - (Multiplication, division, unary addition, unary subtraction)

+, - (Addition, subtraction)

The built in functions are:

abs
acos
asin
atan
atan2
cos
cosh
exp
frac
int
log
log2
log10
mod
pi
sin
sinh
sqrt
tan
tanh

The equals (=) button, (lower right), opens the Units Converter window of Figure 3.7. It allows quick conversion of the value required for the application, into the units presented in the Parameter window. The window is also available by selecting *Units Converter* from the *View* menu.



0

☑ 50 ohms □ 1 ohm

Figure 3.7 SystemView Units Converter

The standard or scientific Windows Calculator is accessed by clicking on the equals (=) button at lower right corner of the Units Converter, or by selecting Calculator from the View menu in the System window.

(TIP: *Once a token is defined, you may bypass the token library screen, and go to the Parameter window. Click the right mouse button and select Edit Parameters.)*

3.3.11 Connecting and Disconnecting Tokens

SystemView has two ways to connect tokens. The user can place the mouse on the "from" token until an "UP" arrow appears, then click the left mouse button and drag the wire to the "to" token. Or to connect tokens, click the Connect button on the toolbar and then click the "from" token, followed by the "to" token.

To disconnect tokens, place the mouse on the token to be disconnected and wait until an "UP" arrow with a break in it appears, then click the mouse and drag to the token to be disconnected Or click the Disconnect button on the toolbar and click the "from" token, followed by the "to" token.

3.3.12 Saving, Recalling, and Exiting

Save a system by selecting *Save* from the *File* menu. Any legal file name can be entered. The .svu file extension is automatically added if no extension is entered. It is good practice to save the work as the system is being designed.

To open an existing file, select *Open* from the *File* menu. The SystemView folder contains numerous example files. To exit SystemView, select *Exit* from the *File* menu.

3.4 Example System

To create the example system shown in Figure 3.8 follow the steps listed. This simple system generates a sinusoid and then produces its square. Refer to Chapter 4 regarding details of system timing.

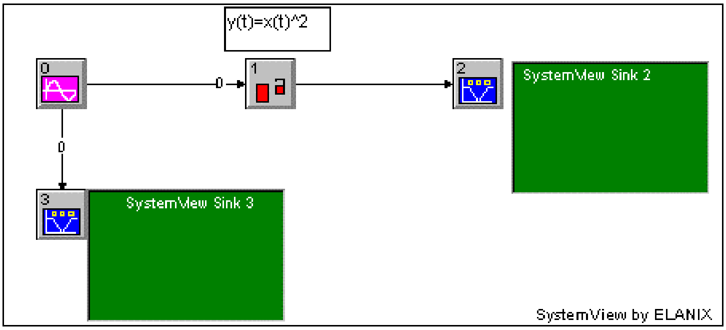


Figure 3.8 Sample system used to create the square of a sinusoid (prior to execution)

1. Click the System Time button (the stopwatch on the toolbar).

Click OK to accept the default values.

Launch a Source token. Double click the token (or click the right mouse button and select Library) to display the Source Library. Click on the Periodic group button, and click "Sinusoid" to select the sinusoid source. Click the Parameters button, and in the Frequency (Hz) text box, enter the value 4. Click OK. Place the mouse arrow on the token and note that the Source defined has a unit amplitude sine wave with a 4 Hz frequency.

2. Launch a Function token. As with the Source token, double click the token to display the Function Library. Click on the Algebraic group button and select X^a. Click the Parameters button. In the Exponent window, delete the flashing default value and enter 2, to produce the square of the input sine wave.

3. Launch a Sink token. Double click and select the Graphic group. Select SystemView as the Sink, and note that it has no parameters. Click OK. Changing to a Real-Time Sink, allows waveform monitoring as the system runs.

4. Connect the Source token to the X^a function token. The default is Sine.

5. Connect the X^a function to the Sink token.

6. Launch another Sink token and define it as a SystemView Sink.

7. Connect the Sinusoid source (Sine Output) to this second Sink token.

8. Click the Execute button. The results will as shown in Figure 3.9.

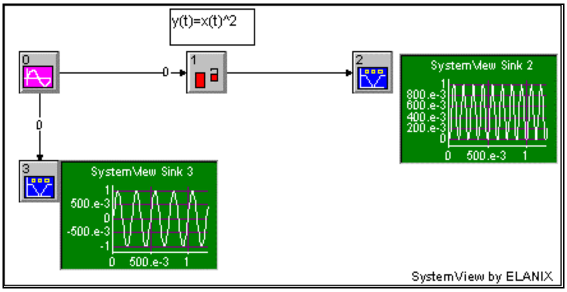


Figure 3.9 Example of system after execution

1. Expand either SystemView plot by selecting the plot window with the mouse, then click/drag one of the size controls on the plot boundary.
2. Click the Analysis button if the plots do not automatically appear, click the New Sink Data toolbar button at the far left on the Analysis toolbar. Next click the Open All button on the toolbar. Two plots should be seen, the input 4 Hz sine wave, and the square of the sine wave.
3. Click the Sink Calculator button at the base of the window, and select the Operators tab. Select Overlay Plots and then select the two sinks in the list that are to be overlaid, then click OK.
4. Click the Sink Calculator again and click the Spectrum tab. Select any one of the spectrum types. Select the newly created overlay window from the list and click OK. A new plot window will appear showing the spectrum associated with the two input signals. Verify that one line occurs at 4 Hz, and that the others (the squared signal) are located at 0 Hz and 8 Hz. Click the Tile Vertical toolbar button to arrange the plot windows.

3.5 Example Feedback System

The example feedback system shown in figure 3.10 and described below will help with the creation of a dynamic system containing feedback. The system results can be seen in figure 3.11.

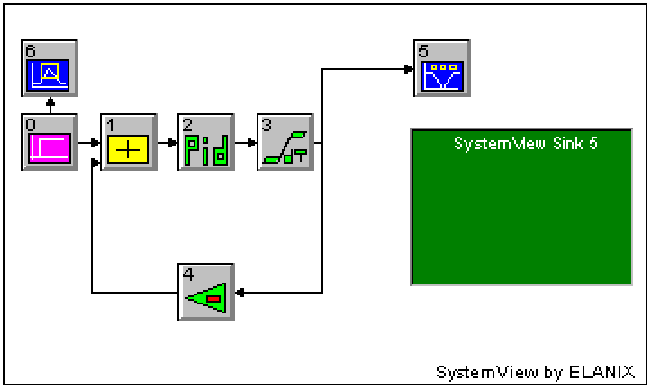


Figure 3.10 Feedback system example (prior to execution)

1. Clear the system using the clear button on the toolbar.
2. Click the System Time stopwatch on the toolbar. Clear the default value and set the number of samples to 512, then click OK.
3. Launch a Source token. Double click the token to view the Source library. Click on the group button titled Aperiodic, and select the Step Fct token. Use the default values and click OK.
4. Launch an Adder token.
5. Launch an Operator token, and then select the Integral/Differential group. Select the PID (Proportional, Integral, Derivative) token. Set the proportional gain and integrator gain to one set the derivative gain to zero.
6. Launch an Operator token and select the Integral /Differential group. Select the Integral token, and use the default values.
7. Launch another Operator token. Select the Gain token from the Gain/Scale group set the Gain Units to linear and the Gain parameter to minus two.

8. Click the right mouse button on the Gain token and select Reverse (or click the reverse token button on the tool bar and apply it to the Gain token.
9. Launch a Sink token, then double-click and select the Graphic group, and select SystemView as the sink type.
10. Launch another Sink token. Double click it and select the Analysis group, and select Analysis as the sink type.
11. Connect the step function Source token (0) to the Adder token (1).
12. Connect the step function Source (0) to the Analysis sink (6).
13. Connect the Adder (1) to the PID token (2).
14. Connect the PID token (2) to the Integral token (3).
15. Connect the Integral token (3) to the System View Sink token (5).
16. Connect the Integral token (3) to the Gain token (4).
17. Connect the Gain token (4) to the Adder token (1) to close the feedback loop.
18. Execute the system by clicking the Run button ▶.

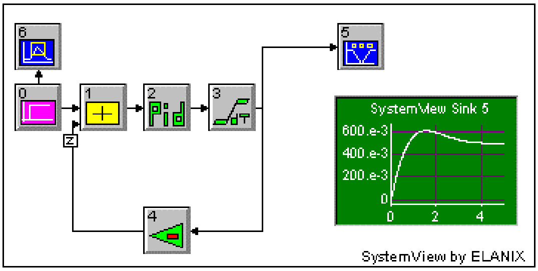


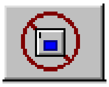
Figure 3.11 Feedback system example after execution

After execution, a [z-1] will appear at the input to the Adder token (1). This represents the location of the implicit feedback delay for the loop that is automatically shown by SystemView. Chapter 13, *Time Delays in Feedback Paths*, describes this implicit delay in detail.

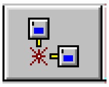
The Data List is another useful way to view the data in SystemView. Return to the System window and redefine each Sink to be a Data List. Data lists will replace the graphical representations. These lists show the numerical data held by each Sink.

3.6 System Tool Bar

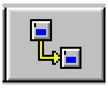
This section summarizes some key Tool Bar operations.



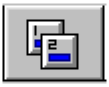
Delete — This button is used to delete a token or a group of tokens.



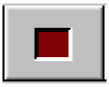
Disconnect — This button is used to disconnect two or more tokens. To disconnect two tokens, click the Disconnect button, then click the "from" token followed by the "to" token. To disconnect a group of tokens, click the Disconnect button and, while pressing the Ctrl key, drag the pointer to outline the group of tokens to be disconnected.



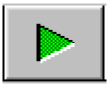
Connect — This button connects two tokens. Click the Connect button, and then click the originating token, and then the destination token.



Duplicate Tokens — This button is used to duplicate one or a group of tokens. To duplicate a token, click the Duplicate button and then the token to be duplicated. To duplicate a group of tokens, click the Duplicate button and, drag the mouse pointer to outline the group of tokens to be duplicated. Note that all paths in the duplicated group are preserved.



Cancel — This button cancels a current pending action (The Esc button may also be used).

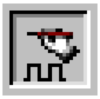


Run System — This button executes the system simulation.

3.7 Speed Optimization

The Speed Optimization button, located at the bottom of the System Window, controls the interface between SystemView and Windows. It allows SystemView to retain a greater share of resources during a simulation run. When used, the sampling rate for user interaction (mouse and keyboard) is reduced, thus increasing effective allocation of the processing resources to the simulation. The user-interface sampling rate becomes a function of the complexity of the model under simulation. For a small model, little delay will be noticeable, however, for a very large model, the delay for user interaction may take seconds. This feature can be enabled or disabled from the Preferences Menu.

3.8 The Dynamic System Probe



The Dynamic System Probe is both an oscilloscope and a spectrum analyzer, integrated into a single tool that is available whenever the system is running. A selection may include internal or external trigger sources (tokens); free-run, positive, negative, or positive and negative trigger slopes, and the trigger threshold level. Refer to Chapter 8 for a complete description of the use of the Probe. Some of its key features include:

Probe Enable/Disable — Click /drag the Probe button (located at the bottom of the screen) to any token within the system.

Trace a signal through the system — Move the probe to each token in succession within the system.

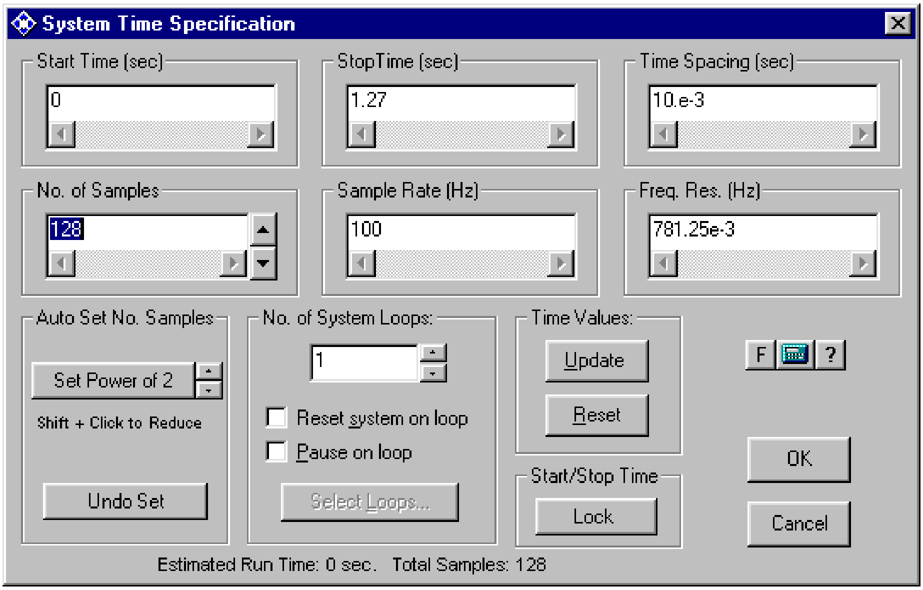
Start/Stop/Pause/End/Restart the run — Control system execution directly from the probe.

Chapter 4. System Time

SystemView is a true dynamic system simulator, meaning the system under design/simulation, will act in SystemView just as it would in the “real world.” Filters have transients, feedback loops exhibit the proper acquisition characteristics, non-linear elements produce the proper time and spectral effects, and so on.

Clicking the System Time button on the toolbar controls timing.

This is how the master clock for a system is defined. Figure 4.1 shows the System Time Specification window as it appears initially. The resulting System Sample Rate will be the maximum rate allowed in the model. If the design is a mixed-mode (analog and digital) or mixed-rate system, use the Sampler, Decimate, Hold, Resampler, or Sampler Delay tokens found in the Operator Library. These are used to change sample rates within the system, or change sample rates using the Extract or Multiplex tokens in the Function Library.



System Time Specification

Start Time (sec): 0

StopTime (sec): 1.27

Time Spacing (sec): 10.e-3

No. of Samples: 128

Sample Rate (Hz): 100

Freq. Res. (Hz): 781.25e-3

No. of System Loops: 1

□ Reset system on loop

□ Pause on loop

Estimated Run Time: 0 sec.   Total Samples: 128

Figure 4.1 System Time Specifications.

4.1 Start Time/Stop Time

These parameters control the time domain range. There are no constraints on the Start or Stop time range, but <u>the Stop Time must be greater than, or equal to, the Start Time</u>.

4.2 Sample Rate/Time Spacing

The two parameters, Sample rate and Stop Time, control the time step used in the system simulation. Since SystemView is running on a time-discrete machine (a PC), the system is inherently a time-discrete system. Time Spacing *or* Sample Rate, can be specified, but the two parameters are *not* independent, and are constrained by the relation Sample *Rate* = 1/ *Time Spacing*. Changing one parameter, automatically forces a proportional change in the other. When simulating analog (continuous) systems, a sample rate selection rule is to specify a value <u>*at least* four times greater than the highest frequency in the system</u>.

4.3 No. (Number) of Samples and Start/Stop Time Lock

This entry specifies the number of time samples that will be executed by the system. The basic relation is *No. Samples = (Stop Time - Start Time) x Sample Rate+1***.** This equation includes three variables, and the following rules apply:

1. When the Start and Stop time are **not locked,** the Start Time will not change but the Stop Time will change in accordance with the change in number of samples.
2. If the Start/Stop Time is **locked**, changes in the number of samples, or sampling rate, will occur automatically.
3. If either, or both, the Start Time and Stop Time are changed, the number of samples will change.
4. Only an integer number of Samples is allowed. If the basic relation does not produce an integer, the No. Samples window will be appropriately rounded to the nearest integer. The system time will begin at the Start Time and execute the simulation for the designated number of time steps. Note that the Sample Rate remains fixed, until it is changed.

4.4 Frequency Resolution

This entry is the available Frequency Resolution, based on the system simulation time span. The value is computed from the formula, *Frequency Resolution = Sample Rate/No. Samples.*

4.5 Update Time Values

When a change is made to a particular time specification, all related time specifications are automatically updated, when the Update or the OK button is clicked. Clicking the Reset button will restore the previous saved values.

4.6 Auto Set No. Samples

This feature facilitates Fourier-transform operations. The SystemView FFT routine uses a radix 2 basis for optimal speed. It will pad zeros if the number of samples is not a power of two. The Auto scale feature allows the setting of the No. Samples field to a power of two. When clicking on the "Power of 2", the No. Samples field is rounded to the nearest power of two. When holding down the shift key and clicking on the "Power of 2" button, the No. Samples field is rounded down to the nearest power of two. The Stop Time is adjusted automatically. The Undo Set returns to the original setting.

4.7 Number of System Loops

The number of loops, Reset / Pause System on Loops. Select Loops, is a powerful feature of SystemView. It allows repeated runs of the system automatically, with different parameters for each loop.

It is important to consider the effects of using the "Reset System On Loop" option. This option controls what happens to the system at the end of each loop. If the option is off, then all system parameters will be remembered from loop-to-loop. If, for example, there is an integrator or filter in the system, the output at the end of one loop simply becomes the initial condition for the next loop.

If the Reset option is activated, then all token values are reset after each loop. Statistical averaging using the Averaging Sink over loops is possible, to make repeated runs, thus averaging the results over many trials. If the basic system is a sine wave with added noise, and is followed by an FFT, the SNR in the signal FFT bin can be improved by setting the number of System Loops to a value (*L*) greater than one. Each run will use a different noise segment. The Averaging Sink then averages the individual time segments together. The signal portion adds coherently, while the noise deviation falls as the reciprocal square root of *L*.

The Pause On Loop feature allows a simulation pause at the end of each loop to analyze the current results. For example, the Analysis window shows the current sink waveforms. Select Loop, can only be enabled when the Pause on Loop option is selected.

4.8 Modifying the System Sample Rate

The sample rate can be modified, resulting in a mixed-rate model. Using the Decimate, Sampler, Hold, Resampler, Delay, the Sampler Delay operator tokens, or the Extract, Multiplex or demultiplex function tokens can do this. The system rate produced by the master clock is not affected.

The Decimate token is used to control the number of samples taken in a simulation. The tokens following the Decimate token will be sampled at the System Sample Rate divided by *N* (*N* is the decimation factor parameter and must be a positive integer). An example of a decimated-unit sine wave with a decimation factor of four is shown in Figure 4.2.

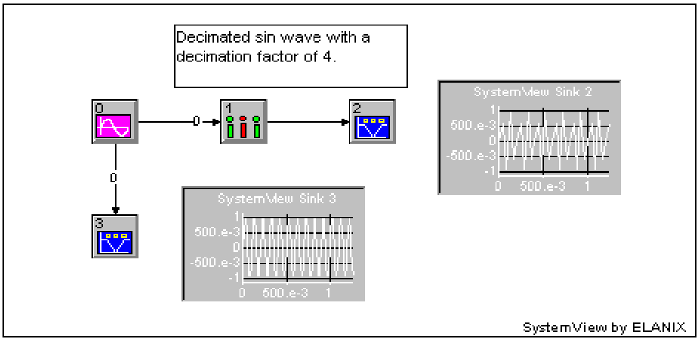

Decimated sin wave with a decimation factor of 4.

Figure 4.2 Decimation example

The sampler token controls the sample rate of the tokens following it. The designated sample rate must be less than or equal to the system sample rate, but is not restricted to multiples of the system sample rate.

The Hold token can be used to get back to the system sample rate following a Decimate or Sampler token. The resulting signal output will be held at the last sample value, or can be set to zero between samples. The Resampler token has the same effect as choosing a Hold token followed by a Sampler token.

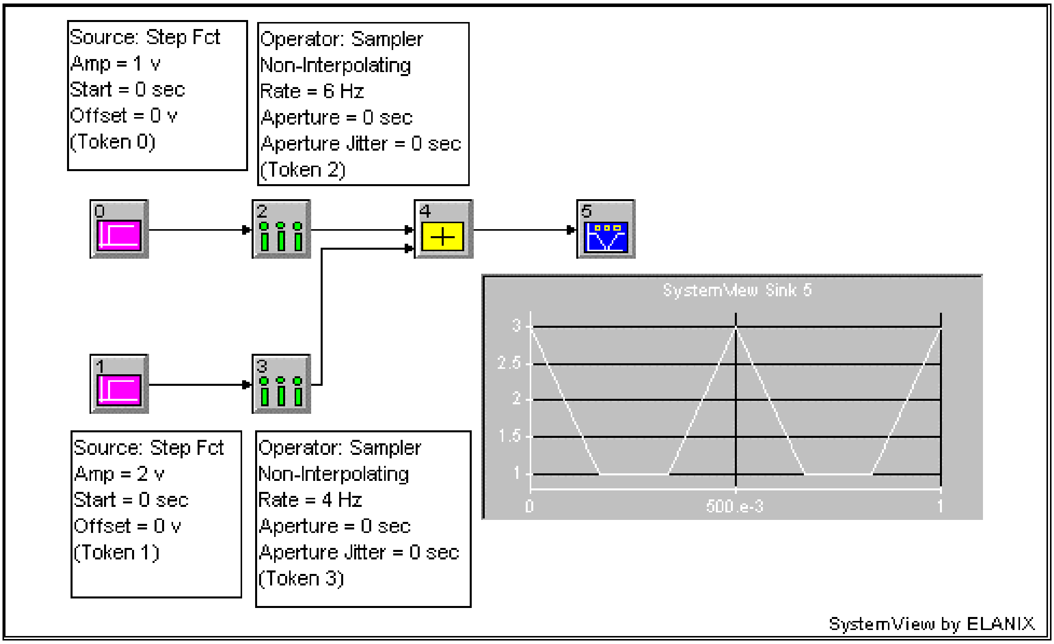
The Delay token is used when a specific amount of delay must be added to an input signal. The amplitude of the output signal will be linearly interpolated when this option is selected. In contrast, the Sample Delay token delays the input signal by a specified integer number of samples. In the Function Library, the Extract token only takes a sample when the control signal exceeds the specified threshold. NOTE: the output is compressed in time.

The Multiplex token takes N_A samples from the input token A, followed by N_B samples from input token B. An error results if the token rate is greater than the system sample rate. The output token rate is increased by a factor of:

$$R_{out} = \frac{N_A + N_B}{T} = R_A + R_B$$

4.9 Tokens and Mixed Rate Inputs

Multiple-input tokens that receive data signals at different rates, sample the data at the same rate as the data signal with the highest rate. See Figure 4.3. Source step function tokens 0 and 1, have amplitudes of 1.0 volts and 2.0 volts respectively. These are followed by Sampler tokens, 2 and 3, running at 6 Hz and 4 Hz respectively. The parameters can be shown in note pads on the SystemView screen by clicking the right mouse button and choosing this option. Token 4, adds the sampled signals and outputs them to a SystemView Sink. In this case the System rate is set at 10 Hz and the simulation duration is 1 second. However, since the highest sample token rate is 6 Hz, this value controls the Adder token. An output to 3 volts will occur only when the inputs to the adder are perfectly aligned in time. This occurs at the least common multiple of 4 and 6 Hz that in this case is every 12 Hz.



Source: Step Fct
Amp = 1 v
Start = 0 sec
Offset = 0 v
(Token 0)

Operator: Sampler
Non-Interpolating
Rate = 6 Hz
Aperture = 0 sec
Aperture Jitter = 0 sec
(Token 2)

Source: Step Fct
Amp = 2 v
Start = 0 sec
Offset = 0 v
(Token 1)

Operator: Sampler
Non-Interpolating
Rate = 4 Hz
Aperture = 0 sec
Aperture Jitter = 0 sec
(Token 3)

Figure 4.3 Mixed-rate inputs

Chapter 5. The *SystemView* Tokens

Tokens are the building blocks (models) of a SystemView simulation. This chapter summarizes the tokens included in a standard SystemView installation and explains the use. The standard SystemView libraries include Sources, Sinks, Functions, Operators, a MetaSystem I/O definition, an Adder, and a Multiplier token.

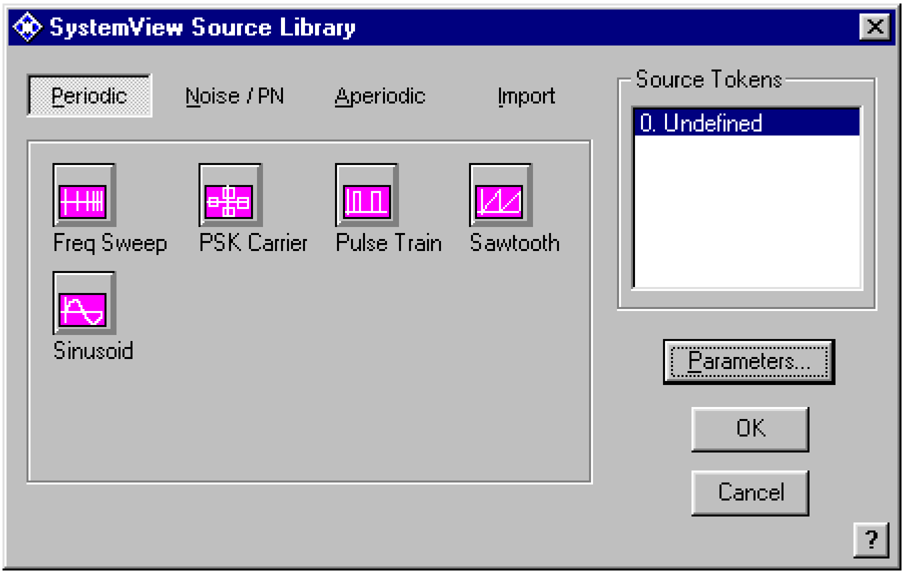
Chapter 14 provides a detailed description of each standard SystemView token. In addition to the standard token libraries, Eagleware-Elanix offers a number of optional libraries, which are described in section 14.7. Optional libraries, when ordered, are supplemented with separate printed technical guides.

5.1 Sources

Every system must have at least one Source. SystemView provides a diverse library of standard sources to provide inputs to the system. The standard sources are supplemented by the Custom Sources and imported Sources. The Custom Sources allow the definition of a variety of source types. The designer can also define an external file as a Source so that SystemView can import signals and data.

5.1.1 Periodic Sources

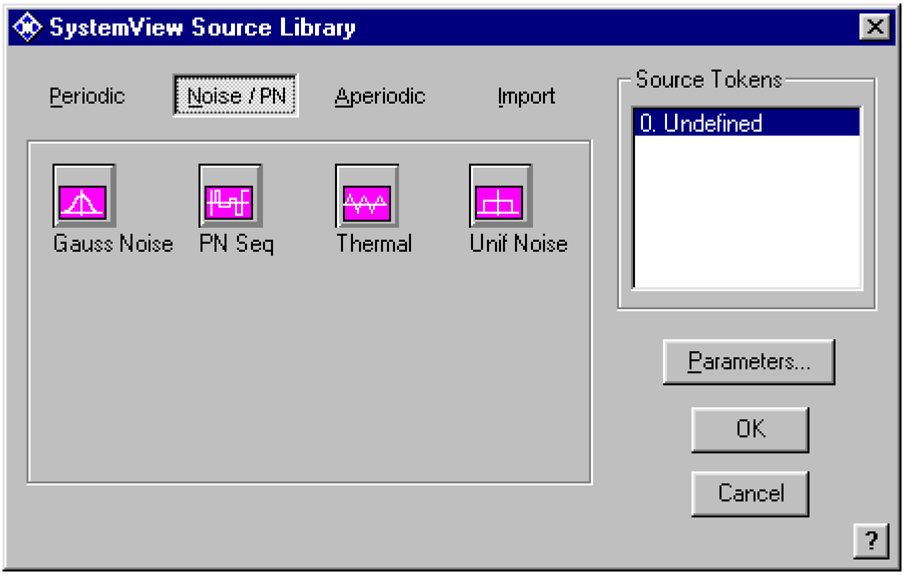
The Periodic sources generate cyclic data, and each one has a set of parameters that define the token. For example, the Sinusoidal Source parameters include frequency, amplitude, and phase of the sine wave. The Periodic source library is shown below.



Periodic Source Library

5.1.2 Noise / PN Sources

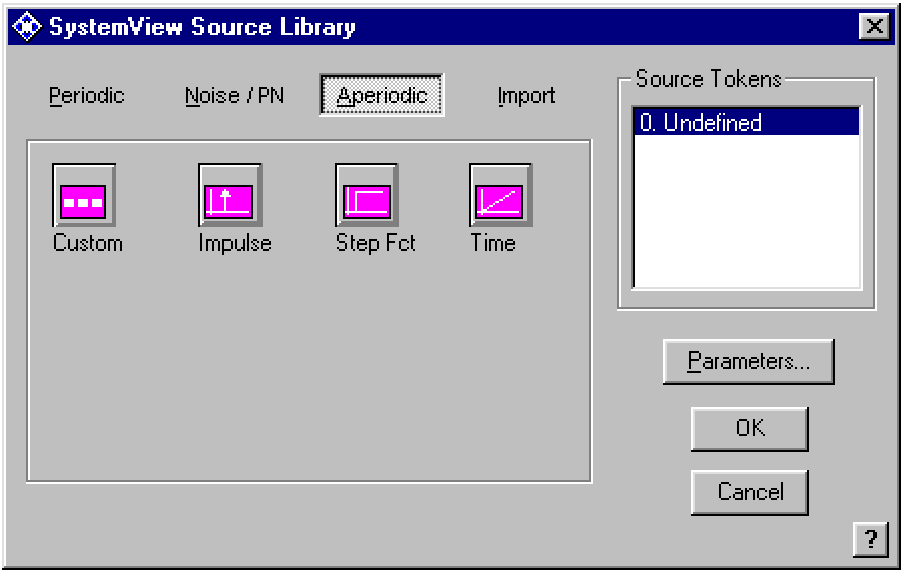
The Noise / PN Sources utilize a random number generator to provide samples to support statistical analysis of a system. The Noise source library is shown below.



Noise Source Library

5.1.3 Aperiodic Sources

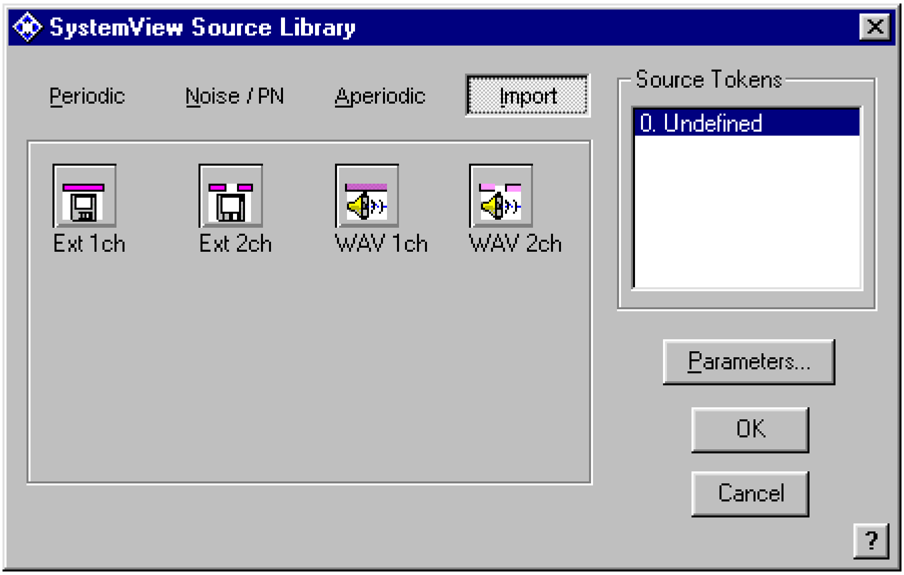
The Aperiodic Sources provide aperiodic data for a simulation, including a custom source token. The Aperiodic source library is shown below.



Aperiodic Source Library

5.1.4 Import Sources

The Import Source is used to import specifically formatted data into the design, from an external file designated by the user. The Import source library is shown below.



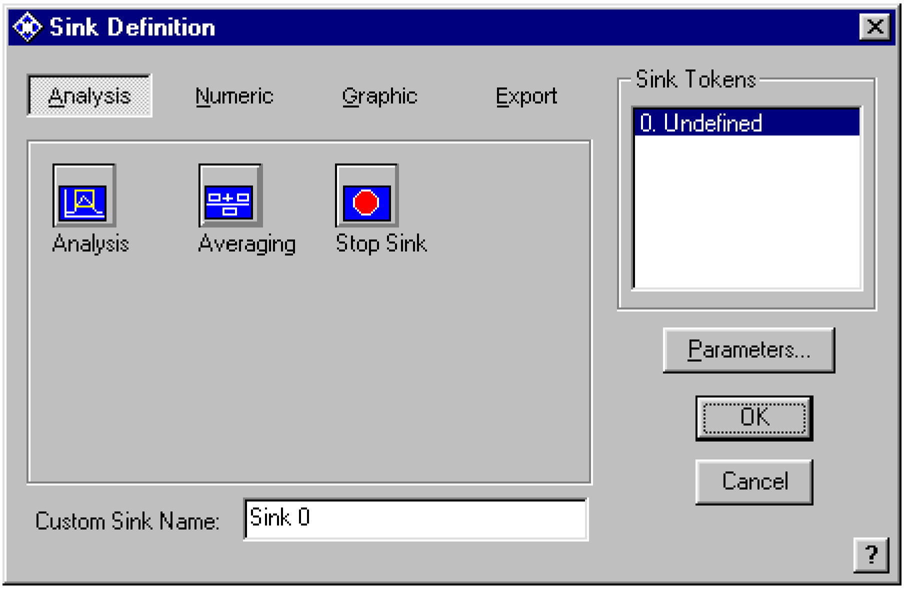
Import Source Library

5.2 Sinks

Every system must have at least one Sink that has been defined and has been connected. A SystemView Sink stores the output of the token connected to it, and the data contained in a Sink is available for detailed examination in the Analysis window. Some sinks present collected data directly in the SystemView Design Window. The following are brief descriptions of the sink libraries.

5.2.1 Analysis Sinks

The Analysis Sinks collect data for later analysis. The Stop Sink allows the termination of a simulation run, based on the results.



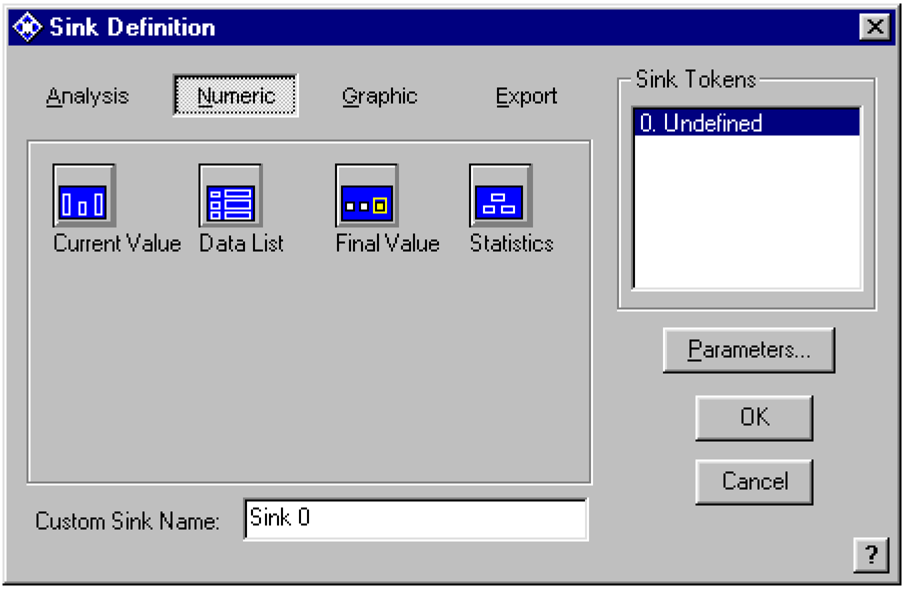
Sink Tokens: 0. Undefined

Custom Sink Name: Sink 0

Analysis Sink Library

5.2.2 Numeric Sinks

The Numeric sinks provide and display data on the screen. The Data List provides the output in tabular form on the system screen. This feature is extremely useful for debugging a system when a point-by-point numerical output is required.



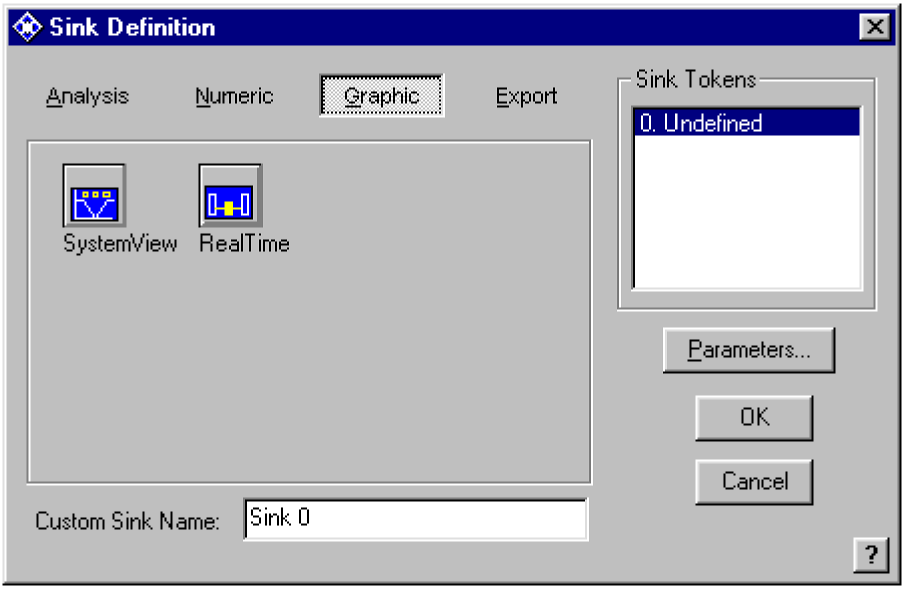
Sink Tokens: 0. Undefined

Custom Sink Name: Sink 0

Numeric Sink Library

5.2.3 Graphic Sinks

The Graphic Sink is used to display the particular graphs in the SystemView design area. The Real-Time Sink displays data in real-time, as the system executes. Outputs may be seen in the analysis window, or the system probe can be used to view waveform development as the system executes.

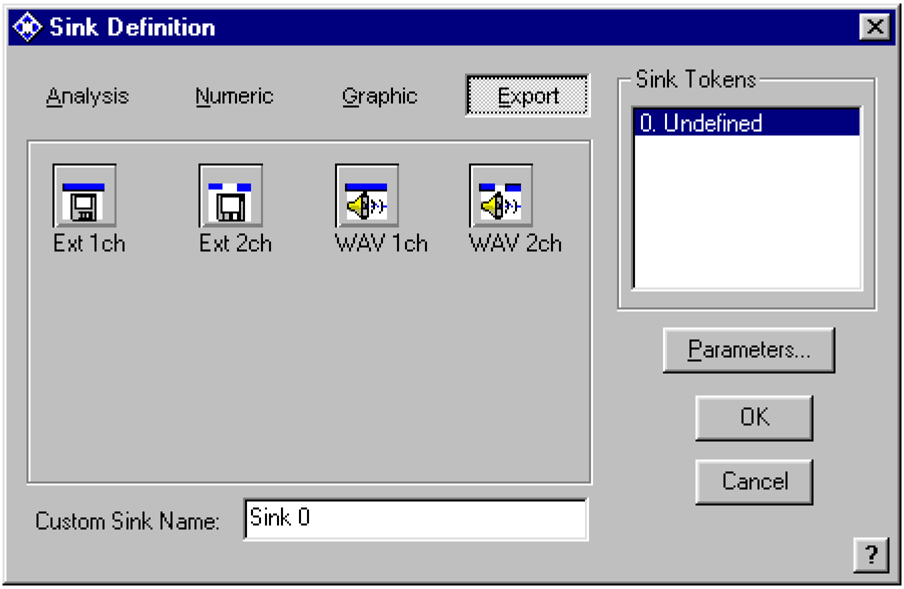


Custom Sink Name: Sink 0

Graphic Sink Library

5.2.4 Export Sinks

The Export sink allows the export of data in a variety of formats. The 1 channel (monaural) and 2 channel (stereo) Wav sink output, is sent to a special windows WAV-formatted disk file.


Sink Definition

Sink Tokens: 0. Undefined

Custom Sink Name: Sink 0

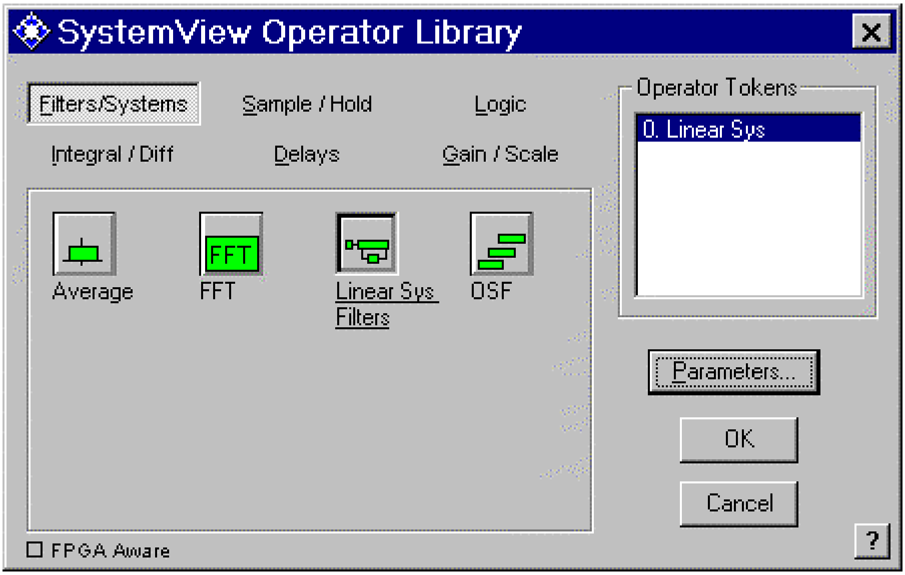
Export Sink Library

5.3 Operators

The Operator library contains tokens that perform an operation on their inputs. The general form is $y(t) = Opr\{x(t)\}$, where $x(t)$ is the input time function, $y(t)$ is the output time function and *Opr{}* is the specified operation (for example, the derivative). Chapter 14 provides a detailed description of the Operators, their mathematical functions, and user-defined parameters.

5.3.1 Filters / Linear System Operators

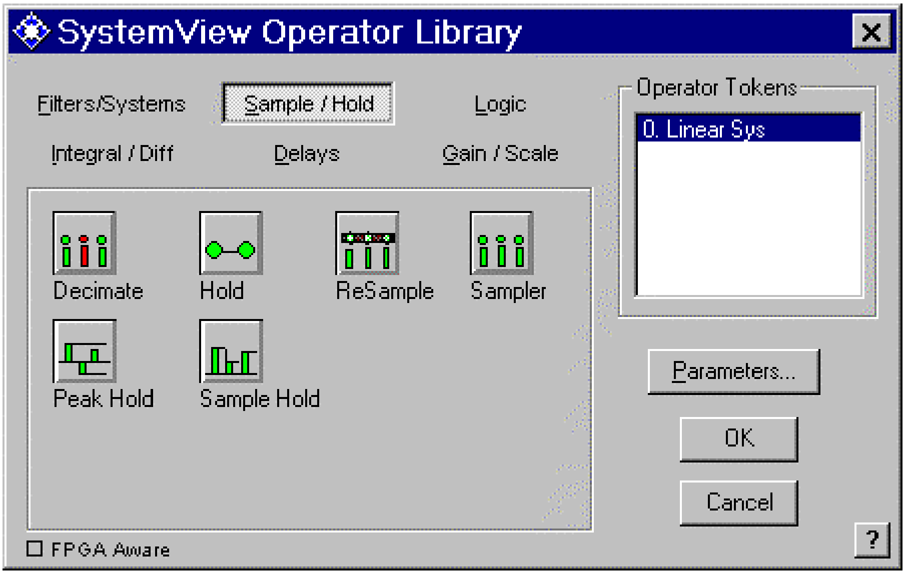
The Filters / Systems Operators perform a variety of linear system operations, including analog and digital filter design. Chapter 6 describes the Linear Sys Filters token in detail.



Filters/Systems Operator Library

5.3.2 Sample/Hold

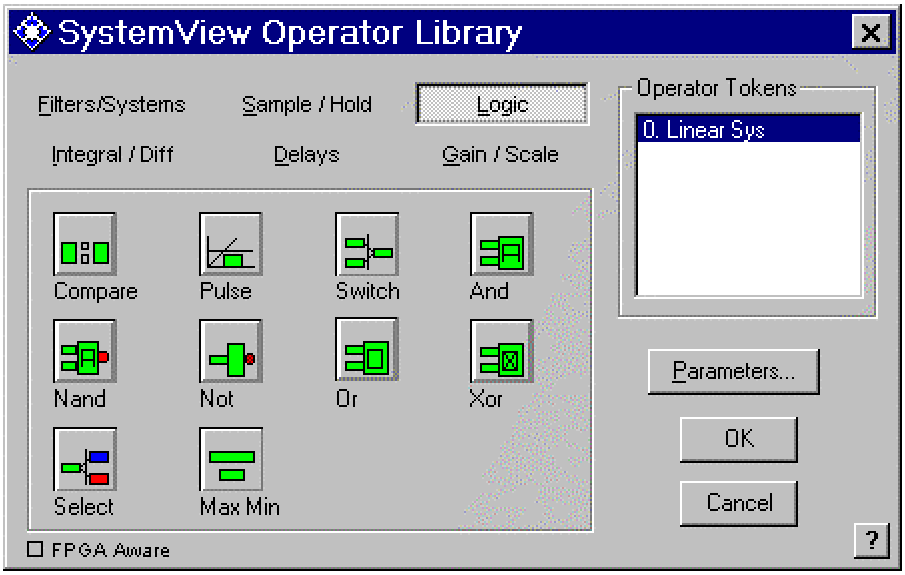
Both the Sample and the Hold Operators modify the sampling rate within the system. This library also allows sampling/tracking, and hold functions.



Sample/Hold Operator Library

5.3.3 Logic Operators

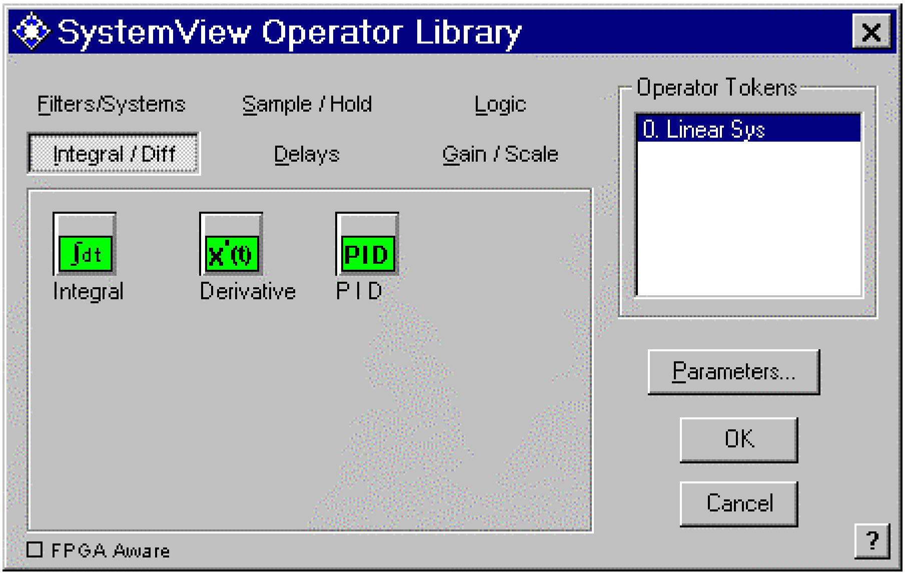
The Logic Operators perform logical operations based on the input sample values. These operators include Boolean, comparison, and switching operations. (Bit-wise logic tokens are included in the optional DSP Library).



Logic Operator Library

5.3.4 Integral / Differential

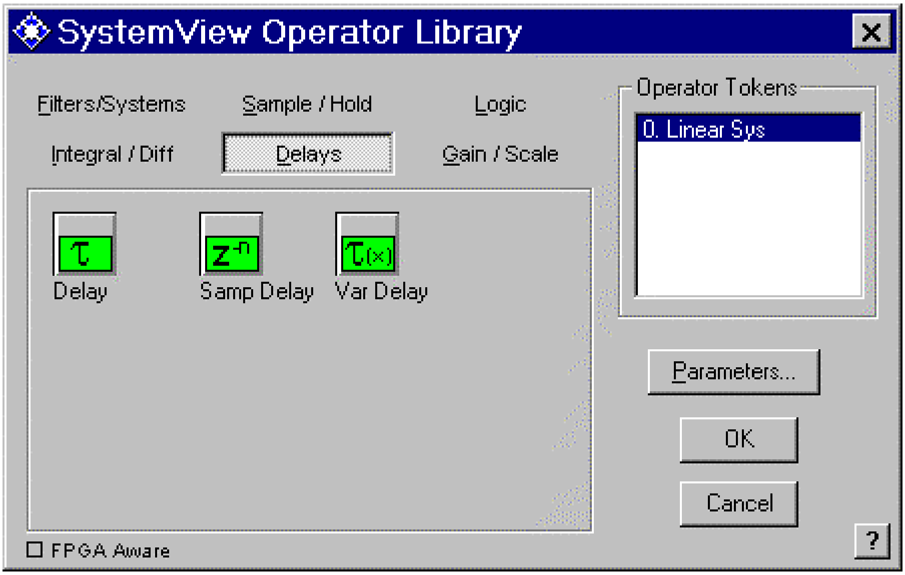
The Integral and Differential Operators perform analytical operations, including a Proportional Integrator/Differentiator.



Integral/Differential Operator Library

5.3.5 Delay Operators

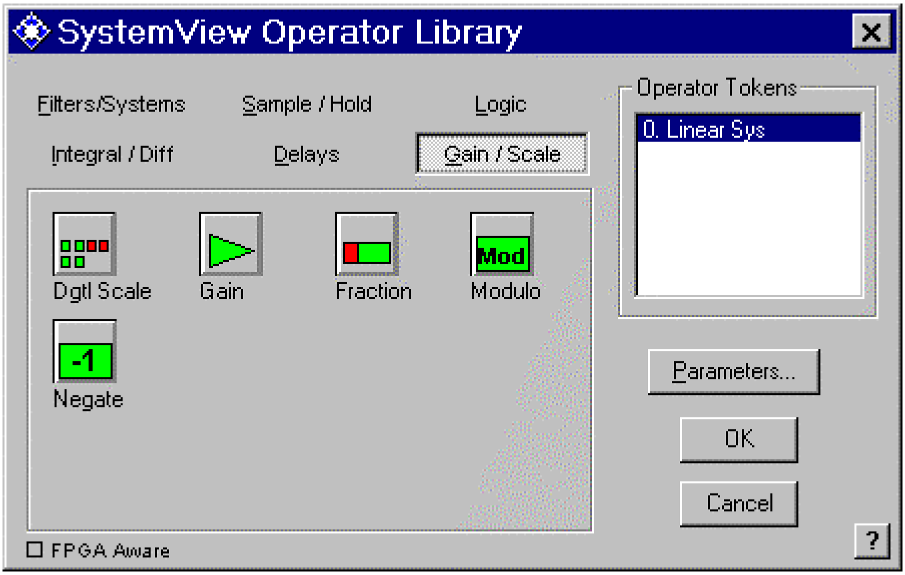
The Delay Operators modify the simulation data stream by adding delay, based on time or the system sample rate.



Delay Operator Library

5.3.6 Gain / Scale Operators

The Gain / Scale Operators perform scalar and modular operations. The Digital (Dgtl) Scale extracts the values represented by the selected bits of a word.



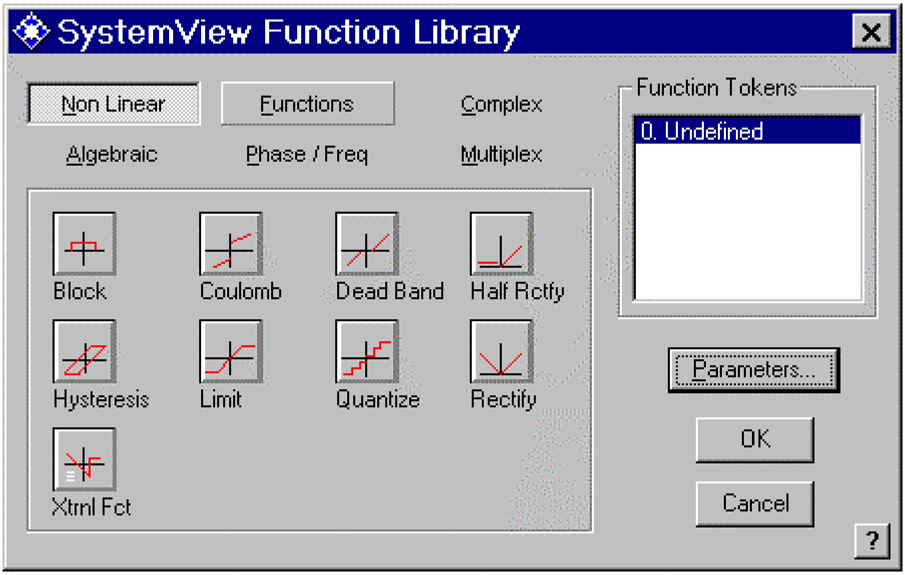
Gain/Scale Operator Library

5.4 Functions

The Function Library contains tokens described by a mathematical expression $y(t)=F\{x(t)\}$, where $x(t)$ is the input time function, $y(t)$ is the output time function and $F\{\}$ is some mathematical expression. The Custom Function in this library allows the user to specify a wide range of additional functions. Chapter 14 provides a detailed description of Functions, their mathematical operations, and their user-defined parameters.

5.4.1 Non Linear Functions

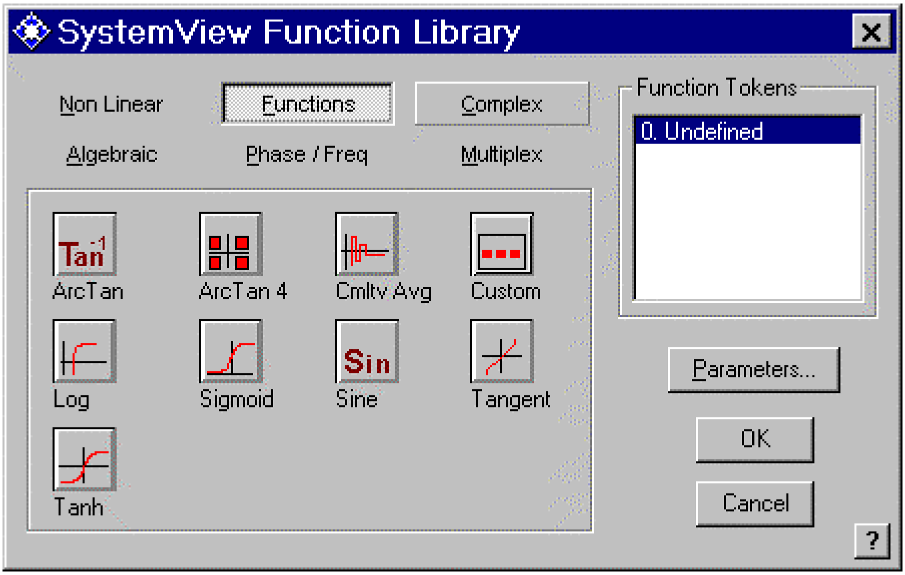
The Non Linear Functions serve as transfer functions. The External Function (Xtrnl) implements a user-defined transfer function from an external file.



Non-Linear Function Library

5.4.2 Functions Group

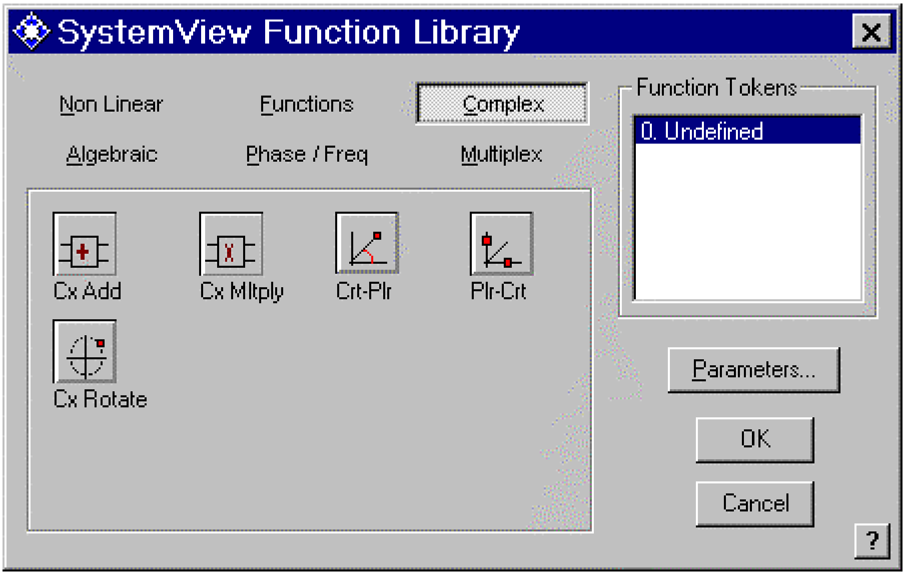
The Tokens in the Functions Group perform transcendental operations, including Arc Tan, Log, Sigmoid, and others.



Functions Group Library

5.4.3 Complex Functions

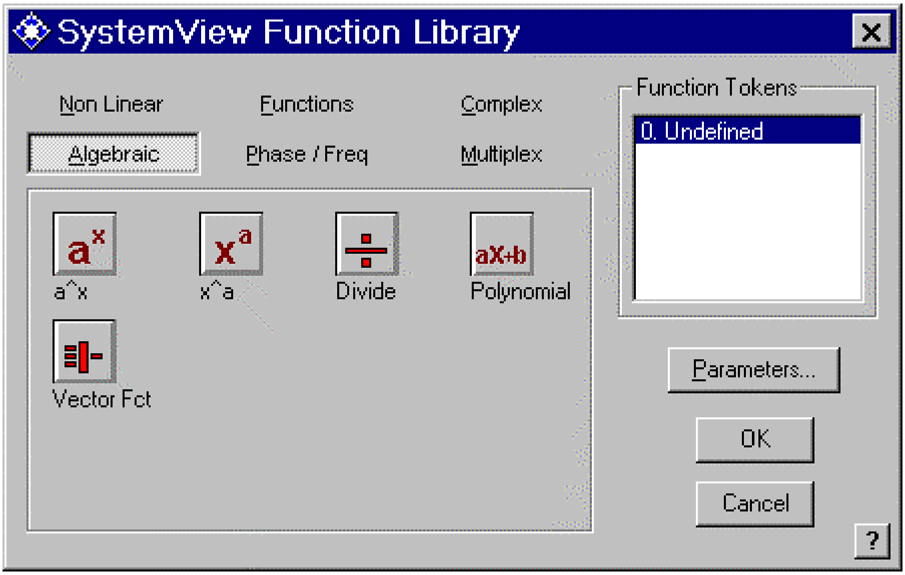
SystemView provides the means for performing complex arithmetic/algebraic operations on input data. It is important to know that the real and imaginary parts of a complex number must be separately defined and maintained in a SystemView simulation. While this may appear somewhat awkward, this constraint is no different than that found in the physical realization of the process being simulated.



Complex Functions Library

5.4.4 Algebraic Functions

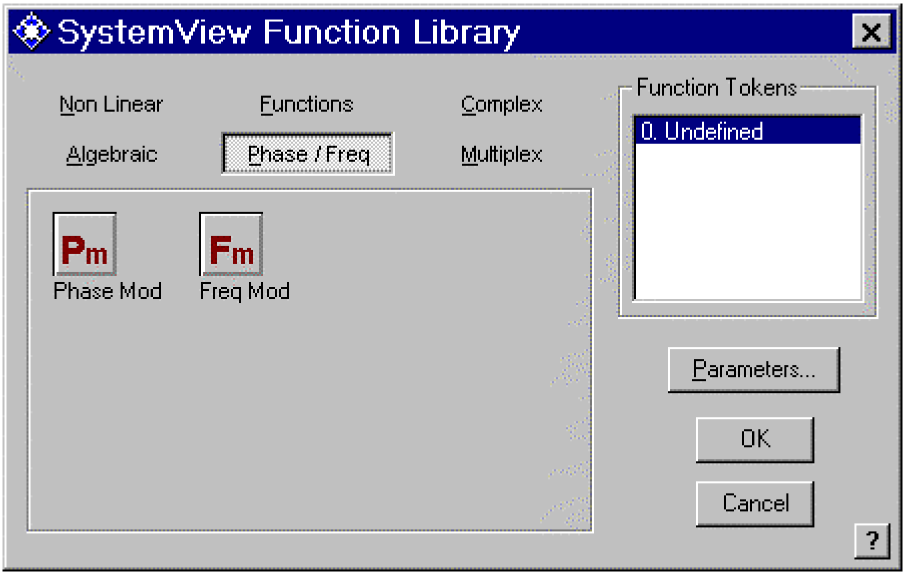
The Algebraic Functions provide algebraic operations on inputs. The Polynomial function can be specified up to the fifth order. The Vector Function takes all of its inputs as a single vector to produce a single output. The method used to calculate the single output is specified by a user-defined parameter.



Algebraic Functions Library

5.4.5 Phase / Frequency Functions

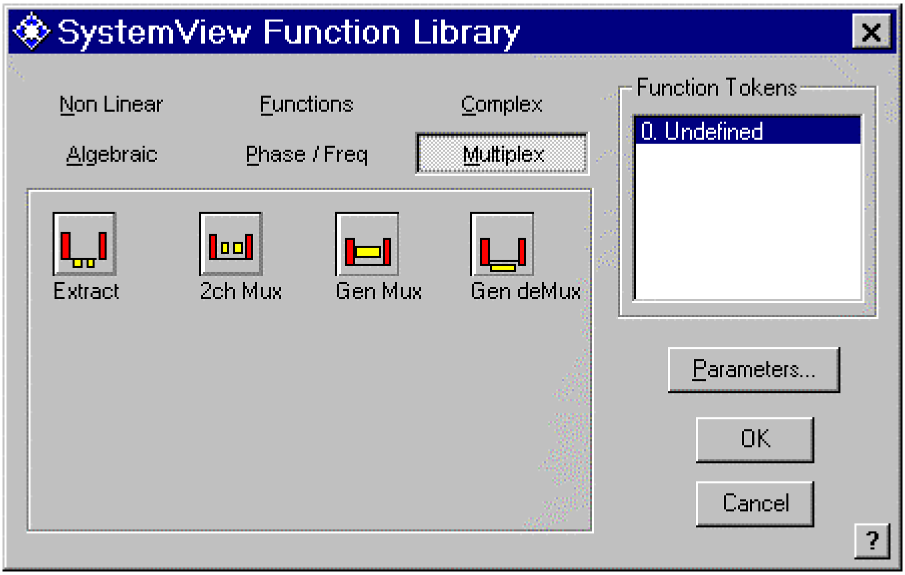
The Phase and Frequency Functions allow the modulation of the phase and frequency of a sinusoid signal used in the simulation.



Phase/Frequency Function Library

5.4.6 Multiplex Functions

The Multiplex Functions provide the means for merging and/or extracting two or more data streams. They are also useful for forming data packets often used in communication systems.



Multiplex Function Library

5.5 Multipliers

X

The Multiplier forms the product of its inputs, and supports up to 20 inputs. If there is only one input, then the output is zero, and a warning message will appear at the beginning of an execution (or when Check Connections is clicked).

The multiplier token may have any number of inputs as is practical, up to 20,and the inputs may have any numerical format. All tokens providing inputs to the multiplier must output data at the same rate and with the same system time marker. Care must be used, when multiplying inputs from time active tokens, since two inputs having different data rates and time markers will not be multiplied properly.

5.6 Adders

+

The Adder token forms the sum of its inputs, and supports up to 20 inputs. If there is only one input, then the output equals the input, and a warning message is displayed at the beginning of an execution (or when Check Connections is clicked).

Inputs to the adder may have any numerical format, but all tokens providing inputs to the adder must output data that at the same rate and the same system time marker. Use care when adding inputs from time active tokens, two inputs having different data rates and time markers may not be added properly.

5.7 MetaSystem I/O

This library has only two tokens, an input to, and an output from, a MetaSystem. The I/O tokens usually have only one input, but may have multiple outputs. A MetaSystem may have several I/O tokens. See Chapter 7 for MetaSystem details.

5.8 Control Logic Scheduler

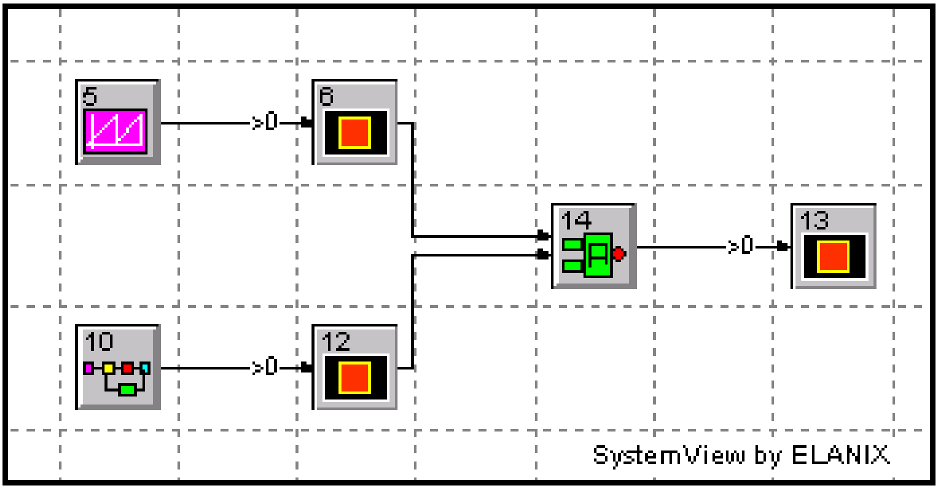
This feature of SystemView allows the designer to control when; different portions of the system operate, or process data as a function of an algebraic/logical expression.

A single scheduler token may control any number of tokens, or MetaSystems. Each scheduler token contains an algebraic/logical expression that is evaluated every system sample. If the expression is true (greater than or equal to the specified threshold), then all tokens and MetaSystems that are controlled by this scheduler token will be called; if the expression is false (less than the specified threshold), then these tokens and MetaSystems will not be called.

The algebraic/logical expression is defined within the scheduler token’s dialog window and can be a function of system parameters (such as current loop and system rate), runtime parameters (such as current time or current sample), parser functions (such as sin () or abs ()), or token output

5.8.1 Complex Control

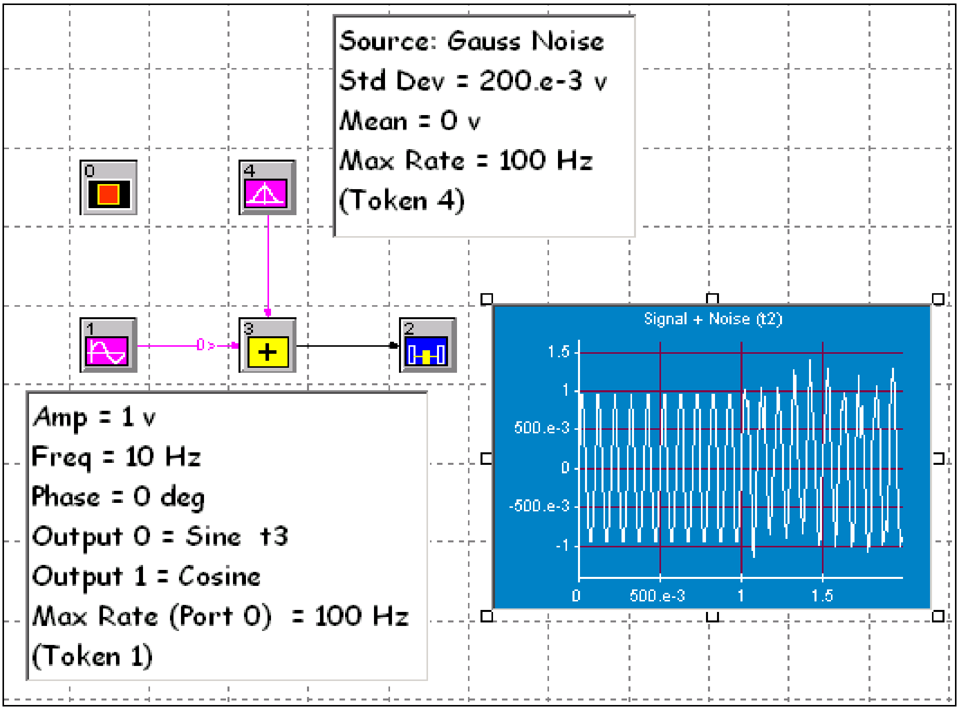
It is possible to use the output of scheduler tokens in conjunction with other SystemView token's to create a complex controller system, without resorting to long algebraic/logical expressions in the scheduler window. Schedulers, MetaSystems, and other SystemView tokens can be used in either series or parallel, to drive the behavior of another scheduler.



In this example, a sawtooth source is driving the scheduler in Token 6, while the output from MetaSystem 10 drives the scheduler in Token 12. The outputs of the scheduler tokens (0 if false and 1 if true) are combined to inform Token 13 to be true only when both Token 6 and Token 12 are identical.

5.8.2 Scheduler Example

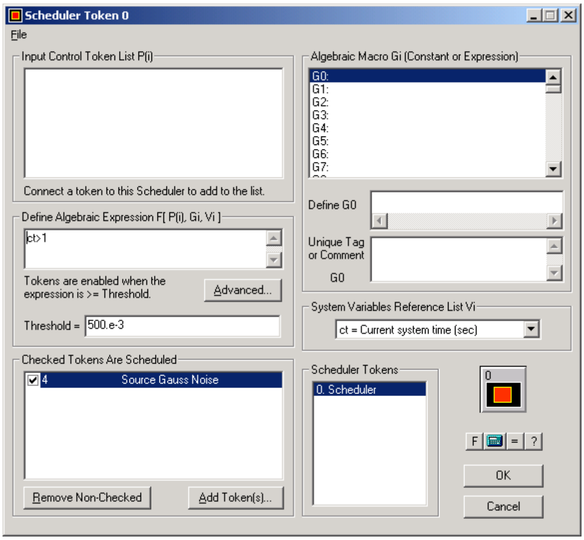
Creating a sine wave source and adding Gaussian noise can create a simple system to display the behavior of the control logic scheduler.



Source: Gauss Noise
Std Dev = 200.e-3 v
Mean = 0 v
Max Rate = 100 Hz
(Token 4)

Amp = 1 v
Freq = 10 Hz
Phase = 0 deg
Output 0 = Sine t3
Output 1 = Cosine
Max Rate (Port 0) = 100 Hz
(Token 1)

Token 0 in this system is the new control logic scheduler. In this system, Token 0 is to control the operating behavior of the additive noise (Token 4). It is set up to control by double-clicking on the Scheduler token, which gives the following dialog:


Scheduler Token 0
File
Input Control Token List P(i)
Connect a token to this Scheduler to add to the list.
Algebraic Macro Gi (Constant or Expression)
G0:
G1:
G2:
G3:
G4:
G5:
G6:
G7:
Define G0
Unique Tag or Comment
G0
Advanced...
System Variables Reference List Vi
Scheduler Tokens
0
F
=
?
OK
Cancel
Remove Non-Checked
Add Token(s)...

Define Algebraic Expression F[P(i), Gi, Vi]: ct>1

Tokens are enabled when the expression is >= Threshold.

Threshold = 500.e-3

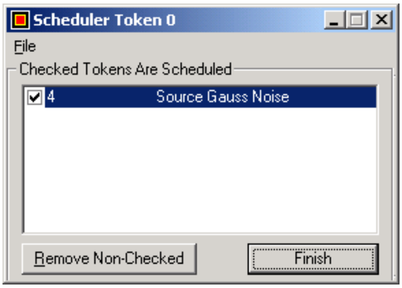
ct = Current system time (sec)

Checked Tokens Are Scheduled: ☑ 4 Source Gauss Noise

Scheduler Tokens: 0. Scheduler

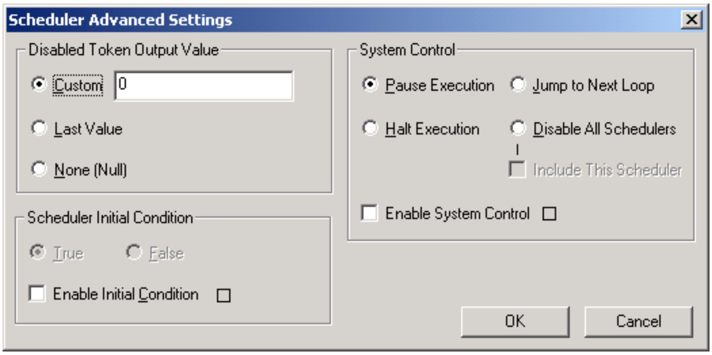
Each scheduler token contains an algebraic/logical operation that is calculated to determine if all controlled tokens and MetaSystems will be called. In this example, we want the noise source to operate only when the current system time (ct) is greater than 1 second. So we define F to be ct>1; therefore, the scheduled tokens (Token 4 – Source Gauss Noise) will not operate until the system time reaches one second.

In order for the scheduler Token 4, to be enabled or disabled under this condition, we go down to the Scheduled Token List below and select the Add Token(s)… button. You can also simply drag and drop tokens into the checked token list. Selecting this button will reduce the Scheduler Token Window down to a smaller size, allowing the user to select the token(s) that are to be enabled/disabled by this scheduler token; the user selected tokens will be automatically added to the Scheduled Token List.



Checked Tokens Are Scheduled: ☑ 4 Source Gauss Noise

The Advanced option in the Scheduler Token dialog box allows the user to define the Disabled Token Output Value, Scheduler Initial Condition, and System Control options. The disabled token may output a custom value, the last value, or null data. The above example sets the disabled token to a custom value of zero. The Scheduler initial condition may be enabled or disabled with either a true or false output. The Scheduler token may also provide System level control regulating the loop control operation, system execution or scheduler activity.



Scheduler Advanced Settings

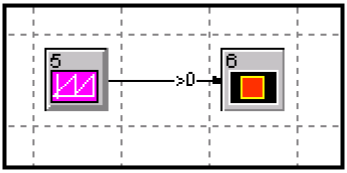
Disabled Token Output Value
- ● Custom: 0
- ○ Last Value
- ○ None (Null)

Scheduler Initial Condition
- ● True ○ False
- ☐ Enable Initial Condition ☐

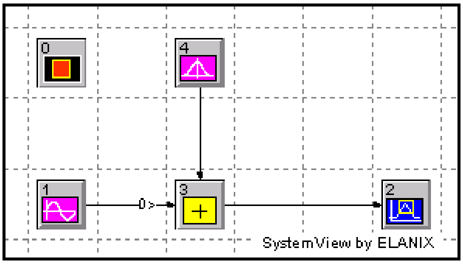
System Control
- ● Pause Execution ○ Jump to Next Loop
- ○ Halt Execution ○ Disable All Schedulers
- ☐ Include This Scheduler
- ☐ Enable System Control ☐

Note that the Scheduler Token dialog box has access to the Global Expressions (Gi) of the system, which can also be used in defining the Scheduler expression F. All of the parser functions available to SystemView users (such as AND, sin (x), and others) are available for use in defining F.

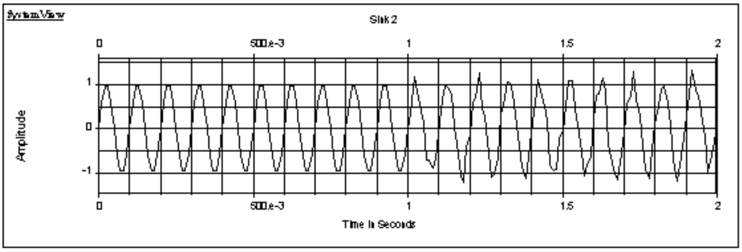
Additionally, the expression F can be defined as a function of the input to the scheduler. The Input Token Control List P (i) is a list of token outputs that are available for use within this scheduler. To include tokens in this list, the user must first connect the output of a given token into the scheduler token of interest, as shown below:



In the example above, the token output from Port 0 of Token 5 may be used in the algebraic/logical expression of the scheduler. So, if we wanted to schedule tokens to operate only when current time is greater than 1 second (ct>1), and when the sawtooth source output is greater than 0.5, then we could define F to be (ct>1)&(P (0)>0.5), where P (0) would appear in the Input Token Control List P (i) identifying Port 0 of Token 5. An option is to use “tkn(5,0)” in the algebraic expression instead of “P(0)”.



In a simple test system, we can run the simulation for 2 seconds, where we should see a sine wave with no additive noise for the first second of the run, (since the scheduler Token 0 sets the output of Token 4 to zero during the first second), and a noisy sine wave for the reminder, as shown in the figure below.



Additional features of the control logic scheduler include:

- During runtime, the center square of the scheduler tokens will display red when they are false and green when they are true, indicating when these tokens are shutting down or activating the controlled tokens and MetaSystems.

- The scheduler token can also enable/disable certain tokens within a MetaSystem or the entire MetaSystem, meaning that all tokens within the MetaSystem will be enabled/disabled.

- The user has the option of defining whether the output of a disabled token is set to zero or to the last value of that disabled token.

- The user has the ability to stop a particular token from being scheduled without removing it from the Scheduled Token List by "un-checking" that token in the Scheduled Token List. The user may remove from the List all tokens that are not being scheduled by selecting "Remove Non-Checked".

- A user-defined threshold can be defined to force values that have an absolute value less than the threshold to be interpreted as zero by the scheduler token.

- Algebraic macro definitions may be imported from or exported to a text file (under the File menu in the Scheduler token dialog window).

- If a token within a MetaSystem is controlled by one scheduler token and that MetaSystem is controlled by another scheduler token, the token will only be enabled when both scheduler tokens are true; the MetaSystem scheduler will override.

5.9 Custom Libraries

The Custom Library supports the design of custom tokens and the integration of third party libraries into SystemView. Custom Libraries, C or C++, is compiled into a 32-bit Windows Dynamic Link Library (DLL) and executed on operating systems supported by SystemView, including Windows 95/98, 2000, and NT. The DLL may include simulations, legacy code, and specialized modules external to SystemView.

A set of examples using custom tokens is distributed with SystemView. The custom library DLL can be found in the svccode.dll folder of the SystemView installation folder. The sample tokens are similar in operation to standard SystemView tokens.

All SystemView tokens are grouped into three types:

- Source tokens, which have no inputs.
- Sink tokens, which have no outputs.
- General tokens, which have both inputs and outputs.

The order of the inputs can be assigned (order dependent) as in a token that performs a division, or unassigned (order independent) like a token that performs an addition. Outputs are always order dependent. Unconnected token inputs are assigned a value of zero by SystemView at the beginning of the simulation. The user can add validation code that will not allow a simulation to continue should this default input value be unacceptable, e.g., a division token that would divide by zero and cause a system fault.

From a development perspective, tokens are also classified into **Time** Passive or Active, in addition to **Type** Passive or Active tokens. Time Passive tokens have output times that match its master input time, whereas Time Active tokens modify the output time for purposes such as decimation. Type Passive tokens pass on the master input data type and its attributes, whereas the Type Active tokens force a set of output data types and their attributes, to distinguish their outputs from the default IEEE double precision type. While most tokens are Type Passive, DSP Library tokens are generally Type Active. Custom library tokens can be any combination of Time and Type Active or Passive.

5.9.1 Environments

This document and the examples, assume the use of Microsoft Visual C++ Integrated Development Environment, version 4.1 or later. Microsoft Visual C++ is currently used in the development of tokens for SystemView. Custom tokens can be developed using any language system that produces 32 bit Windows compatible DLLs. Appendix data shows how a custom library is developed in the Borland C++ 5.0 IDE.

This manual applies to custom library development for SystemView Version 2.0 and later. While Version 2.0 of SystemView supports 32-bit custom libraries developed under earlier or previous versions of SystemView, those custom library object files cannot be used with the Automatic Program Generation (APG) option.

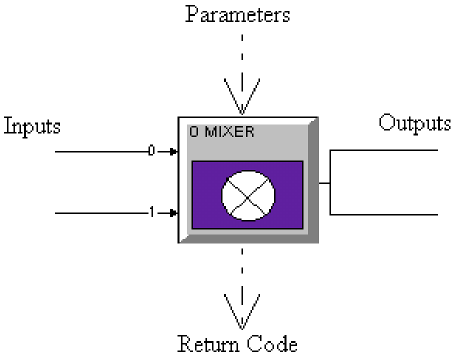
The custom library example that is presented in this manual is APG compatible. An additional text file with extension, .SVA, is required by APG to point to object files that created the custom library DLL. This file must be located in the same folder as the custom library DLL. The .SVA file in the Examples\Custom folder assumes that SystemView was installed in the folder Program Files\SystemView on the C drive.

5.9.2 Features

Custom library functions that have parameters, are capable of being edited while executing a simulation, but dynamic parameter editing is allowed for only one token at a time. Attention should be paid to allocating token instance memory at initialization and clearing the token instance memory pointer at finalization. Editing does not call for finalization before the next initialization call, which can lead to memory leakage during a simulation. Dynamic parameter editing provides only for the initialization of the token that is being dynamically edited and not for other tokens.

Custom library development is oriented toward third party software developers. Features have custom library access restriction in conjunction with SystemView, through a password scheme, with password management left to the custom library. If a PASSWORD function is not wanted, eliminate the PASSWORD entry in the EXPORT section of svucode.def. SystemView permits the use of custom parameter dialogs in place of standard user code dialog. The custom library example EQUALIZER, demonstrates the use of slider controls.

5.9.3 SystemView Operation



A SystemView simulation consists of a system clock, and an interconnected set of tokens. With each system time step, tokens are called in sequence based on the “compiled” execution sequence always beginning with source tokens and ending with sink tokens. During a simulation, SystemView in one of three separate system states calls each token: 1) Initialization, 2) Execution, and 3) Finalization.

To create outputs, a custom token may use (1) token parameters set by the user before or during the simulation, (2) global variables, supplied by SystemView during the simulation, as in the system time window, (3) inputs from other tokens, and (4) internally kept constants and variables. Inputs and outputs are passed via token connections.

All custom library token behavior is defined by member functions residing in one or more Dynamic Link Libraries (DLLs). Each DLL can have up to 80 custom functions. The DLL name is user defined and can be located in any folder.

5.9.4 Custom Library Examples

The Custom folder in the Examples folder has a custom library DLL with functions,

svucode.dll

Included in the same folder are the C source files that created the svucode.dll

svucode.h

resource.h

usercode.rc

svucode.def

usercode.c

svucode.c

Included in the same folder are source files that create an equivalent svucode.dll in C++,

svucode.h

resource.h

usercode.rc

svucode.def

usercode.c

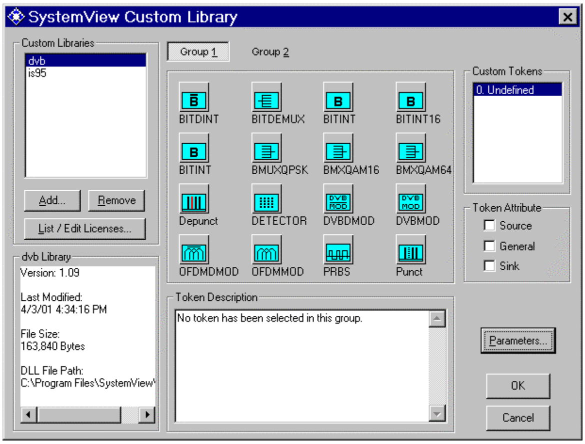
svucode.hpp

svucode.cpp

5.9.5 USING a Custom Token in SystemView

5.9.5.1 Selecting Tokens

A custom library token is defined the same way as a built-in token. Select a new (generic) custom library token from the optional library reservoir and bring it into the design area. Click the right mouse button on the new token and select Library, or simply double click on the new token. A SystemView Custom Library Dialog Window shown below will appear. This dialog window can also be accessed by clicking on the Tools->UserCode->Edit Library menu item.



Custom Libraries: dvb, is95

Custom Tokens: 0. Undefined

dvb Library
Version: 1.09

Last Modified:
4/3/01 4:34:16 PM

File Size:
163,840 Bytes

DLL File Path:
C:\Program Files\SystemView\

Token Description: No token has been selected in this group.

The Custom Library Dialog Window

5.9.5.2 Adding/Removing Dynamic Link Libraries (DLLs)

To add a new custom library, click the Add Library button on the SystemView Custom Library Dialog Window, and a file dialog window will appear. Select the desired custom library file (DLL) and click Open. SystemView will add this library to the current list of custom libraries. To remove a custom library from the current list, select that library with the mouse and click the Remove Library button. Removal of libraries that are no longer used, allows SystemView to be launched faster.

5.9.5.3 Selection of Functions

When a custom library has been selected from the list, each of the custom functions contained in that library will be automatically listed in the Member Functions box. To select a specific function (token), click the function's icon. The Token Description box contains helpful information about the selected function.

All SystemView tokens are designated as a Source (no inputs; one or more outputs), a Sink (one or more inputs; no outputs), or a General token (one or more inputs; one or more outputs). The attributes for the selected function are shown in the Token Attribute box. General input/output (I/O) characteristics for the selected function are displayed by selecting the token and then clicking the right mouse button.

5.9.5.4 Token Parameters

To set custom token parameters, click the Parameters Button from the Dialog Window, to show a standard Parameter Dialog, or a custom Parameter Dialog Window created by the user. In a standard Parameter Dialog, user tokens may have up to nine double precision floating-point parameters and one string parameter. A token may not have parameters, resulting in the Dialog Button being disabled. For the standard Parameter Dialog, the string parameter box is only visible when the string Label field for that function is filled out.

5.9.5.5 Unconnected User Token

An unconnected custom library token will not be included in the simulation execution sequence. SystemView analyzes the token connections, to determine the execution sequence. Any group of connected tokens must have both a source and a sink token or will be considered isolated, and won't be executed.

5.9.6 Building a Custom Library DLL

To build a DLL file, a compiler that supports the generation of Windows DLLs is required. If a DLL can be generated, the compiler will have information on DLL programming. Shown is a partial list of C/C++ compilers that create Windows DLLs: Borland C++ Windows, MetaWare High C/C++, Microsoft Visual C++, Symantec C++.

5.9.6.1 Example Files

The distributed example files can be used as templates for new custom libraries and for practice builds. The files can build C and C++ versions of the example custom library:

svucode.h (“standard header”)

resource.h (“custom parameter dialog window header”)

usercode.rc (“custom parameter dialog window and its controls”)

svucode.def

usercode.c

For the C Version, svucode.c (uses svucode.h)

For the C++ Version, svucode.hpp, svucode.cpp (both use svucode.h)

Usercode.rc and resource.h are used for functions that require custom parameter dialog in place of the standard user code dialog. The example EQUALIZER function that incorporates slider controls, demonstrates a custom parameter dialog.

5.9.6.2 Building the C / C++ Examples

Without loss of generality, the following sections will build the C version of the custom library. If the C++ version is desired, simply use the C++ set of sources. The C and C++ versions both use svucode.h, resource.h, usercode.rc, svucode.def and usercode.c.

5.9.6.2.1 Using Microsoft Visual C++ 4.1

Perform the following steps:

1. Move the source files (e.g., Svucode.h, Resource.h, Usercode.rc, Svucode.def, Usercode.c, and Svucode.c.) into a folder intended for custom library development.

2. Start the Microsoft Visual C++ version 4.1 compiler.

3. Create a Project Workspace in the Integrated Development Environment.

- Click File Menu, then Click New Menu option
- Select Project Workspace
- Click OK
- Select workspace type as Dynamic Link Library (DLL)
- Add project name (e.g., Svucode)
- Choose a project folder. If the project folder is the project name, remove the specific folder name, i.e., the last folder qualifier, so that the project name is not duplicated.
- Click Create.

4. Add the example files into the project.

  - Click Insert Menu
  - Click Files into Project
  - Select the desired files by type or by grouping of types
  - Click OK
  - Repeat above steps if necessary

5. Set up the project for SystemView compatibility.

  - Click Build Menu
  - Click on Settings
  - Click C/C++ tab
    - Set Use Run-time Library box to Multithreaded
    - Set Struct Select Category box; scroll down to "Code Generation"
    - Member Alignment box to 8 Bytes

6. Click OK

- Generate the custom library DLL.
- Click Build Menu
- Click Rebuild All

5.9.6.2.2 Using Microsoft Visual C++ 5.0

1. Move the source files into a folder intended for the custom library development.

2. Start the Microsoft Visual C++ 5.0 compiler.

3. Create a Project Workspace in the Integrated Development Environment.

- Click File menu, then click New and click Project tab
- Click Win32 Dynamic-Link Library, and enter project name (e.g., Svucode)
- Choose project folder. If the folder is the project name, remove the specific folder name, i.e., the last folder qualifier, so the name is not duplicated.
- Click Create new workspace, and click OK

4. Add the example files into the project.

- Click Project menu, then Click Add to Project
- Click Files
- Select the desired files by type or by grouping of types
- Click OK
- Repeat above steps if necessary

5. Set up the project for SystemView compatibility.

- Click Project menu
- Click Settings
- Click C/C++ tab
  - Select Category; scroll down to Code Generation
  - Select Use Run-time Library as Multithreaded
  - Select Struct member alignment at 8 Bytes
- Click OK

6. Generate the custom library DLL.

- Click Build menu
- Click Rebuild All

5.9.6.2.3 Using Microsoft Visual C++ 6.0

Perform the following steps:

1. Move the source files into a folder intended for the custom library development.

2. Start the Microsoft Visual C++ 6.0 compiler

3. Create a Project Workspace in the Integrated Development Environment

- Click File Menu
- Click New
- Click Project tab
- Click Win32 Dynamic-Link Library

- Enter project name (e.g., Svucode)

- Chose project folder. If the project folder is the project name, remove the specific project folder name, i.e., the last folder qualifier, so that the project name is not duplicated.

- Click OK, and the following prompt will appear: "What kind of DLL would you like to create? " Select an empty DLL project, click finish, and then OK.

4. Add the example files (i.e., Svucode.h, Resource.h, Usercode.rc, Svucode.def, Usercode.c, and Svucode.c) into the project.

- Click Project menu, then click Add to Project and click Files

- Select the desired files by type or by grouping of types

- Click OK

- Repeat above steps if necessary

5. Set up the project for SystemView compatibility.

- Click Project menu, then Click Settings

- Click C/C++ tab

- Select Category; scroll down to Code Generation

- Select Use Run-time Library as Multithreaded or Debug Multithreaded

- Select Struct member alignment at 8 Bytes

- Click OK

6. Generate the custom library DLL.

- Click Build menu, then click Rebuild All.

5.9.6.3 Creating A New Custom Library

While a new custom library can be created, the novice custom library programmer can use the distributed SystemView custom library files as source code templates. This section details the steps necessary to add new functions to the source code template, after which the user would build the library. Once the new functions work properly, the user can delete the source specific to the supplied example functions.

5.9.6.3.1 C Source

Perform the following steps:

1. Create a new folder of source files:

    - From the installation Examples\Custom folder, copy the svucode.h, resource.h, usercode.rc, svucode.def, usercode.c, and svucode.c files into a new folder. Rename svucode.def, and svucode.c to *anyname* with the corresponding extensions. You will want to create custom library project *anyname* in this folder later.

2. Change the *anyname.def* file:

    - Add the new function names to the EXPORTS Section.

3. Change the *usercode.c* file:

    - Change the "Number of functions" parameter in the CodeInfo structure to reflect the number of functions that you are adding.

    - Add a CodeInfo function description for each function. This description includes the name, type, input list, output list, and the parameter list.

4. Change the *anyname.c* file:

    - Add your functions. All C custom library functions must be declared as DLL Export long SVUFCN (SVGLOBAL *, SVLOCAL *). A UserCode Function macro is provided for this purpose.

5.9.6.3.2 C++ Source

1. Create a new folder of source files:

- From the installation Examples\Custom folder, copy the svucode.h, resource.h, usercode.rc, svucode.def, usercode.c, svucode.hpp, and svucode.cpp files into a new folder. Rename svucode.def, svucode.hpp, and svucode.cpp to *anyname* with the corresponding extensions. You may want to create custom library project *anyname* in this folder later.

2. Change the *anyname.def* file:

- Add the new function names to the EXPORTS Section.

3. Change the *anyname.hpp* file:

- Add a class definition for each function: class *function:* public Token {};

4. Change the *usercode.c* file:

- Change the "Number of functions" parameter in the CodeInfo structure to reflect the number of functions that are being adding.

- Add a CodeInfo description for each function. This description includes the name, type, input list, output list, and the parameter list.

5. Change the *anyname.cpp* file:

- Add the function entry points. All C++ custom library functions must be declared as "extern "C" DLL Export long SVUFCN (SVGLOBAL *, SVLOCAL *). A UserCode Function macro is provided for this purpose.

- The entry point will call function, CommonExecutor, which will call the class constructor (with the help of an indirect class constructor), destructor or run functions for the token. Consult the C++ example. CommonExecutor is defined in the .hpp file.

- Each *function* class is from a base class named Token with its own constructor, destructor, and run member functions predefined. In the *function* constructor, the number of Inputs and Outputs must be declared to the base class Token constructor as *function::function*( ): Token(#Input, #Output).

- At every time step, the member function void *function::run* ( ) is called.

- If explicit object destruction is necessary at finalization, provide the destructor member function *function::~function*( ) to perform destruction of the *function* object (custom library token).

5.9.6.4 Token Pictures (Icons)

SystemView supports user defined token icons and will provide a default icon if user icons are not supplied. SystemView supports the following file formats for user defined token icons with the generating tools in parenthesis.

- .bmp, bitmap file (MS paint and many other editors)

- .ico, icon files (any icon editor, VC++ AppStudio, VB iconwrks)

- .wmf, windows metafiles (Corel and others)

The icon files for the custom tokens must be in the same folder as the Custom Library DLL. The name of the picture file for a token is defined in the custom library interface structure CodeInfo.

Note: Wmf is the preferred icon format since it is the most scaleable.

5.9.6.5 Testing with the Debugger

One of the major benefits of the Microsoft Visual C++ environment is the ability to execute the latest release of SystemView with the debug version of the custom library DLL. To test in the debug mode, follow the steps listed below:

- Build the debug version of the custom library DLL.
- Load the custom library into SystemView.
- Create a test system using members of the library.
- Save the test system (.svu) and exit SystemView.
- Re-enter the C++ environment.
- Set breakpoints in the code where it is desired.
- Click on Build->Settings->Debug, and input the complete path to the SystemView executable, e.g., "c:\SysVu_32\sysvu_32.exe".
- Click on Build->Debug->Go to start SystemView.
- Open the test system (.svu).
- Change any parameter input to the custom library token.
- Run the system; the debugger will stop at the selected breakpoints.

When SystemView is launched and terminated from the debugger, the custom libraries that are added during the debug session are not necessarily remembered between debug sessions. Following either of the following steps can prevent this situation:

- Create the simulation system; add the libraries prior to the debugging session, and save the simulation system.
- After the custom library has been created, launch SystemView from the debugger, add the libraries, exit SystemView, and restart the debug session.

5.9.7 Custom Library Files

The following source files are available to help create your SystemView DLL, where *anyname* is distributed as Svucode. They can be added to and/or changed as needed:

- Svucode.h
- Resource.h
- Usercode.rc
- Usercode.c
- *anyname*.DEF (required for SystemView to access DLL interface information)
- *anyname*.HPP (used only with the CPP file)
- *anyname*.C or *anyname*.CPP

The following paragraphs explain how these files are used. The Custom Library section has partial source code organized by file type for the distributed example.

5.9.7.1 Svucode.h

The Svucode.h file contains the standard SystemView header information including structure definitions and globally defined macros like UserCode Function. This file should be included with each custom library .C or .CPP file. Svucode.h includes the following structure definitions:

- SystemView Global Variables structure (SVGLOBAL). This structure includes variables associated with the Time Window, and other variables associated with the simulation control such as initialize and finalize.

- SystemView Token input/output structure (SVinout): Every input and output is described with this structure.

- SystemView Token Local Variables structure (SVLOCAL): For each token, this structure specifies the number of inputs, a pointer to an array of pointers to input structures, number of outputs, and a pointer to an array of output structures. Also, the number of parameters, a pointer to the parameter array, the string parameter, a user token memory pointer, the token number, and a pointer to an array of booleans. The booleans indicate whether or not an output for the token is connected. The token number and the array of output booleans are available for SystemView version 4.5 and above.

- SystemView Custom Parameter Dialog Request typedef (UserCodeDialogInfo): This structure transmits requests from SystemView to a specific user code custom dialog such as opening and closing a custom dialog.

- A structure typedef that describes the parameter memory for each token with a custom parameter dialog: A pointer to this parameter memory is available in the both the SVLOCAL and the UserCodeDialogInfo structures. In the example, F10Parms is a structure used both in the custom parameter dialog section in usercode.c, and in the initialization section of the EQUALIZER function found in svucode.c.

5.9.7.1.1 SVGLOBAL

SVGLOBAL contains simulation control information that is available for all tokens. The version and not Version variables can be used as checks for byte alignment and software version verification during debugging. The pointer error Message_ points to a shared message buffer of length errorMessageLength.

Element	Description
version_	Version of SystemView. This should be version 2.0 or greater.
systemStartT_	Start Time in seconds (System Time Window).
systemTimeStep_	Time Spacing in seconds (System Time Window).
systemRate_	Sample Rate in Hertz (System Time Window).
nLoops_	Number of System Loops (System Time Window).
nIterations_	Number of samples in one system loop (System Time Window).
loopReset_	Reset on loop Boolean (System Time Window), True=1, False=0.
tKelvin_;	Temperature (degrees Kelvin) used in some circuit simulations; set in SystemView Preferences Menu.
randomSeed_	Seed for Fixing Random events; set in SystemView Preferences Menu.
useSeedBool_	Boolean to use either a fixed random number generator seed or a random number generator seed based on the platform clock tick. True=1 use fixed seed; set in SystemView Preferences Menu.
nullInput_	Unconnected Input pointer value.

systemT_	System time, changes for each time step of the simulation, (in seconds).
loop_	Current loop, increments at each major loop from 0 to nLoops_-1.
iteration_	Current sample or iteration, increments at each time step from 0 to nIteration_-1.
initialize_	Initialization flag, True<>0 and False=0.
finalize_	Finalization flag, True<>0 and False=0.
errorMessage_	Error message buffer (shared) contains last error message.
nDebugs_	Number of debug outputs.
debug_	User token debugs output.
tweak_	(Reserved)
notVersion_	Bit complement of version.

5.9.7.1.2 SVUinout

Each token input or output is defined as an SVUinout structure, as shown in the following table.

Name	Description
value_	Data value
tokenT_	Token time in (seconds)
tokenRate_	Token rate (Hz)
decimate_	Decimated data is not valid, true <>0,false = 0
dataType_	DSP Library type for value
registerSize_	DSP Library bit length for value
exponentSize_	DSP Library sub bit length for value, exponent length for floating point, and fraction length for integral data types.

5.9.7.1.3 SVLOCAL

The SVLOCAL structure contains information that is unique to each token. The parameter fields are valid only at initialization.

Name	**Description**
nInputs_	Number of connected inputs for a token
input_	Pointer to array of pointers to input structure
nOutputs_	Number of outputs for a token
output_	Pointer to array of output structures
nParameters_	Number of parameters for a token
parameter_	Pointer to array of parameters
stringParameter_	Pointer to a null terminated character string parameter
userMemory_	Pointer to memory allocated for a function to support multiple instances (tokens) of itself
tokenID_	Token number that is displayed on the icon
outputUsed_	Pointer to a Boolean array indicating if the output is connected (0=Not Connected, 1=Connected)

5.9.7.1.4 UserCodeDialogInfo

The UserCodeDialogInfo typedef provides a structure for SystemView requests to a user code custom parameter dialog and also resources to satisfy these requests.

Name	Description
request	Request code
parameter	Pointer to custom parameter structure
info	Pointer to string memory
hSvuWindow	SystemView handle.

5.9.7.2 anyname. DEF

This file exports custom library function names for reference by SystemView. The USERCODE special function name is always required, as well as the names of all other functions that should be made known to SystemView. Two optionally exported special function names are PASSWORD and PARAMETER. In conjunction with SystemView, the PASSWORD function allows the use of code to place access restrictions on its use. If PASSWORD is not in the EXPORT section, this feature is disabled. The PARAMETER function allows the use of code to have custom parameter dialogs. If PARAMETER is not in the EXPORT section, this feature is disabled.

5.9.7.3 anyname. HPP

This file includes the Token base class and the class definitions for each of the example functions. By using inheritance on the Token class, most details of the simulation control can be handled in the base class without requiring any further consideration in a custom library function of the inherited class.

5.9.7.4 anyname. C/CPP

The custom library functions are implemented in these files. Although in the example, the .C and .CPP file contain multiple custom library functions, a one-function implementation per file is easier to maintain and more desirable from an APG perspective. The function CommonExecutor, which is required by the C++ examples, is located in the .CPP file.

5.9.7.4.1 USERCODE.C

The special function, USERCODE, is required for proper custom library operation and must be declared _declspec(dllexport) void WINAPI USERCODE (struct CodeInfo *). A pointer to the structure CodeInfo is returned through the argument when called. SystemView may call this function several times in the course of a session. The structure CodeInfo, describes the set of custom functions that are available from the DLL. In USERCODE, information is input into CodeInfo for each custom library function. All information is then visually presented when a custom token is defined.

Type	**Name**	**Description**	**Limits**
int	ref	Reference Number	
int	numFunctions	Number of functions	1 to 80
svFunction *	functlist	Pointer to an array of svFunction structures	

CodeInfo Structure Description

Type	Name	Description	Limits
int	type	1-source, 2-general, 3-sink.	1,2 or 3
char *	name	Name of function as in source code.	256
char *	abbreviation	A short name for the function is used in custom token identification.	8
char *	comment	Comment is displayed in custom library dialog.	512
char *	pictureFileName	Filename for token picture, .wmf is preferred. The referred file should be in the same folder as the DLL.	256
int	numInputs	Expected maximum number of Inputs.	20
BOOL	assignedinputs	TRUE means input order important, (Divide); FALSE means order is unimportant (Add).	TRUE, FALSE
char **	inlabel	Pointer to an array of input labels for assigned inputs.	20
Int	numOutputs	Number of outputs.	20
char **	outlabel	Pointer to array of output labels.	20
int	numParam	Number of floating point parameters.	9
char **	paramLabel	Pointer to array of parameter labels.	20
double *	paramValue	Pointer to array of default values for the floating point parameters.	IEEE double precision

char *	stringLabel	String parameter label. Argument is activated when character string assigned.	40
char *	stringValue	Default value for string parameter.	300

svFunction Structure Description

The special function, PASSWORD, is required only for user code access restriction and must be declared BOOL WINAPI PASSWORD (struct UserCodePasswordInfo *). The UserCodePasswordInfo structure has the following fields.

Name	**Description**
request	Request code (0:check if required, 1:check password)
prompt	Pointer to a prompt or error message string
password	Pointer to password input provided by SystemView

If PASSWORD is exported in .def file, this function is called when the user code library is added and every time the simulation initializes a token from the library. This function allows SystemView to have the user enter the password string when requested and subsequently verify that string. The initial call to PASSWORD by SystemView determines if the user needs to enter the password string. If so, SystemView calls for a custom password prompt string and then calls for a password string verification. If verification fails, the library is not loaded and the simulation is not run. There are no requirements on how the user code regulates and maintains the password facility. In the example, the coding is contained in the PASSWORD function found in usercode.c. The example demonstrates the use of the registry to store library access information.

The special function, PARAMETER, is used only for custom parameter dialog and must be declared int WINAPI PARAMETER (int functionNo, UserCodeDialogInfo *dialogInfo). The first parameter specifies the function custom dialog being addressed. Beginning with zero, the functionNo corresponds to the index into the CodeInfo structure for each function description. The second parameter points to a UserCodeDialogInfo structure. If PARAMETER is exported in .def file and supports the custom parameter dialog, dialog appears when the parameter window is required or dynamic parameter editing is used. The PARAMETER function also allows the display of parameter values when requested and to save all parameter values in a .svu file. The custom parameter dialog handles the messages sent by its controls through Windows.

5.9.7.4.2 Functions

Each custom library function *must* be declared with the Pascal calling convention to a C routine. For Microsoft C, this is _declspec(dllexport) long WINAPI. For Microsoft C++, this is extern “C”declspec(dllexport) long WINAPI. The return value is a 4-byte type integer. A UserCodeFunction macro is provided for custom library function declaration.

5.9.7.4.3 Function Arguments

All custom library functions receive two pointers as arguments. The first points to the structure SVGLOBAL that contains global simulation variables, and the second points to the structure SVLOCAL that contains token specific data on inputs, outputs, parameters, and the user token memory pointer.

5.9.7.4.4 Processing Flow

In a system simulation, each token is called with three separate SystemView states. The states are initialization, execution, and finalization. The initialization call occurs at the beginning of simulation loop iteration, and the finalization call occurs at the end of the simulation loop iteration. While in the execution state, each token is called at every system time step. At the time step, the token can take the inputs pointed to by SVLOCAL, global data from SVGLOBAL, and internal data to produce outputs that are referenced in SVLOCAL. In addition, the token can decimate its outputs, i.e., produce a null output. The decimate_ Boolean, for each input and output indicates whether or not its value is valid for that time step.

If a token has multiple inputs, some can be decimated and others cannot. Even if the inputs are not decimated, they may have different token times. Default processing of multiple inputs by custom library functions requires that a master input be chosen. This becomes a reference for output time and output rate, e.g., if the master input is decimated, the outputs are decimated. If other input times do not match the master input time within a reasonable time tolerance, those inputs are considered sampled to a zero with one exception. When an input is feedback, the input is not sampled to a zero, if the feedback input time is delayed by one master token time step. Initially, each output value is set to zero or to an initial value that may be specified for an implicit feedback delay. Therefore, *do not* set output values and their decimation Booleans at initialization.

For source and general tokens, there are descriptor variables for each token output. For most tokens, these outputs descriptors have the same values as the input descriptors. Each input and output for all tokens has the following attributes:

	Initialization	**Run**	**Nominal Output Value**	**Active Output Limits**
value_	Not assigned	Computed		
tokenT_	Assigned	Assigned	Input value	Should be greater than or equal to system time.
tokenRate_	Assigned		Input value	Must be less than or equal to system rate.
decimate_	Not Assigned	Assigned	Input value	TRUE or FALSE
dataType_	Assigned	Assigned	Input value	See DSP Library manual.
registerSize_	Assigned	Assigned	Input value	See DSP Library manual.

5.9.7.4.5 Sources and Sinks Processing Flow

Since source tokens have no inputs, they must completely fill their output structures. The following table shows when and what values are assigned to source tokens. Sink tokens do not assign values to an output structure, since they have no output.

	Initialization	**Run**	**Nominal Output Value**
value_	Not assigned	Computed	
tokenT_	Assigned	Assigned	SystemT_
tokenRate_;	Assigned		systemRate_
decimate_;	Assigned		0 (FALSE)
dataType_;	Assigned		5 (Double Precision)
registerSize_	Assigned		64
exponentSize_	Assigned		11

5.9.7.4.6 Returned status value

The custom library function returns a 4-byte integer status value. When the function successfully finishes a call, a status value of 0 should be returned. Otherwise, system execution will terminate with a message box reporting the non-zero return value, the token number, and the token error text in the global error message buffer. Reserved return codes -1001, Controlled execution termination, 1002 Break point, 1003 Alert, and 1004 that causes SystemView to go to the next simulation loop, are used for APG stop sinks:

5.9.7.4.7 CommonExecutor and the class Token

In the C++ example, the base class Token from which all custom library function classes are derived, performs standard processing as indicated by the Token initializer. The function CommonExecutor, connects the SystemView function call to the constructor, run and destructor member functions of the token. In CommonExecutor, static variables of the Token base class, like the pointers to the SVGLOBAL and the SVLOCAL structures, are initialized. Standard token processing at initialization include:

- Construct the token instance object and save its address to userMemory.
- Find the master input, if the token is not a source.
- Calculate token time step.
- Calculate token time step error tolerance.

Standard token processing for non-sink types at execution include:

- Pass the master input attributes to outputs.
- When the master input is decimated, outputs are also decimated.

Standard token processing at finalization include:

- Return token instance object, pointed to by userMemory_, back to free the memory pool.

5.9.8 Custom Library Topics

5.9.8.1 Common Problems

Some common custom library problems include the following:

The function name is missing from the definition file (.DEF). Information about the new function must be added to the CodeInfo structure, the function must be defined in a .C or .CPP file, and the function name placed in the EXPORT section of the .def file.

There is an inconsistent function specification between information placed in the CodeInfo structure and the custom library function declaration, such as a mismatch in the number of inputs.

Wrong default alignment for structure members was specified. Value should be 8 bytes.

Incompatible versions of custom library support are found. Previous versions of custom library are still supported, but only as a custom library without APG compatibility.

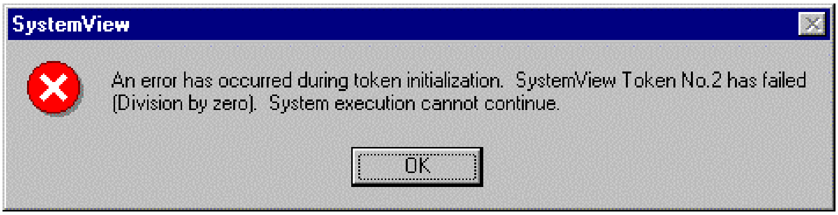
A 16-bit Windows DLL was generated where a 32-bit DLL is required.

The parameter array, which is only valid in the initialization state, is used either in the execution or finalization state.

A custom library DLL sharing violation occurs when an attempt is made to replace a DLL that is currently in use.

Effects of incorrectly initialized pointers cause SystemView to fail.

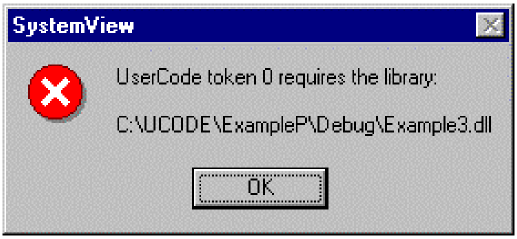
An illegal coprocessor state causes SystemView to halt (e.g., a calculation that produces an infinity value). For these errors, a message is displayed as shown.


SystemView

An error has occurred during token initialization. SystemView Token No.2 has failed [Division by zero]. System execution cannot continue.

Abnormal behavior caused by a full disk.

Opening the SystemView simulation file (e.g., my.svu), before loading the required custom library (.dll), generates the following message. No existing library can be on the list of user code libraries, with the same name that can acceptably serve as a substitute. Visually, the affected custom library token changes to a generic custom token.


SystemView

UserCode token 0 requires the library:

C:\UCODE\ExampleP\Debug\Example3.dll

Errors may occur when token specific data is stored to static variables.

Pointers to allocated memory are not set to NULL after being returned to free memory

Errors occur in C custom libraries when output data types are not updated for Type Passive tokens, especially, with DSP token.

Not updating the output time or rate can produce errors In C++, the base token automatically performs the updates for Type Passive tokens.

Debugging the Windows Dialog Procedure, associated with a user code function's custom parameter dialog, may run into concurrency problems when Windows messages arrive and when the source code executes at a breakpoint.

5.9.8.2 SystemView Custom Library Error Messages

For the listed problems, follow the procedures in the order specified.

Error	Description
1	An internal error has occurred. (1) Restart SystemView. (2) Reboot and start SystemView. (3) Remove all custom libraries, restart and add the custom libraries. (4) Recreate the .SVU file. (5) Contact Eagleware-Elanix
2	Library cannot be loaded. (1) Check the file path for the library. (2) Check library to be a 32bit Windows DLL. (3) Check for other DLLs that the library may require. (4) Recompile the custom library. (5) Contact Eagleware-Elanix
3	Attempt to delete a library not on the list. (1) Restart SystemView. (2) Remove all custom libraries from SystemView, restart SystemView, and add back the custom libraries. (3) Contact Eagleware-Elanix
4	Custom library description not found in the library. (1) Check for SVUCODE or USERCODE routine in the custom library, and if the routine exists, check the function declaration. (2) Check for .DEF file, and if it exists, check the list of custom library functions under the EXPORTS section.
5	Cannot refresh library information, out of memory. (1) Close other applications. (2) Configure system for greater virtual memory. (3) Add memory.
6	Cannot add library to the list (out of memory). See fix in error 5.
7	Cannot load library information (out of memory). See fix in error 5.
8	Custom library description cannot be accessed. (1) Check CodeInfo member numFunctions, it may be zero. (2) Is CodeInfo pointer modified? It can't change.
9	# Is not a valid number of functions. Number of functions in custom library must be positive and not exceed maximum allowed (128).

	must be positive and not exceed maximum allowed (128).
10	Invalid pointer to the description of function #.
11	Function # is not a source, general, or sink. Unknown token type #.
12	Invalid pointer to the name of function #.
13	Function # name too long (over 256 bytes).
14	Function # name is blank. Number of functions in custom library description must correspond to the actual number of functions, and listed consecutively.
15	Invalid pointer to the abbreviated name of function #.
16	Function # abbreviated name too long (over 9 bytes).
17	Invalid pointer to the comment of function #.
18	Function # comment too long (over 512 bytes)
19	# Is not a valid number of parameters for function #. Number of parameters must be positive and not exceed maximum allowed, (9).
20	Invalid pointer to the parameter name # of function #.
21	Function # parameter name # too long (over 21 bytes).
22	Invalid pointer to default value for parameter # of function #.
23	# Is not a valid number of assigned inputs for function #. Number of assigned inputs must be positive and not exceed maximum allowed, (20).
24	Function # is declared a source and may not have inputs.
25	Function # is declared a sink or general and must have inputs.

26	Invalid pointer to the input label # of function #.
27	Function # input label # too long (over 21 bytes).
28	# Is not a valid number of unassigned inputs for function #. Number of unassigned inputs must be positive and not exceed maximum allowed, (20).
29	Function # has labels for unassigned inputs. Unassigned inputs cannot be labeled.
30	# Is not a valid number of assigned outputs for function #. Number of assigned outputs must be positive and not exceed maximum allowed, (20).
31	Function # is declared a sink and may not have outputs.
32	Function # is declared a source or general and must have outputs.
33	Invalid pointer to the output label # of function #.
34	Function # output label # too long (over 21 bytes).
35	Invalid pointer to the picture file name of function #
36	Function # picture file name too long (over 256 bytes).
37	Invalid pointer to the label of string parameter of function #.
38	Function # label of string parameter too long (over 41 bytes).
39	Invalid pointer to the default value for string parameter of function #.
40	Function # default value for string parameter too long (over 300 bytes).
41	Function # is not found in the library.
42	Execution internal error has occurred. See fix in error 1.

5.9.8.3 Memory Usage

Custom library functions have variables that require persistent memory and other variables that do not. Variables that need to exist only during a function call can be declared as variables in the automatic storage class. Memory-consuming variables like very large structures or arrays should be allocated. *If you make these allocations in the run state, don't forget to delete them when you exit the function!* Persistent variables must exist between function calls. They can be allocated at initialization and should be de-allocated by the custom library function at the finalize call to the function. For this last call, the SVGLOBAL structure variable, *finalize_*, will be set true (! =0). In the C example, each function must explicitly check this Boolean and perform memory de-allocation and cleanup. In the C++ example, the class destructor called from the CommonExecutor routine automatically handles memory de-allocations and cleanup.

Note: It is strongly recommended that pointers to allocated memory be set to NULL after the object is returned to free memory. In particular, the instance pointer userMemory_, in the SVLOCAL must be set to NULL after its memory is returned to Windows.

5.9.8.3.1 UserMemory

The userMemory_ variable holds the reference to a token instance. Each instance of a token, receives its own SVLOCAL structure, thus its own userMemory_ variable. Persistent memory of a specific instance is preserved between calls when it is stored in an allocated structure that is pointed by userMemory_. If they are stored in static memory, the second instance of the same token will corrupt those variables. in the simulation, but this technique is not recommended.

5.9.8.4 DSP Tokens

DSP arithmetic types for input and output can be manipulated for the custom library. Every input and output has an SVinout structure that has three members for DSP types.

Numerical values in the DSP library, are described by data type, register size, and exponent size. A description of each is shown in the following table.

Item	Description
dataType_	Valid types are 1 to 10 listed in the following chart. The standard SystemView type is 5, representing IEEE double precision floating point.
registerSize_	Represents the word size. The standard SystemView size is 64 bits
exponentSize_	Represents the number of the bits allocated to the exponent. The standard SystemView size is 11.

A numerical value in the DSP library is described by dataType, register Size, and exponent Size, depending on type. SystemView passes IEEE double precision floating-point values between tokens, on all data types.

The table shows data type, and the related register and exponent size.

DataType	Description	Register Size	Exponent Size
0	Reserved		
1	Unsigned integer	1-52	0
2	Signed integer	1-52	0
3	Signed fraction	1-52	0-51 (fraction Size)
4	IEEE single	32	8
5	IEEE double	64	11
6	IEEE general	2-64	1-11
7	C4x short	16	4
8	C4x single	32	8
9	C4x extended	40	8
10	C4x general	2-64	1-10

5.9.8.4.1 Type Passive Tokens

Passive tokens pass through master input type information, without validating or changing.

5.9.8.4.2 Type Active Tokens

Type Active tokens modify and/or validate data, based on the type. Source tokens are Type Active, generating data with data Type=5. Most DSP tokens are Type Active.

5.9.8.4.3 DSP Library Output Flags

Flag outputs from a DSP library token such as Overflow (OF), Carry (CF), Zero (ZF), Sign (SF)or Underflow (UF), are special outputs. These Flag outputs have the dataType=1 (unsigned integer) with a registerSize=1. A flag value of zero corresponds to a false condition, and a flag value of one corresponds to a true condition. These outputs indicate the state of internal computations by the issuing token.

5.9.8.5 Multiple Input Tokens

Within the same system time step, if a token has multiple inputs, some can be decimated, and others cannot. Even if inputs are not decimated, they may have different token times. The default processing of multiple inputs by custom library functions require that a master input be chosen. This master input becomes a reference for output time and output rate, e.g., if the master input is decimated, the outputs are also decimated. If other input times do not match the master input time within a reasonable time tolerance, those inputs are considered sampled to a zero with one exception. A feedback input is not sampled to a zero, if the feedback input time is delayed by one master token time step. This default processing for multiple input tokens is used for most SystemView tokens.

5.9.8.6 Changing Output Rate

If a function changes its output rate relative to its input rate, the function becomes Time Active. At initialization this function must show its output rate for each output of the token, as in a simulation loop, and this output rate must not exceed system rate. Both token rate and start time for all outputs are set at initialization. If the output rate is less than the system rate, the output decimation Boolean must be set true, otherwise, it is set false.

Do *not* set output values at initialization! For output times that are not a multiple of the system time step, an output event clock must be constructed and updated by the output time step. At runtime, an output event is declared when the clock time is less than or equal to the system time plus a time tolerance, epsT. For SystemView, epsT is set to the system time step divided by 1024.

5.9.8.7 Random Number Generation

If random number generation is required within a custom library function, that function should call the routine SetRandomSeed at initialization that can be found in the example source file Usercode.c. SetRandomSeed, provides a standard method of seeding the C run time random number generator, located in the SystemView library. This same function exists in the APG environment, and does not need to be included in custom library object modules that are linked in the APG option. The distributed example Usercode.obj contains all custom library elements that should not be linked with APG.

5.9.8.8 Password Protection

To protect a custom library DLL, code must be incorporated into the custom library, which calls an exported function in your DLL named PASSWORD. This interface function allows the user to add the custom library to its list of available custom libraries. The user may create a secret registry entry, and check the entry when the custom library is in use to see if authorization has already been given or if a password check is required.

PASSWORD is called every time the custom library DLL is added to SystemView and every time a simulation containing a custom library function from the DLL is initialized. Coding for PASSWORD should check to determine if the authorization has been given, before requesting the password dialog. If the PASSWORD function is not present in the .DEF file EXPORT section, the custom library may be used. The protocol is as follows:

- SystemView calls the PASSWORD function with a request for the dialog prompt. The return value indicates (1) the authorization to use the custom library has already been given or (2) SystemView should display a dialog requesting it. The prompt string for the dialog must be filled, if the dialog is requested.

- If the custom library requests the dialog, SystemView displays the prompt string and prompts the user for the password string. The PASSWORD function is again called with a request to check the password. The return value indicates if the input and the password match. If it fails, a message box is displayed for the failure message, or a success message. If the prompt string is null, it displays a default success message. For implied success with no further dialog, return a value of -100.

- If the password is accepted and the custom library is not already on the list of available custom libraries, the custom library is added.

An update request to the PASSWORD function can be initiated by the user, to modify the authorization for a custom library that is already on the SystemView list of available libraries. Pressing the Update License button in the Custom Library Editor will activate an update request. The protected custom library can either (1) handle the request with its dialog within that one call, or (2) it can destroy any authorization record and repeat the previous steps. An "authorization record" is any proprietary mechanism the custom library uses to remember the original authorization.

The following example written in the C language illustrates the basic functionality of the PASSWORD function.

```
#include  <windows.h>
#include  <stdio.h>

/*  Placeholders for proprietary functions for managing the authorization record
 *  To be developed by the custom library programmer.
 *  Their return value of TRUE indicates success and FALSE indicates failure
 */
extern BOOL DoesPasswordMatchPrompt(char *, char *);
extern BOOL CreateAuthorizationRecord();
extern BOOL DeleteAuthorizationRecord();
extern BOOL IsAuthorizationRecordPresent();

/*
 *  Possible SystemView requests
 */
#define   REQUEST_CHECK_AUTHORIZATION        1
#define   REQUEST_MODIFY_AUTHORIZATION       -1
#define   REQUEST_CHECK_PASSWORD             0

/*
 *  Possible PASSWORD function return values
 */
#define   PASSWORD_UNAUTHORIZED              0
#define   PASSWORD_AUTHORIZED                1
#define   OTHER_ERROR                        -100
```

```
/*
 *  Structure for data exchange between the custom library
 *  PASSWORD function and SystemView
 */
typedef struct
{
    BOOL request;
    Char *prompt;
    Char *password;
} UserCodePasswordInfo;

/*
 *  Basic PASSWORD function
 */
BOOL CALLBACK PASSWORD(UserCodePasswordInfo *PasswordInfo)
{
    if (PasswordInfo->request)
    {
        /*
         *  Check or modify authorization record
         */
        if (PasswordInfo->request == REQUEST_CHECK_AUTHORIZATION)
        {
            /*
             *  If there is an authorization record, then the library can be used
             */
            if (IsAuthorizationRecordPresent())
            {
                return PASSWORD_AUTHORIZED;
            }
        }
        else if (PasswordInfo->request == REQUEST_MODIFY_ AUTHORIZATION)
        {
            /*
             *  Delete the authorization record
             */
            if (!DeleteAuthorizationRecord())
            {
                MessageBox(NULL,
                    "Error deleting the authorization record.”,
                    "Library Error",
                    MB_ICONERROR);
                return OTHER_ERROR;
            }
        }

        /*
         *  Ask SystemView to display a password dialog
         */
        sprintf(PasswordInfo->prompt, "Please enter password.");
        return PASSWORD_UNAUTHORIZED;
    }
    else
    {
        /*
         *  Check the password returned by the SystemView
         *  password dialog, it must match the prompt
         */
        if (DoesPasswordMatchPrompt(PasswordInfo->password, PasswordInfo->prompt))
        {
            /*
             *  Create an authorization record
             */
            if (!CreateAuthorizationRecord())
            {
                MessageBox(NULL,
                    "Error creating the authorization record.”,
                    "Library Error",
                    MB_ICONERROR);
                return OTHER_ERROR;
            }
            return PASSWORD_AUTHORIZED;
        }
        else
        {
            sprintf(PasswordInfo->prompt, "Bad password.");
            return PASSWORD_UNAUTHORIZED;
        }
    }
}
```

The custom library DLL remains loaded throughout the PASSWORD protocol; so static variables can be used. The original prompt string and password string is returned when checking password, potentially eliminating a need for static variables.

The PASSWORD function alone may not be sufficient protection. The authorization should also be checked during initialization of the custom library tokens. That may present a problem if the library is intended for use with APG. In APG modules, the PASSWORD function is not called, and should not be incorporated into the module like the USERCODE function. An APG module may be run on a different machine. Consider placing the verification code in a separate module and removing protection in the APG by linking a different verification module through the input object list, .sva file.

For manipulating the registry, use the two Win32 APIs, named RegSetValue and RegQueryValue. The following code fragments illustrate how these APIs are used.

```
char *buffer = “authorized”;
long size = 10;
RegSetValue(HKEY_CURRENT_USER,
    “Software\\VB and VBA Program
Settings\\SystemView\\License\\MyUserCode”,
    REG_SZ,
    buffer,
    size);
char buffer[16];
long size;
RegQueryValue(HKEY_CURRENT_USER,
    “Software\\VB and VBA Program
Settings\\SystemView\\License\\MyUserCode”,
    buffer,
    &size);
```

For both APIs, the first argument is one of the root registry keys, and the second argument is the subkey. For RegSetValue, the third argument must be REG_SZ. REG_SZ is defined when the header file, windows.h, is included. The buffer argument is the address of the string to be stored or of a string to receive a registry value.   The size argument is the length of the string to be stored or the length of a string to receive a registry

5.10 Conversion from the Custom Library (User Code)

In a previous version, the User Code Option was renamed Custom Library. The changes include the splitting of the SVU input argument structure into two new structures, SVGLOBAL and SVLOCAL, the addition of an optional string parameter, changes to the standardized header file (Svucode.h), and the way inputs and outputs are accessed. Procedural steps are presented for the C and C++ examples. The following specifics apply to both examples.

- CodeInfo replaces structure SVUCodeInfo. Since the formats are different, it is easier to copy the strings and refill the fields with the previous values. The USERCODE function replaces the SVUCODE function.

- Split off the DLL specific functions like DLLMain and USERCODE, into its own file, to make the custom library more APG compatible.

- SystemView uses an 8-byte structure member alignment. For Microsoft, click Building->Settings… select C/C++ tab, in the category box choose “code generation” and set the structure alignment to 8 bytes. While in “code generation”, choose the Multithreaded run time library.

5.10.1 C Example

- Reference the new structures (#include Svucode.h) in the code file.

- Change the token function declaration to have “two pointers”, e.g., (svg, svl). A UserCodeFunction macro is provided for this.

- Replace the following references to reflect the changes in the header file, Svucode.h. If desired, reintroduce the #define(s) for the SystemView variables.

    - FALSE replaces VBFALSE.

    - TRUE replaces VBTRUE.

    - svl->parameters_ replaces parameter.

    - svg->errorMessage_ replaces errorMessage.

- svl->input_[0]->value_ replaces svu->input_[0].
- svl->output_[0].value_ replaces svu->output_[0].

5.10.2 C++ Example

Reference the new structures (#include Svucode.h) in the code file.

Replace the token class with the updated token class in Svucode.hpp.

Replace the function, CommonExecutor, with the updated source from Svucode.cpp.

Change the token function declaration to have “two pointers”, e.g., (svg, svl). A UserCodeFunction macro is provided for this.

Update the call to CommonExecutor, which now has three arguments.

Replace the following references to reflect the changes in the header file, Svucode.h. If you wish, you can reintroduce the #define(s) for the SystemView variables.

FALSE replaces VBFALSE.

TRUE replaces VBTRUE.

svl->parameters_ replaces parameter.

svg->errorMessage_ replaces errorMessage.

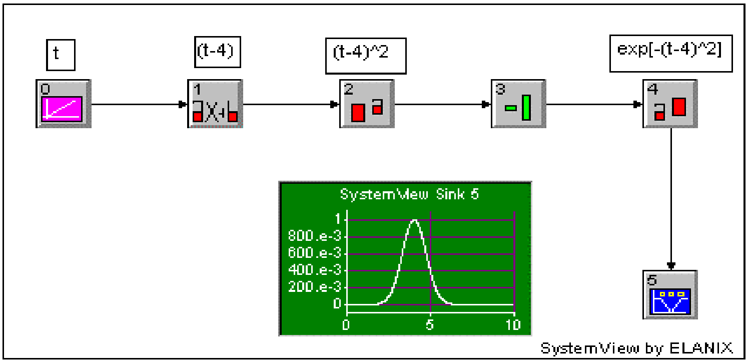
svl->input_[0]->value_ replaces svu->input_[0].

svl->output_[0].value_ replaces svu->output_[0].

5.11 Examples

5.11.1 Gaussian distribution

To illustrate the use of a Function token, the Gaussian function has been generated using the system shown below. $y(t) = e^{-(t-4)^2}$, $0 \le t \le 8$



In the Time Window set the Start Time to 0, Stop Time to 10 seconds, and Sample Rate to 100 Hz.

Launch a Source token. Define as a Time token with the gain parameter = 1. This token is in the Aperiodic Sources.

Launch a Function token. Define it as a Polynomial under the Algebraic functions. Set the linear coefficient x^1 to 1, and the constant coefficient x^0 to -4. Connect the Source token to this polynomial function.

Launch a Function token. Define it to be "X^a" under the Algebraic functions token group. Enter two as the exponent parameter.

Connect the output of the Polynomial token to the “X^a” token for an output of $(t-4)^2$.

Launch an Operator Negate token defined as “-1” (sign inversion) under Gain / Scale.

Connect the output of the X^a token to the “-1” Negate token, for the output $-(t-4)^2$.

Launch a Function token, using “a^X” in Algebraic functions and default parameter “e”.

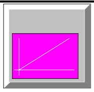
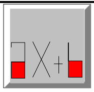
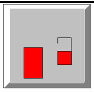
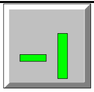
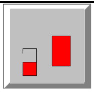
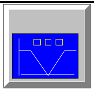
Connect the negate token output to the “a^X” token.

Finally, launch a Sink token. Define it as SystemView under the Graphic Display group.

Connect the output of the “a^X” token to the Sink token, and click the Run button.

Click on the Analysis Window button to examine the output of Sink 5.

To verify the correct function, redefine the Sink as a Data List, and examine the function

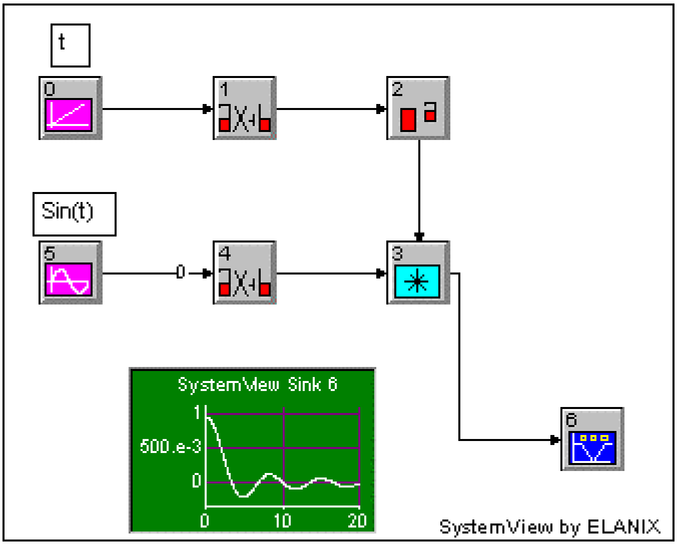
Order	Token	Name	Group	Parameters
0		Time	Aperiodic Source	Gain =1 Offset =0
1		Polynomial	Algebraic Functions	X^5 Coeff=0 X^2 Coeff=0 X^4 Coeff=0 X^1 Coeff=1 X^3 Coeff=0 X^0 Coeff=-4
2		X^a	Algebraic Functions	Exponent = 2
3		Negate	Gain / Scale Operators	None
4		a^X	Algebraic Functions	Base = *e (default*)
5		System View	Graphic Sinks	None

Summary of tokens used in the Gaussian example

As summarized, function tokens were used in the example in three ways: 1) as a polynomial to shift the time function, 2) as a squaring operation, and 3) as the exponentiation operation. There are other ways to achieve the same result. For example, write: $-(t-4)^2 = -t^2 + 8t - 16$ and use the Polynomial token to directly produce the right-hand side of the equation.

5.11.2 Division

To perform a divide function in SystemView, use the Divide token in the Function library. An alternate method is to invert the denominator and multiply. This is accomplished by using the X^a Function with the exponent *a* = -1. For example, create *Sin*(*t*)/*t* using the system shown below.



Division using the invert and multiply technique

Division by zero is undefined. You can protect against this situation by offsetting the denominator by a very small number (1e-10 in the above example). For this example, the system time is set to a range from 0 to 20 seconds, with a sample rate of 20 Hz.

The following summarizes the tokens used in this example. When connecting the Sine Source Token (5) to Polynomial Token (4), select the Sine output.

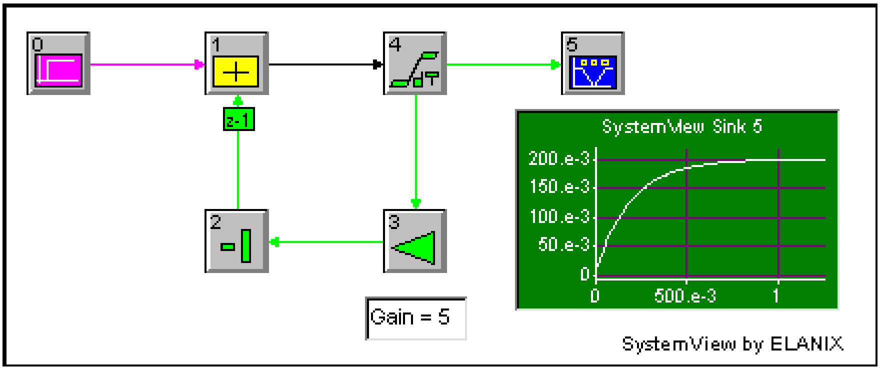
Order	Token	Name	Group	Parameters

0		Time	Aperiodic Sources	Gain =1
				Offset =0

1	X+	Polynomial	Algebraic Functions	X^5 Coeff=0 X^2 Coeff=0 X^4 Coeff=0 X^1 Coeff=1 X^3 Coeff=0 X^0 Coeff=1 e^{-10}
2		X^a	Algebraic Functions	Exponent value = -1
3	X	Multiplier	Multiplier	None
4	X+	Polynomial	Algebraic Functions	X^5 Coeff=0 X^2 Coeff=0 X^4 Coeff=0 X^1 Coeff=1 X^3 Coeff=0 X^0 Coeff=1 e^{-10}
5		Sinusoid	Periodic Sources	(1) Amplitude = 1 (2) Frequency = .150 (Hz) (3) Phase =0
6		System View	Graphic Sinks	None

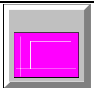
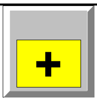
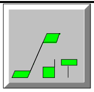
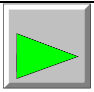
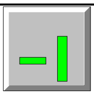
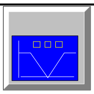
5.11.3 Feedback Example

The following shows an example of a simple feedback control loop using three Operator tokens.



System Time is set to the default values (Sample Rate=100 Hz, Stop Time=1.27 seconds, Number of Samples=128). Input to the system, token zero, is a unit Step Function. The error signal out of token one, the Adder, drives the Integrator in token two. The output of the Integrator is multiplied by five in Gain token three. Finally, the feedback signal is subtracted from the input to complete the loop and produce the error signal.

To make the connection lines point backwards, click the Reverse token direction button, or click the right mouse button on tokens three and four and select Reverse. Note that when executed, a [z], is generated to indicate the presence of an *implicit feedback delay* of one sample (see Chapter 13 for a detailed discussion of implicit delays).

Order	Token	Name	Group	Parameters
0		Step Function	Aperiodic Sources	(1) Amplitude =1 (2) Start time (sec) =0 (3) Bias=0
1		Adder	Adder	None
2		Integrator	Integral/ Diff Operators	Integration Order - Zero Order (2) Initial condition = 0
3		Gain	Gain / Scale Operators	Gain Units - Linear Gain value =5
4		Negate	Gain / Scale Operators	None
5		System View	Graphic Sinks	None

Summary of tokens used in the Feedback example

Chapter 6. Filters and Linear Systems

The Linear Filters / Systems token, is part of the Filters / Systems group of the Operator Library, and is one of the most versatile and powerful SystemView tokens.

This token provides the tools that allow:

Specifying completely arbitrary gain and phase characteristics (e.g., filter synthesis) using the mouse, text entry, or an external file in the Custom filter design window.

- Design of Low Pass, Halfband Low Pass, High Pass, Band Pass, Band Stop, Differentiators, and Hilbert transformers (90 degree phase shift) FIR filters.
- Design of five types of analog (Bessel, Butterworth, Chebyshev, Elliptic and Linear Phase), Low Pass, High Pass, Band Pass and Band Stop filters.
- Defining specialized filters in the Communications group such as Gaussian, Sin(t)/t, Root-Raised Cosine, and Raised Cosine.
- Defining custom linear system transfer functions, in either the continuous Laplace s-domain or in the time discrete z-domain.
- Exporting and importing filter tap values into and out of SystemView designs.

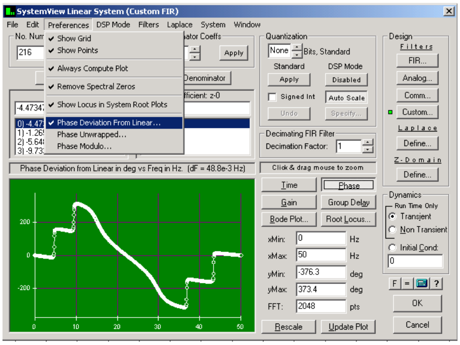
There is a wide range of options for customizing Linear System tokens, including:

- Transient or non-transient behavior at the beginning of the simulation,
- Quantization of filter and linear system coefficients,
- Windowing applied to the filter and linear system coefficients, (Bartlett, Blackman, Hamming, Hanning, Ready, and Kaiser).
- Root locus and Bode plots (frequency, gain and phase) for detailed analysis.

The user has the option to display the phase deviation from linear, the unwrapped phase, or the phase modulo when designing a filter. This allows the user to analyze the corresponding phase of the filter being designed.

6.1 Linear System Token Definition

Enter the SystemView Linear System window by clicking the right mouse button in the design window, and selecting New Filter/Linear System, or launch and double click on a generic token to access the Operator library. Select Linear Sys & Filter, and click the Parameters button, or double-click the Linear Sys token for the window in figure 6.1.


SystemView Linear System (Custom FIR)

No. Num: 216

-4.4734

0) -4.47
1) -1.26
2) -5.64
3) -9.73

Phase Deviation from Linear in deg vs Freq in Hz. (dF = 48.8e-3 Hz)

Quantization: None Bits, Standard

☐ Signed Int

Decimating FIR Filter Decimation Factor: 1

Click & drag mouse to zoom

xMin: 0 Hz
xMax: 50 Hz
yMin: -376.3 deg
yMax: 373.4 deg
FFT: 2048 pts

Dynamics: Run Time Only ● Transient ○ Non Transient

○ Initial Cond: 0

Figure 6.1 The Linear System Design window

The Linear System Window contains a number of tools, including:

- A menu bar at the top of the window that allows access to files, editing functions, preferences, DSP arithmetic mode selection (for the optional DSP Library), filters (FIR, analog, and custom), Laplace system design, Root Locus or Bode plots, all under the System menu, and windowing applications for enhanced filter designs.
- A text box to define the **number of numerator coefficients** of a z-domain H (z) transfer function.
- A text box to define the **number of denominator coefficients** of a z-domain H(z) transfer function.
- A text box for entry or viewing the **numerator and denominator coefficients.** Clicking the mouse on the down arrow in these text boxes will scroll through the list of coefficients.
- A text box where the coefficients can be **quantized** to arbitrary bit numbers.
- A **Transient or Non-Transient** option for specifying the behavior of the linear system at the system Start time. The Transient option assumes the linear system

begins at a zero energy state, while the Non-Transient option pre-loads the linear system using the first input sample value. The input sample value is used as a constant for the time preceding the system Start time, applies the final value theorem.

- A two-dimensional preview **plot area**, where frequency, phase, and group delay response is displayed. The buttons to the right of the plot are used to select Time domain (impulse response), Gain (frequency domain), Phase, and Group Delay plots of the linear system. Click the **right mouse button** to customize the plot and to copy or print the plot. Click and drag the mouse to zoom the plot.
- **Bode Plot and Root Locus** buttons are located to the right of the plot area. Click to view the Bode plot display (frequency and phase on a linear or logarithmic frequency scale) or an interactive root locus, where editing poles and zeros with the mouse can be done. Click and drag the mouse to zoom the plot.

Also, the scale of the plot region can be changed by entering values for x and y Max, and Min, or the FFT size. Clicking the Update button displays these changes. Customize by clicking the right mouse button in the plot, or use Preferences Menu.

Linear System tokens may be defined in several different ways:

- Design a **FIR filter** by clicking the FIR button.
- Design an **analog IIR (continuous) filter** by clicking the Analog button.
- Design a **communications filter** (Gaussian, Raised Cosine, Root-Raised Cosine and Sin(t)/t) by clicking the Comm button.
- Design a **custom filter** (frequency, gain and phase) with the Custom button.
- Specify the system in the **Laplace s-domain** by clicking the Laplace Custom button. SystemView will automatically compute the z-domain coefficients.
- Enter z-domain coefficients $\{a_k, b_k\}$ or define H(z) by clicking **Z-domain**.
- Import the coefficients $\{a_k, b_k\}$ from an external file.

The result of the system specification is a z domain transfer function H(z), of

$$H(z) = \frac{a_0 + a_0 z^{-1} + \ldots a_n z^{-n}}{b_0 + b_1 z^{-1} + \ldots b_m z^{-m}}$$

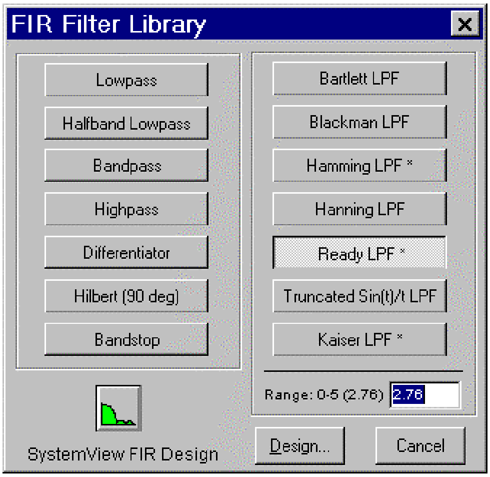
6.2 Finite Impulse Response (FIR) Filter Design

There are four groups of FIR filters supported by SystemView.

- Clicking the FIR button accesses the Standard FIR filter group.
- Clicking on the FIR button also accesses the Window FIR filter group.
- Clicking the Comm button accesses the Communications filter group.
- Clicking on the Custom button also accesses custom filter synthesis.

Group 1, Standard This group has several classes of FIR filters as seen in figure 6.2.

- Low Pass
- Halfband Low Pass
- Band Pass
- High Pass
- Hilbert Transforms
- Differentiators
- Band Reject



Range: 0-5 (2.76) 2.76

Figure 6.2 FIR filter library window

In each case, when the desired type of filter is selected, a graphical design window appears prompting the definition of the pass band, transition band, and stop band of the filters. In addition, prompting for the definition of the pass band ripple in the appropriate filter types. The **Parks-McClellan algorithm** is used to generate the FIR coefficients

Note: *All frequencies are specified as a fraction of the sampling rate seen by the filter.*

For example, if the system Sample Rate is 1 MHz and the design is a 50 KHz FIR lowpass filter, the filter cutoff frequency should be entered as 0.05 (50 KHz / 1 MHz) in the FIR design window. If a Decimator or Sampler token precedes the filter, then the frequencies should be a fraction of the sample rate seen by the filter.

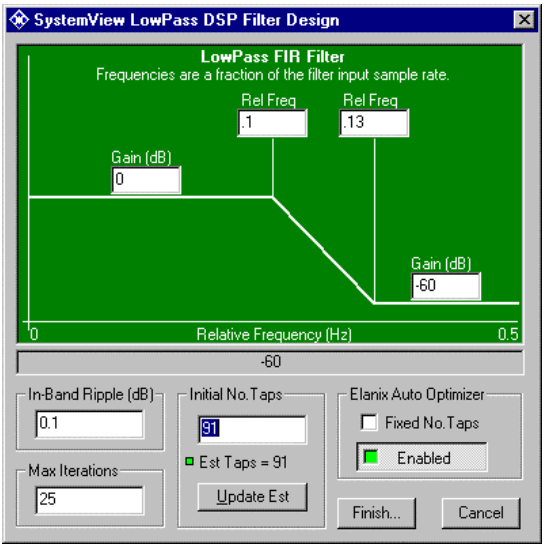
The estimated number of taps required, can be displayed by clicking the Update Tap Estimate button. The Eagleware-Elanix Bit-True Auto-Optimizer provides optimization of FIR designs. Based on the FIR specifications the Auto-Optimizer generates the bit-true tap values, using the required minimum. The user may also fix the number of taps and let the Auto-Optimizer generate the bit-true tap values that correspond to the minimum width of the band transition, that satisfy the specifications.

Upon completion of the calculation, the maximum transition width, ripple, error, and filter coefficients are displayed and the filter response is plotted. The default plot is in the time domain, however, the user can select gain, and phase or group-delay plots as well.

Changing the scale of the plot is possible by modifying the values of xMax, xMin, in addition to the Number of Samples at the bottom of the display. The plot may be zoomed in or out, by clicking the left mouse button and drawing a box around the desired area.

6.2.1 Example LowPass FIR filter

From the Filters button group click FIR, then Low Pass, the following appears:


SystemView LowPass DSP Filter Design
LowPass FIR Filter
Frequencies are a fraction of the filter input sample rate.
0
Relative Frequency (Hz)
0.5
-60
Update Est
Finish...
Cancel

Rel Freq: .1

Rel Freq: .13

Gain (dB): 0

Gain (dB): -60

In-Band Ripple (dB): 0.1

Max Iterations: 25

Initial No. Taps: 91

Est Taps = 91

Elanix Auto Optimizer: □ Fixed No. Taps; 🟩 Enabled

Figure 6.3 The Low Pass filter design window

On the bottom of the window, text boxes are provided to define the number of FIR taps, the In-Band Ripple and the maximum number of iterations of the Park-McClellan algorithm to calculate the FIR tap coefficients. (In most situations, the coefficients will converge to their proper value before the maximum number of iterations is reached.)

The window also contains text boxes to define the gain in the filter pass band, the corner frequencies of the pass band and transition band, and the Gain of the filter in the stop band. To define a low pass filter, enter the following values:

- Corner frequency of the Pass = 0.1 Hz
- Corner frequency of the Stop band = 0.13 Hz
- Stop band Gain = -60 dB

Notice that the No FIR Taps text box is empty at first. Enter any desired number of taps, or click the Update Tap Estimate button. An estimate of the number of taps to fulfill the design requirements will appear in the text box directly above the button. (**Tip:** *Enable the Eagleware-Elanix Auto Optimizer to minimize the number of taps.)*

After entering the design parameters, the coefficients will be calculated by clicking the OK button. During the coefficient computation, a progress bar is displayed to indicate the status (as a percent of Max Iterations) of the coefficient calculations. By default all coefficients are calculated in IEEE double precession. When the coefficient computation is complete, the time domain impulse response of the filter appears in the plot area as shown in figure 6.4. Click the Gain option for the frequency domain plot in figure 6.5.

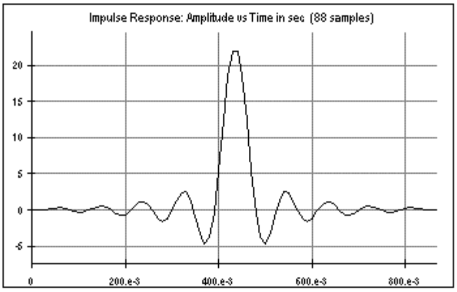


Figure 6.4 FIR filter time (impulse) response

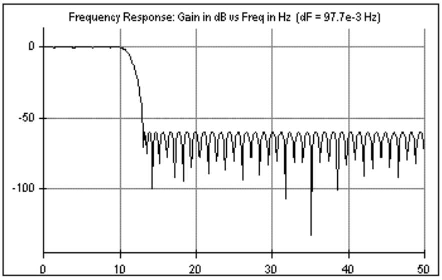
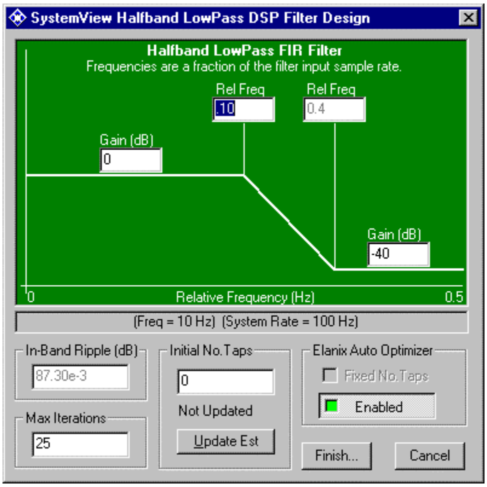


Figure 6.5 FIR filter gain response

Other filter types are designed in a similar fashion using the design windows shown in Figures 6.6 through 6.10. Hilbert Transforms, which introduce a 90 degree phase shift in the input wave form, over a selected bandwidth, (usually that of the signal) are generated in the Hilbert Transform design window, Figure 6.10, and a Differentiator Filter is designed using the window in Figure 6.9.



SystemView Halfband LowPass DSP Filter Design

Halfband LowPass FIR Filter

Frequencies are a fraction of the filter input sample rate.

Rel Freq: .10

Rel Freq: 0.4

Gain (dB): 0

Gain (dB): -40

(Freq = 10 Hz) (System Rate = 100 Hz)

In-Band Ripple (dB): 87.30e-3

Max Iterations: 25

Initial No.Taps: 0

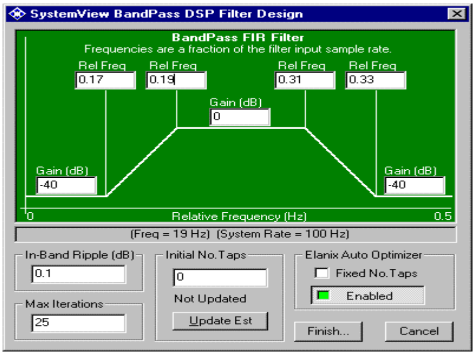
Not Updated

Elanix Auto Optimizer

□ Fixed No.Taps

🟩 Enabled

Figure 6.6. The Halfband LowPass filter design window



SystemView BandPass DSP Filter Design

BandPass FIR Filter

Frequencies are a fraction of the filter input sample rate.

Rel Freq: 0.17

Rel Freq: 0.19

Rel Freq: 0.31

Rel Freq: 0.33

Gain (dB): 0

Gain (dB): -40

Gain (dB): -40

(Freq = 19 Hz) (System Rate = 100 Hz)

In-Band Ripple (dB): 0.1

Max Iterations: 25

Initial No.Taps: 0

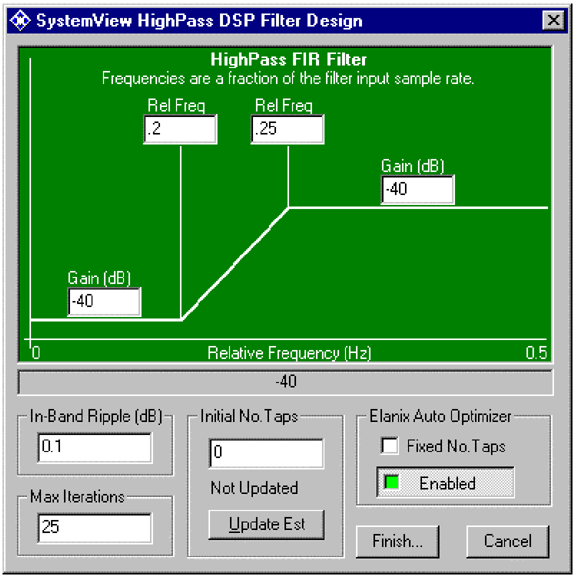
Not Updated

Elanix Auto Optimizer

□ Fixed No.Taps

🟩 Enabled

Figure 6.7. The BandPass filter design window


SystemView HighPass DSP Filter Design

HighPass FIR Filter

Frequencies are a fraction of the filter input sample rate.

Rel Freq: .2

Rel Freq: .25

Gain (dB): -40

Gain (dB): -40

-40

In-Band Ripple (dB): 0.1

Max Iterations: 25

Initial No.Taps: 0

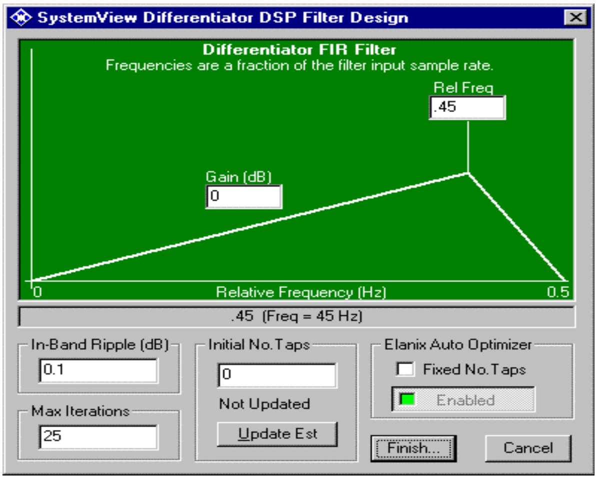
Not Updated

Elanix Auto Optimizer

□ Fixed No.Taps

🟩 Enabled

Figure 6.8. The High Pass filter design window


SystemView Differentiator DSP Filter Design

Differentiator FIR Filter

Frequencies are a fraction of the filter input sample rate.

Rel Freq: .45

Gain (dB): 0

.45 (Freq = 45 Hz)

In-Band Ripple (dB): 0.1

Max Iterations: 25

Initial No.Taps: 0

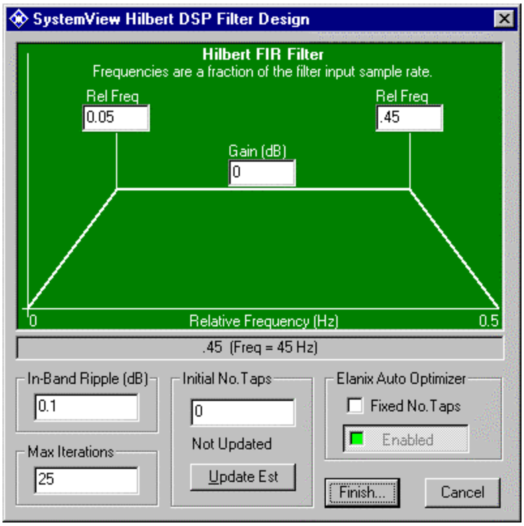
Not Updated

Elanix Auto Optimizer

□ Fixed No.Taps

🟩 Enabled

Figure 6.9. The Differentiator design window


SystemView Hilbert DSP Filter Design
Hilbert FIR Filter
Frequencies are a fraction of the filter input sample rate.
0
Relative Frequency (Hz)
0.5
Elanix Auto Optimizer
Update Est
Finish...
Cancel

Rel Freq: 0.05

Rel Freq: .45

Gain (dB): 0

.45 (Freq = 45 Hz)

In-Band Ripple (dB): 0.1

Max Iterations: 25

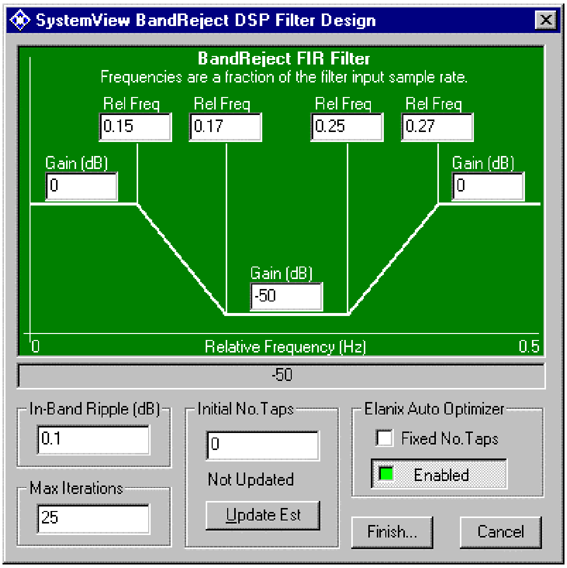
Initial No. Taps: 0

Not Updated

☐ Fixed No. Taps

🟩 Enabled

Figure 6.10. The Hilbert (90 degrees) filter design window


SystemView BandReject DSP Filter Design
BandReject FIR Filter
Frequencies are a fraction of the filter input sample rate.
0
Relative Frequency (Hz)
0.5
Elanix Auto Optimizer
Update Est
Finish...
Cancel

Rel Freq: 0.15

Rel Freq: 0.17

Rel Freq: 0.25

Rel Freq: 0.27

Gain (dB): 0

Gain (dB): 0

Gain (dB): -50

-50

In-Band Ripple (dB): 0.1

Max Iterations: 25

Initial No. Taps: 0

Not Updated

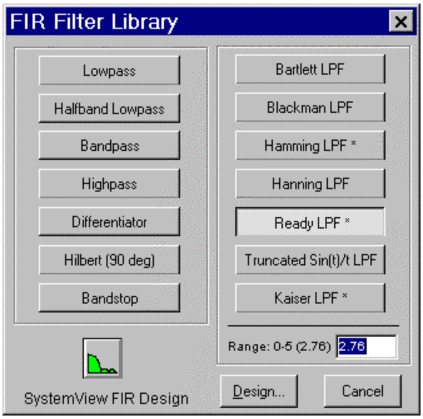
☐ Fixed No. Taps

🟩 Enabled

6.11. The Band Reject filter design window

6.2.2 Group 2, Windows

The second group of FIR filters, are all Low Pass designs based on standard impulse responses associated with common window functions. The following types of Window FIR filters are generated in the Linear System design window:



Range: 0-5 (2.76) 2.76

Bartlett

$$w(n) = 2n/N - 1,\ \ 0 \le n \le (N-1)/2$$

$$= 2 - 2n/(N-1),\ \ (N-1)/2 \le n \le N-1$$

Blackman

$$w(n) = 0.42 - 0.5 \cdot Cos\left(\frac{2\pi n}{N-1}\right) + 0.08 \cdot Cos\left(\frac{4\pi n}{N-1}\right)$$

Hanning

$$w(n) = \frac{1}{2}\left(1 - Cos\left(\frac{2\pi n}{N-1}\right)\right)$$

Hamming

$$w(n) = 0.54 - 0.46 \cdot Cos\left(\frac{2\pi n}{N-1}\right)$$

Ready

$$w(n) = w_H(n) \otimes w_H(n)$$

(Auto-convolution of the Hamming window)

Truncated Sin(t)/t

$$w(n) = \frac{Sin\left(\frac{2\pi n}{N-1}\right)}{\left(\frac{2\pi n}{N-1}\right)}$$

Kaiser

$$w(n) = \frac{I_0\left[\alpha\sqrt{\left(\frac{N-1}{2}\right)^2 - \left[n - \left(\frac{N-1}{2}\right)\right]^2}\right]}{I_0\left[\alpha\left(\frac{N-1}{2}\right)\right]}$$

I_0 Is the modified Bessel function, zero order of the first kind, and α is a parameter adjusting the width of the main-lobe, and the peak of the side lobe amplitude, within the range of (4 < α < 9).

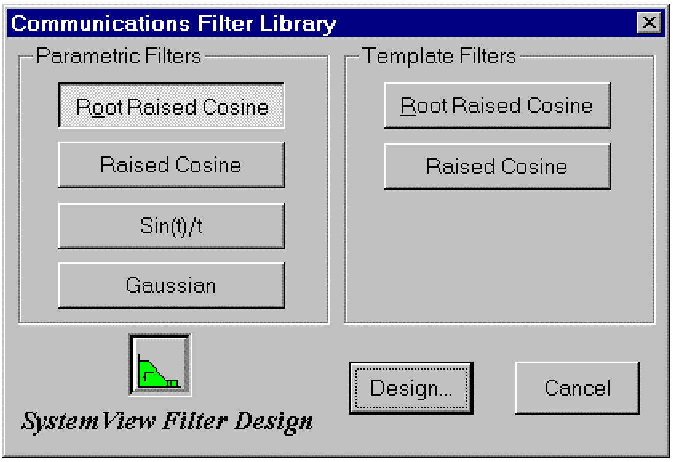
The READY window is the auto-convolution of the Hamming window, and has -80dB sidelobes at the expense of a wider main lobe width.

Begin the design by selecting the desired window filter type. A design window appears showing the general shape of the filter. As with the previous FIR designs, the window contains data input areas that prompt the user for filter parameters. After the data is entered, the number of taps is estimated by clicking the Update Tap Estimate button. In this case, the estimated number is the actual number used, and no other choice is available. Clicking OK generates the filter coefficients.

6.2.3 Group 3, Comm.

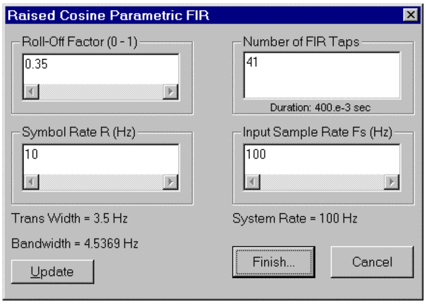
The third group of FIR filters are all Low Pass designs often used in communication systems.

- Raised Cosine (Template)
- Raised Cosine (Parametric)
- Root Raised Cosine (Template Mode)
- Root Raised Cosine (Parametric)
- Truncated Sin(t)/t (Parametric)
- Gaussian (Parametric)



Template Mode refers to the use of a Low Pass template similar to that used for Low Pass FIR design. Select this option if you want to specify the out-of-band rejection level. Parametric refers to the use of a closed form (algebraic) expression to generate the filter impulse response. Select this option for an “exact” pass band and transition band.

Raised Cosine and Root-Raised Cosine filters


Raised Cosine Parametric FIR

Roll-Off Factor (0 - 1): 0.35

Number of FIR Taps: 41

Duration: 400.e-3 sec

Symbol Rate R (Hz): 10

Input Sample Rate Fs (Hz): 100

Trans Width = 3.5 Hz

System Rate = 100 Hz

Bandwidth = 4.5369 Hz

Figure 6.12 The Root-Raised cosine filter parametric specification window

When specifying a cosine filter in the parametric mode, the following definitions apply (see Figure 6.12):

$$
\begin{aligned}
\left|H(f)\right| &= 1, && \left|f\right| \le \frac{R}{2}(1-\beta) \\
&= \cos^{p} \frac{\pi}{2\beta R}\left(\left|f\right| - \frac{R}{2}(1-\beta)\right), && \frac{R}{2}(1-\beta) < \left|f\right| \le \frac{R}{2}(1+\beta) \\
&= 0, && \left|f\right| > \frac{R}{2}(1+\beta)
\end{aligned}
$$

Where $|H(f)|$ is the magnitude of the ideal frequency response of the filter, (p =1 for the raised cosine filter and p=0.5 for the root-raised cosine filter). β is the Roll-Off Factor and R is the symbol rate of the signal into the filter.

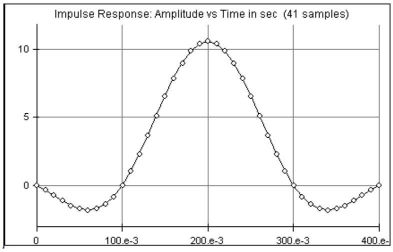
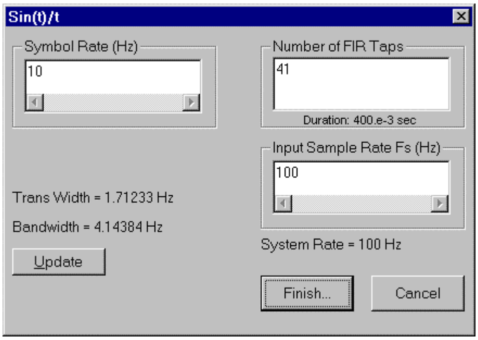


Figure 6.13 cosine filter impulse response with values in Figure 6.12

Truncated Sin(t)/t filters.



Sin(t)/t

Symbol Rate (Hz): 10

Number of FIR Taps: 41

Duration: 400.e-3 sec

Input Sample Rate Fs (Hz): 100

Trans Width = 1.71233 Hz

Bandwidth = 4.14384 Hz

System Rate = 100 Hz

Figure 6.14 The *Sin*(*t*)/*t* parametric design window

The impulse response of the truncated *Sin*(*t*)/*t* filter is defined as follows,

$$h(t) = G \cdot \left(\frac{Sin(\pi Rt)}{\pi Rt} \right)$$

Where *R* is the specified Symbol Rate and *G* is an automatically computed gain factor used to enforce 0 dB gain in the Pass Band.

Because this filter is implemented in SystemView as an FIR, the sampled data version of the above impulse response becomes,

$$h(n) = G \cdot \left(\frac{Sin\left(\pi\, n \frac{R}{Fs} \right)}{\left(\pi\, n \frac{R}{Fs} \right)} \right), \quad n = 0,1, \ldots N-1$$

Where *Fs* is the sample rate as seen by the token and *N* is the specified Number of FIR Taps. The duration of the impulse response is then $(N-1)/Fs$ seconds. **When the input rate to this filter is a consequence of a System rate change,** SystemView will automatically re-compute the filter (in the background).

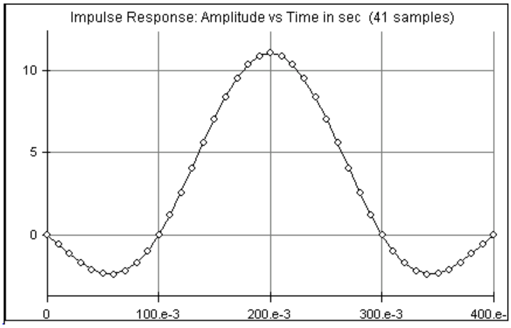
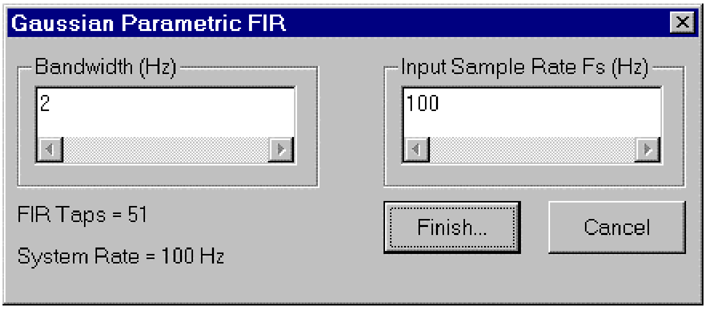


Figure 6.15 Example of *Sin*(*t*)/*t* impulse response using the values specified in Figure 6.14

Gaussian Filters



Bandwidth (Hz): 2

Input Sample Rate Fs (Hz): 100

FIR Taps = 51

System Rate = 100 Hz

Figure 6.16 The Gaussian filter parametric design window.

The Gaussian filter has a Gaussian time and frequency response. The frequency response is given by,

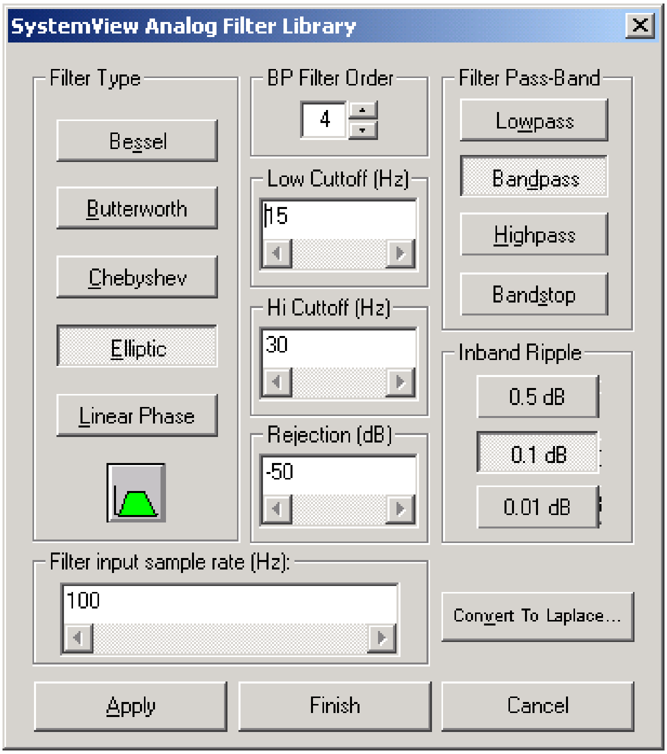
$$H(f) = e^{-\left(\frac{f}{B}\right)^{2} \cdot \left(\frac{\ln(2)}{2}\right)}$$

Where *B* is the specified Bandwidth in Hz. SystemView automatically computes the number of FIR taps required, and will automatically update the FIR tap values, if the input rate to the filter changes.

6.2.4 Group 4, Custom

The final group of FIR filters, are custom designs with arbitrary gain and/or phase characteristics that can: 1) be defined using the mouse, 2) defined by entering gain, phase, and frequency values in the table, 3) defined from a text file, or 4) any combination of the above.

The file template used in the example is shown below, and contains samples obtained through the Analysis window and Analog Filter design of a four-pole elliptic bandpass filter. The SystemView Analog Filter Library window below shows the configuration of the elliptic filter. The filter was driven with an impulse and the sink calculator of the analysis window computed the gain and phase response. The gain and phase were combined on one window and exported as a text file.



BP Filter Order: 4

Low Cuttoff (Hz): 15

Hi Cuttoff (Hz): 30

Rejection (dB): -50

Filter input sample rate (Hz): 100

To import a custom filter response template from a file click the file menu and select Open Filter Template. The file must be ASCII text, and must have the following format for N entries:

Freq value 1	Tab or space	Gain value 1	Tab or space	Phase value 1
Freq value 2	Tab or space	Gain value 2	Tab or space	Phase value 2
...	...	...	...	...
Freq Value N	Tab or space	Gain value N	Tab or space	Phase value N

Phase value is optional

The following format may also be used

Freq value 1	Tab or space	Gain value 1
Freq value 2	Tab or space	Gain value 2
...	...	...
Freq Value N	Tab or space	Gain value N

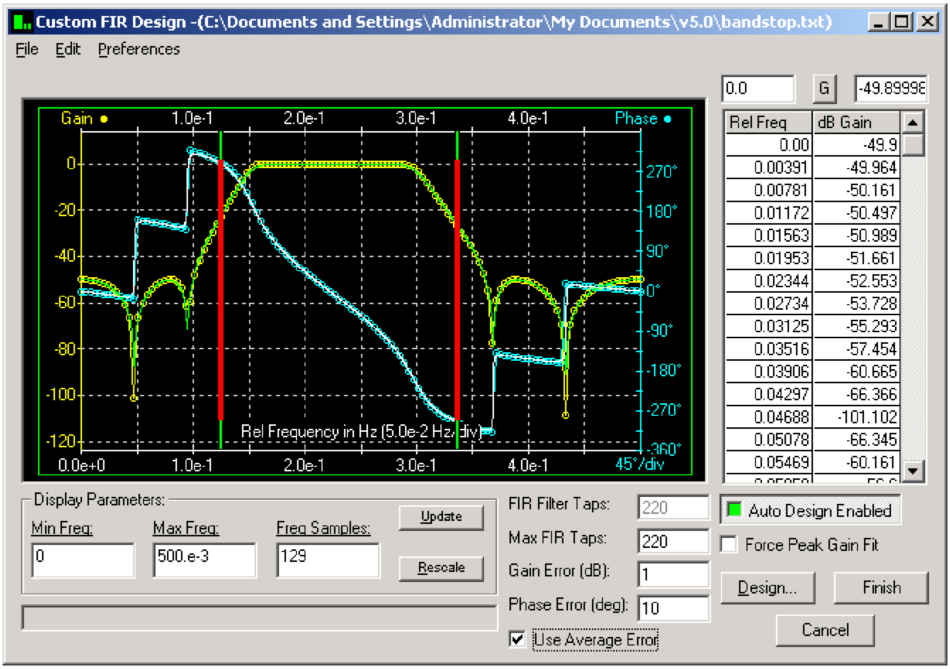
Freq value 1	Tab or space	Phase value 1
Freq value 2	Tab or space	Phase value 2
...	...	...
Freq Value N	Tab or space	Phase value N

Note: phase value is optional

Once the gain and phase values are input to the Custom FIR Filter Design window, the graphic display represents the gain and phase response of the FIR filter. The user may modify the samples within the display area to design a filter. Just drag either a gain or phase sample point and move it to the desired location. It may be zoomed in on, for any region to perform a more detailed design. Use the rescale button to display the full dynamic range. The user may add as many samples to graphically generate the filter. The number of samples is not associated with the number of FIR taps. Therefore, the number of graphical samples may exceed the number of FIR Filter Taps.

The Custom Filter Design as shown in Figure 6.17, allows user flexibility of modeling filters within defined tolerances. The user has control over the maximum number of FIR taps, the maximum acceptable gain error, within the red bands, and the maximum acceptable phase error within the red bands. These error values can be either average or maximum levels.

Red bars within the graphic display isolate the optimize region. Acceptable values of gain and phase error are set by the user. These values may be entered as either absolute error or average error. Select the check box to accept the tolerance values as average error within the error computation region. Note the gain error is set to 1dB and phase error of 10 degrees in the figure below. The average error box is checked; therefore, the optimization will provide an average error of 1 dB for gain and 10 degrees for phase within the selected computation region. The design option will optimize the filter design to use up to the maximum number of FIR taps to attain a gain and phase tolerance within the designated region.



0.0 | G | -49.89998

Rel Freq	dB Gain
0.00	-49.9
0.00391	-49.964
0.00781	-50.161
0.01172	-50.497
0.01563	-50.989
0.01953	-51.661
0.02344	-52.553
0.02734	-53.728
0.03125	-55.293
0.03516	-57.454
0.03906	-60.665
0.04297	-66.366
0.04688	-101.102
0.05078	-66.345
0.05469	-60.161

Display Parameters:

Min Freq: 0

Max Freq: 500.e-3

Freq Samples: 129

FIR Filter Taps: 220

Max FIR Taps: 220

Gain Error (dB): 1

Phase Error (deg): 10

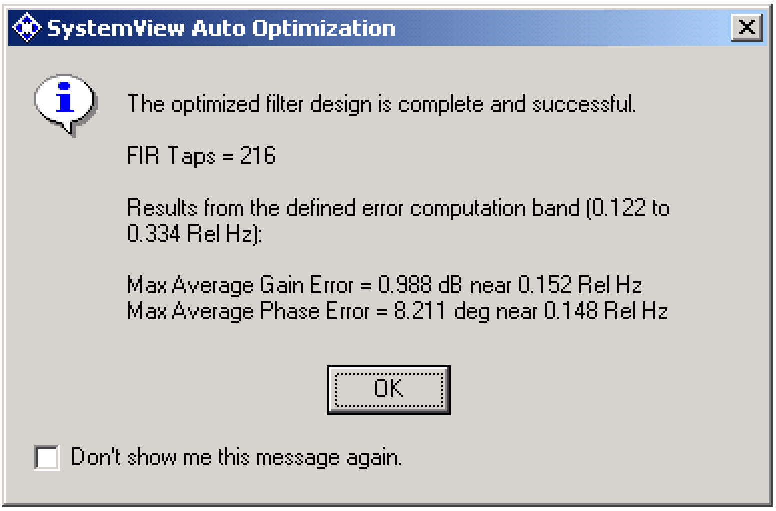
☑ Use Average Error

🟩 Auto Design Enabled

☐ Force Peak Gain Fit

Figure 6.17 The Custom filter design window

The SystemView Auto Optimization window below indicates that 216 FIR taps are required to build the desired filter. The maximum average gain error within the optimized region is 0.988 dB at 0.152 relative Hz. The maximum average phase error is 8.211 degrees at 0.148 relative Hz.



SystemView Auto Optimization

The optimized filter design is complete and successful.

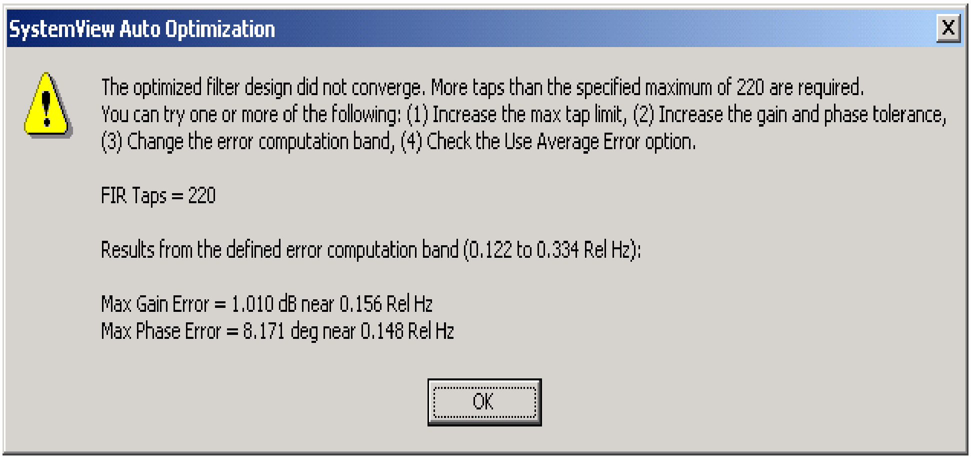
FIR Taps = 216

Results from the defined error computation band (0.122 to 0.334 Rel Hz):

Max Average Gain Error = 0.988 dB near 0.152 Rel Hz
Max Average Phase Error = 8.211 deg near 0.148 Rel Hz

□ Don't show me this message again.

If the Use Average Error option has not been selected, the filter would not have converged using these tolerances. The figure below indicates the best optimization that could be achieved within the user specified constraints. SystemView provides four suggestions to update the design. 1) Increase the maximum number of taps, 2) Increase the gain and phase tolerance values, 3) Change the error computation region, or 4) Check the Use Average Error option.



The optimized filter design did not converge. More taps than the specified maximum of 220 are required.
You can try one or more of the following: (1) Increase the max tap limit, (2) Increase the gain and phase tolerance, (3) Change the error computation band, (4) Check the Use Average Error option.

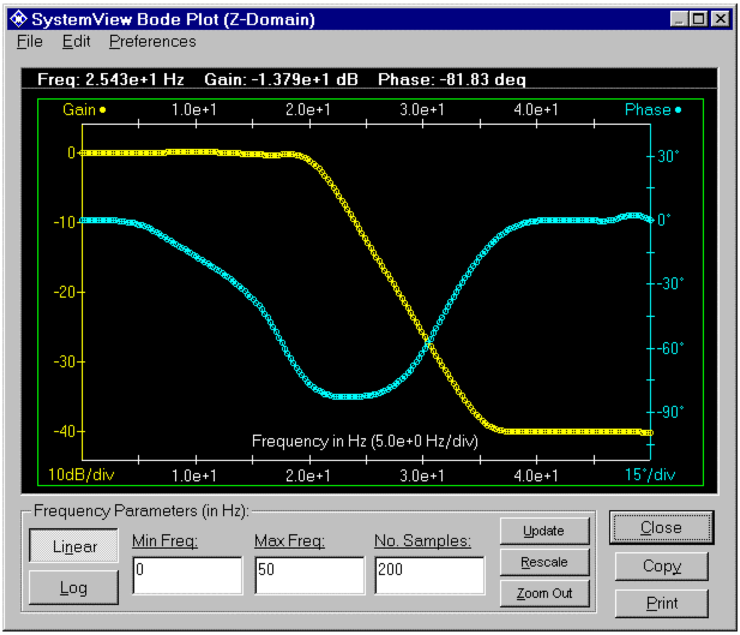
FIR Taps = 220

Results from the defined error computation band (0.122 to 0.334 Rel Hz):

Max Gain Error = 1.010 dB near 0.156 Rel Hz
Max Phase Error = 8.171 deg near 0.148 Rel Hz

When finished with the design, click the OK button and SystemView will create a FIR filter having a custom gain response. Increasing the number of taps in the FIR Filter Taps box will generally lead to a more accurate filter response.

A typical custom FIR gain and phase response of a filter design is shown in figure 6.18, and indicates the results when the Bode Plot button is clicked in the Linear System window. Figure 6.19 shows the corresponding impulse response as copied from the Linear System window display (click the right mouse button within the plot for a selection of options).



SystemView Bode Plot (Z-Domain)

Freq: 2.543e+1 Hz   Gain: -1.379e+1 dB   Phase: -81.83 deg

Frequency Parameters (in Hz):

Min Freq: 0

Max Freq: 50

No. Samples: 200

Figure 6.18 Bode plot for the custom filter design

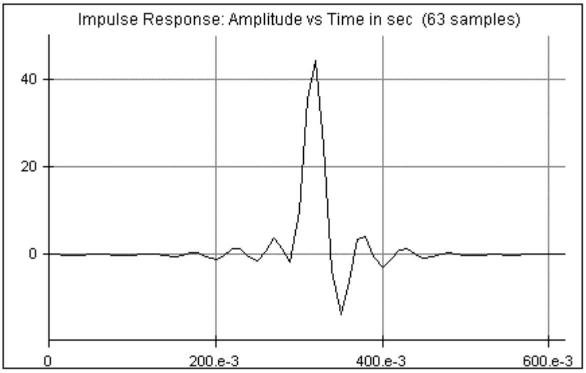


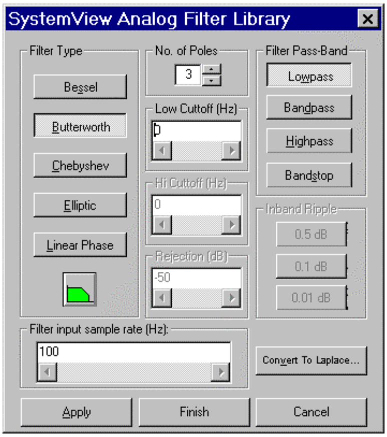
Figure 6.19 Impulse response for the custom filter design

6.3 Analog (Continuous) Filter Design

Five classes of analog filters can be designed using this selection menu, including:

Butterworth Bessel Chebyshev Elliptic linear Phase

These can be Low Pass, High Pass, Band Pass, or Band Reject filters. Click the Analog button in the Filter Design group to open the window shown in Figure 6.20.



No. of Poles: 3

Low Cuttoff (Hz): 0

Hi Cuttoff (Hz): 0

Rejection (dB): -50

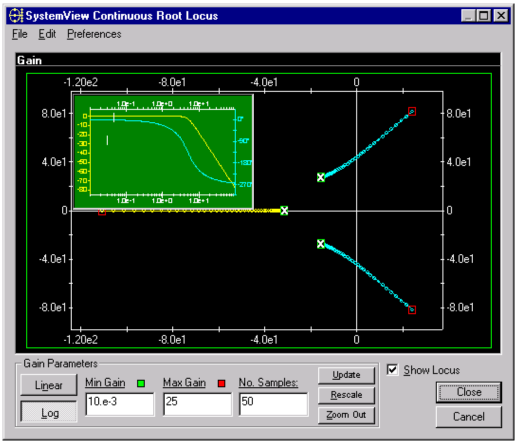
Filter input sample rate (Hz): 100

Figure 6.20 The analog Filter design window

Ultimately the *H(s)* of the analog filter that is created, will be simulated in the *H(z)* domain. The Analog filter is converted to the *H(z)* domain by using the bi-linear transformation, and the pre-warping is automatic.

The type determines the general shape of the selected filter. The data required is the order of the filter (i.e., the number of poles), the 3 dB cutoff frequencies in Hertz, and when appropriate, the pass band or phase ripples in dB.

The button labeled Convert To Laplace, automatically converts the current filter design to a Laplace representation *H*(*s*), and opens the Laplace Design window. The Laplace representation may be edited to view/edit the poles and zeros of individual sections or all sections of the filter. It is possible to move poles and zeroes while simultaneously monitoring the filter bode plot. To do this, first create an analog filter, then click on the Convert To Laplace tab, and select Root-Locus. Now select a pole, or a zero, with the mouse and move it.



Min Gain 🟩 10.e-3

Max Gain 🟥 25

No. Samples: 50

☑ Show Locus

Figure 6.21 Simultaneous View of Bode Plot and Root Locus

To design a Chebyshev low pass filter, define the system Sample Rate to 1 MHz and Click the Filters menu then select Analog. To define the filter, click Chebyshev, and Bandpass. Now choose seven poles (this filter type can have a maximum of nine), an in band ripple of 0.1 dB, and lower and upper cutoff frequencies of 10 KHz and 200 KHz.

When these entries are made, click OK and the impulse response of the filter will be plotted. The gain of the filter versus frequency, (relative to the system sample rate), is plotted by selecting the Gain button. Click the Bode Plot button for an interactive display of gain and phase as shown in Figure 6.22.

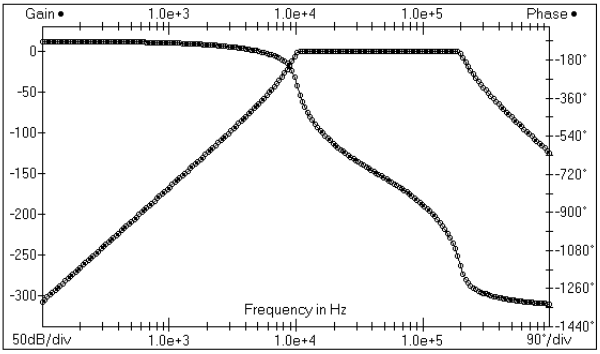


Figure 6.22 Seven-pole Chebyshev Band Pass filter gain and phase response

6.4 Laplace Systems

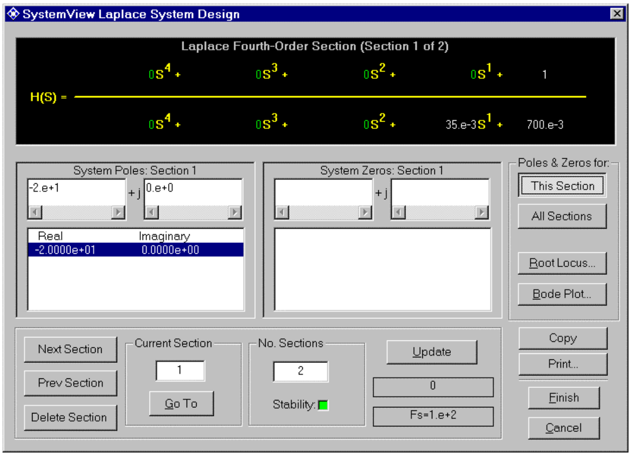
SystemView provides the capability to directly implement a continuous linear system in a single token, if SystemView has its Laplace transform. The implementation architecture consists of breaking the system into a product of fourth-order sections:

$$H(s) = \prod_{k=1}^{n} H_k(s)$$

$$H_k(s) = \frac{a_{4,k}s^4 + a_{3,k}{}^3 + a_{2,k}s^2 + a_{1,k}s + a_{o,k}}{b_{4,k}s^4 + b_{3,k}{}^3 + b_{2,k}s^2 + b_{1,k}s + b_{0,k}}$$

Each fourth-order section is automatically transformed into the z-domain by using the bilinear z transformation, $s = 2f_s \dfrac{1-z^{-1}}{1+z^{-1}}$ Where f_s is the sample rate, as seen by the token. SystemView automatically senses the presence of decimators and/or sampler tokens preceding this token, and adjusts the z-domain coefficients accordingly.

The Laplace System design window, shown in Figure 6.23, appears when the Laplace menu is selected. Figure 6.24 shows the first of two, fourth-order sections corresponding to the Laplace system discussed in the following example:


SystemView Laplace System Design

Laplace Fourth-Order Section (Section 1 of 2)

$$H(S) = \frac{0S^4 + 0S^3 + 0S^2 + 0S^1 + 1}{0S^4 + 0S^3 + 0S^2 + 35.e\text{-}3S^1 + 700.e\text{-}3}$$

System Poles: Section 1: -2.e+1 + j 0.e+0

Real	Imaginary
-2.0000e+01	0.0000e+00

System Zeros: Section 1: + j

Current Section: 1

No. Sections: 2

Stability: 🟩

0

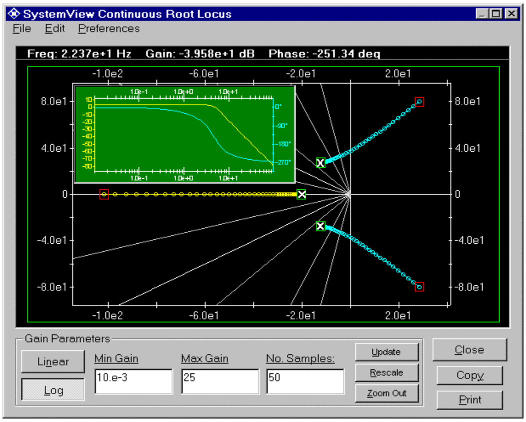
Fs=1.e+2

Figure 6.23 The Laplace System design window

For example, to implement a linear system having two sections,

$$H(s) = \left(\frac{0.6}{0.03s + 0.7}\right) \cdot \left(\frac{1}{0.001s^2 + 0.025s + 0.9}\right)$$

- Clear the system. Define the system time with a Sample Rate of 100 and Number of Samples = 128 (i.e., the default time values).
- Launch an Operator token. Double click the Operator token and then double click the Linear System token in the library. Click the Custom Laplace button to open the Laplace system design window.
- Enter “2” in the Number of Sections window. The Section Number window will initially read “1”.
- Enter the coefficients for the first section in each s-coefficient text box.
- Click the Next Section button and the Section Number changes to “2”.
- Enter the coefficients for the second section.
- Click the Update Roots button to see the system poles and zeros.
- Click the Root Locus button to see the plot shown in Figure 6.24. Press the Ctrl key and click/drag the mouse to zoom any portion of the display.
- Placing the mouse over a pole changes the pointer to indicate the capability to move the pole to a new location to see the resulting gain and frequency response.


SystemView Continuous Root Locus

Freq: 2.237e+1 Hz Gain: -3.958e+1 dB Phase: -251.34 deq

Gain Parameters

Min Gain: 10.e-3

Max Gain: 25

No. Samples: 50

Figure 6.24 Root Locus plot (and Quick-View Bode plot) for the third-order example

- Click the Bode Plot button to see the plot in Figure 6.25. Press the Ctrl key and click/drag the mouse to zoom any portion of the display. Click the Close button.

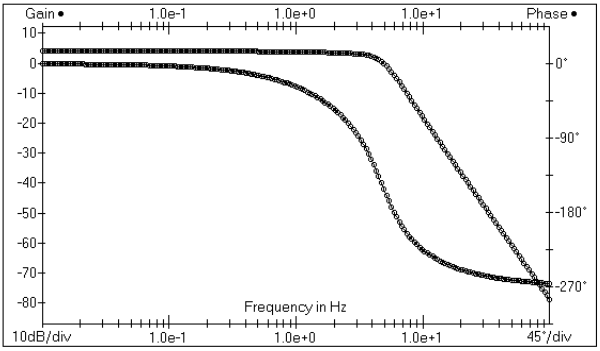


Figure 6.25 Bode plot for the third-order example

- Click the OK button in the Laplace window. Note that the z-domain coefficients of this system are now in the Numerator Coeffs and Denominator Coeffs pull-down windows, respectively.

View the time and frequency domain characteristics (Figure 6.26) of this system by clicking the appropriate selections located to the right of the display.

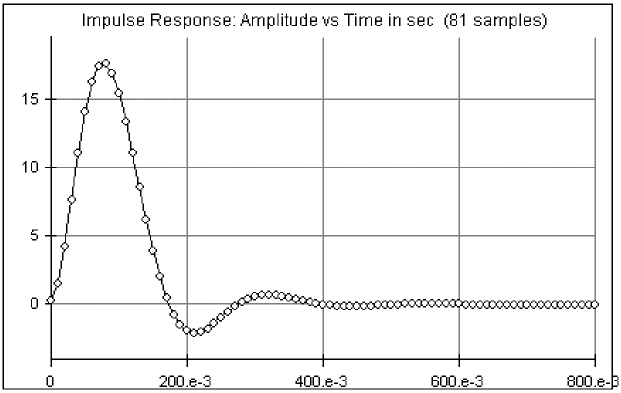
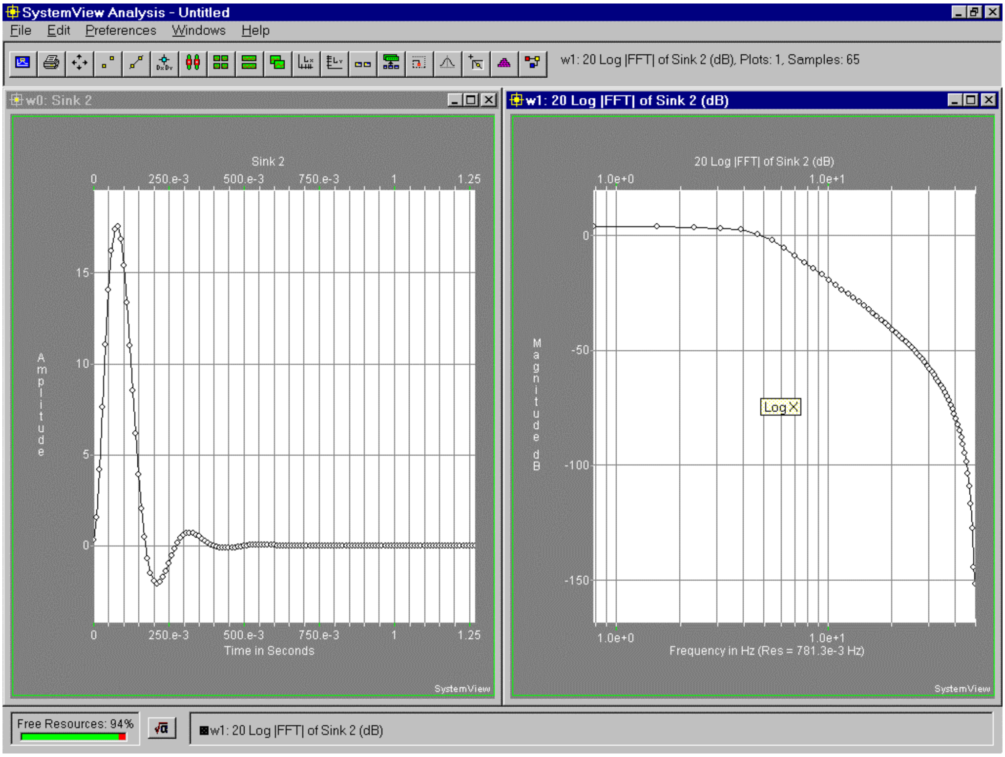


Figure 6.26 Time domain in Linear System window for third-order example

•Click OK in the Linear System window to return to the System window.

Finish the example by connecting an Impulse Source (use the default parameters) to the Laplace token, and connect the output of the token to a Sink token.

Execute the system and view your results in the Analysis window. Click the Sink Calculator, then click the Spectrum tab and select the 20 Log(|FFT|) option and verify the time and frequency response of the system. For a logarithmic frequency, select the spectral plot window and click the LogX button on the toolbar. The results are shown in the Analysis window (Figure 6.27).



w1: 20 Log |FFT| of Sink 2 (dB), Plots: 1, Samples: 65

Free Resources: 94%

w1: 20 Log |FFT| of Sink 2 (dB)

6.27 Simulation results in Analysis window for the third-order example

6.5 Manual Entry of Coefficients

It is possible for the user to enter as many coefficients as needed, into both the numerator *N*(*z*) and the denominator, *D*(*z*) of the transfer function using:

$$H(z) = \frac{N(z)}{D(z)}$$

To manually enter coefficients in the numerator, or denominator, select the text box that defines the number of coefficients and enter the desired number. Each successive coefficient is added using the down arrow key or pressing Enter.

After entry of the coefficients is complete, the impulse response of the transfer function can be plotted versus normalized time. This is done by selecting the Time button, this is the default plot, or the versus frequency plot, by selecting Gain. Clicking the appropriate button can plot phase response and the Group Delay. System coefficients may be saved by selecting the Save Coefficient File item from the File menu at the top of the Linear System design window.

Example Custom Transfer Function:

$$H(z) = \frac{0.019 + 0.039 \cdot z^{-1} + 0.02 \cdot z^{-2}}{1 - 1.58 \cdot z^{-1} + 0.65 \cdot z^{-2}}$$

Enter the number of Numerator coefficients in the No. of Numerator Coeffs text box (three in this example), and enter 3 in the No. of Denominator Coeffs text box.

Enter the following set of coefficients in the numerator: 0.019, 0.039, and 0.02. In the denominator, enter 1.0, -1.58, and 0.65, and click on the Update button to the right of the display. The impulse response will appear in the plot area as shown in Figure 6.28.

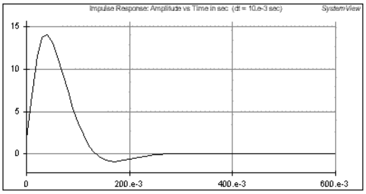


Figure 6.28 Impulse response

The user can examine the frequency response by selecting the Gain button. It should appear as shown in the frequency domain plot in Figure 6.29. You may also use the Z-domain tab in the Linear System Window to enter *H(z)*.

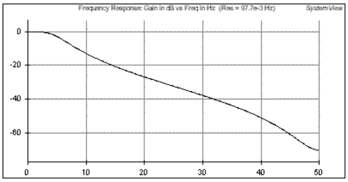


Figure 6.29 Frequency response

Custom IIR Example: Simpson's rule of numerical integration

As a second example of the custom IIR capability, consider the expression: $I = \Delta \cdot \left(y_1 + 4y_2 + y_3 \right) / 3$, which approximates to order Δ^5, the integral of a function y using three points separated by step size Δ. This relation, when extended to include more than three points, becomes: $I_n = I_{n-2} + \left(y_n + 4y_{n-1} + y_{n-2} \right) \cdot \Delta / 3$, taking the z transform of the relation yields the transform of the integrator as (ignoring the scale factor Δ/3):

$$H(z) = \frac{1 + 4z^{-1} + z^{-2}}{1 - z^{-2}}$$

Set up a linear IIR system as described above.

- No. Numerator Coeffs: 3
- No. Denominator Coeffs: 3
- Coefficients: 1, 4, 1 (Numerator)
- Coefficients: 1, 0, -1 (Denominator)

Normalize using a Gain token with a value of Time Step/3. Drive this system with any input source. Verify that the output is the integral of the input. This idea can be extended indefinitely to even higher orders of integration.

6.6 Importing System Coefficients for Custom Design

Linear System coefficients (numerator and denominator) can be imported from an external file (one in which the filter coefficients have been calculated previously by some other means). To properly import the coefficients, format the file as follows:

- The coefficient data must be in either text format (ASCII), or in text format with embedded binary single precision IEEE floating-point values.
- The data for the numerator is first. This data must have a header defining the number of coefficients in the numerator, directly followed by the coefficients.
- The column continues without a break by defining the number of coefficients in the denominator, directly followed by the denominator coefficients.

Consider the following text coefficient file (the "=" character is the only required identifier in the header of each list),

```
Numerator=3
0.019
0.039
0.019
Denominator=3
1.0
-1.58
0.66
```

This file defines the same transfer function that was defined in section 6.5. Coefficient files must also be imported using this format. To import coefficients from a file, select the File menu in the Linear System design window, then select Open Coefficient File from the menu. A window appears, allowing selection of the desired file.

6.7 Root Locus and Bode Plots in the Linear System Design Window

SystemView will compute interactive Root Locus and Bode plots for a linear system, when the Root Locus or Bode Plot button is activated. The Root Locus is the locus of the poles (as a function of the loop gain, *k*) of the closed-loop feedback system, whose open-loop transfer function *H(s)* is the linear system specified in the Linear System design window. Thus it is a plot of the poles of:

$$\frac{H(s)}{1+kH(s)}$$

Or equivalently, the roots of:

$$D(s)+kN(s)=0$$

Where the formula for the linear system that has been defined is:

$$H(s)=\frac{N(s)}{D(s)}$$

If the system is defined in the z- domain, then the above definitions are in terms of *H*(*z*).

The Bode plot is a plot with the magnitude and phase of *H*(*s)* as a function of frequency *f* in Hz (with $s = j2\pi f$).

While in either the Root Locus or Bode Plot window, zoom the display by pressing the Ctrl key and click/drag the mouse to outline the area to be zoomed. The number of samples, the gain or frequency range, and linear or log (gain or frequency) spacing may be specified. See chapter 15 for additional information on Root Locus and Bode Plots,.

6.8 Windowing

In some designs it is desirable to window existing filter coefficients. Several window types are available: Hamming, Hanning, Bartlett, Blackman, Ready, and the Kaiser window.

These windows are applied after filter definition by selecting the desired window type from the Window menu. Upon selection, the window is immediately applied to the numerator coefficients. The results are plotted as the selected time, gain, and phase or group delay function. The application of the window can be removed by selecting Undo Apply.

As an example, apply the Hamming window to the Low Pass filter designed earlier in this chapter. To do this, select Hamming from the pull-down menu under Windows. The window is immediately applied to the coefficients of the filter and the impulse response plotted. If selecting Gain, notice that the filter ripple in the pass band has been significantly attenuated. To remove the influence of the window, select Undo Apply.

6.9 Coefficient Quantization

When implementing a digital filter an important issue arises regarding the number of binary bits (Quantization levels = 2^bits) required representing the otherwise "continuous" (IEEE double precision, 64 bits) filter coefficients, since filter performance may be critically dependent on coefficient quantization.

SystemView has the capability to help make this determination. In the Linear System design window there is a text box labeled Quantization Bits. The default is "None". In this case, all filter-design algorithms produce coefficients in full precision. To see the effects of quantizing a filter to eight bits, go the text area, enter "8", and click the Apply button. The numerator and denominator coefficients are automatically quantized to eight bits, and change accordingly. The plot will change to show the new configuration filter performance. To return to the original non-quantized condition, click the Undo button.

6.10 Summary Of Filter Tokens In The Linear System Window

	Analog Band Pass	Continuous Band Pass linear filter, Class can be Butterworth, Bessel, Chebyshev, Elliptic, or Linear Phase
	Analog Band Reject	Continuous Band Reject (notch) linear filter, Class same as above.
	Analog High Pass	Continuous High Pass linear filter, Class same as above.
	Analog Low Pass	Continuous Low Pass linear filter, Class same as above.
G Φ	Custom Filter	Custom FIR (gain and phase) linear filter
H[S]	Custom Laplace	Custom filter derived from Laplace transformation H(s)
	FIR Band Reject	Band Reject FIR filter
	FIR Band Pass	Band Pass FIR filter
	FIR Differentiator	Differentiator FIR filter
H	FIR Halfband	Halfband FIR filter
	FIR High Pass	High Pass FIR filter
90	FIR Hilbert Transform	Hilbert (90 degree phase shift) FIR filter
	FIR Low Pass	Low Pass FIR filter
G	Gaussian Filter	Gaussian FIR filter
Hz	Custom Linear Systems	Custom z-domain linear system with file or manual entry coefficients.
	Raised Cosine	Raised Cosine linear FIR filter (template or parametric)
r	Root Raised Cosine	Raised Root Cosine linear FIR filter (template or parametric)
T	Sin(t)/t	Truncated Sin(t)/t linear FIR filter (template or parametric)
	Window Filter	Low Pass linear FIR filter: Bartlett, Blackman, Hamming, Hanning, Ready, Kaiser, or Truncated Sin(t)/t

Chapter 7. MetaSystems

MetaSystems are SystemView's mechanism for incorporating hierarchy into a design. MetaSystems are single tokens containing entire subsystems that can easily be created in SystemView. MetaSystems may contain other MetaSystems, providing multiple levels of hierarchy.

Important uses of MetaSystems are:

- To build a library of custom subsystems,
- To simplify the visual presentation of a system by reducing logical groups of tokens into a single token,
- To allow the use of the same MetaSystem in different systems.
- To support back-to-back simulation of different design options.

When MetaSystems are used effectively, a system becomes less complex to view and easier to understand. MetaSystems help in building large simulations consisting of hundreds of individual tokens that would otherwise be difficult to manage.

7.1 Creating a MetaSystem

From a SystemView Model Using the Mouse: A MetaSystem can be created automatically from a group of tokens by clicking the Create MetaSystem button, then click/drag the mouse to outline the tokens to be included in the new MetaSystem. Release the mouse button, and SystemView will automatically insert MetaSystem I/O tokens where needed.

From a File: Launching the generic MetaSystem from the token Reservoir can create a MetaSystem. Double click on the new token, and the file dialog box will appear. A system file (.svu) or MetaSystem file (.mta) may be selected.

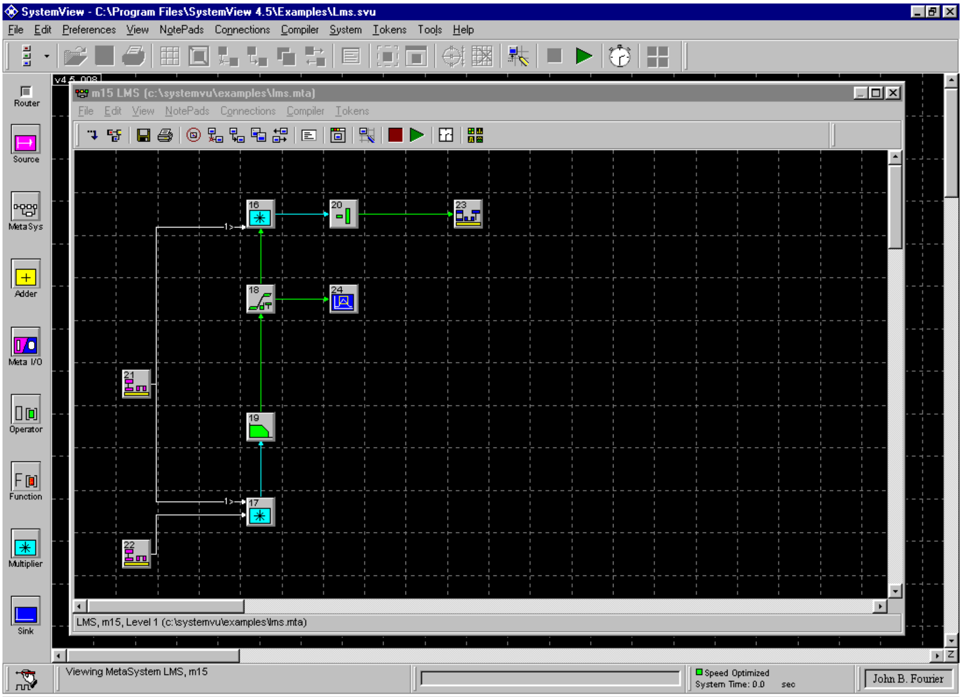
Duplicating: To duplicate a MetaSystem token, click the duplicate button then the token to be duplicated. To duplicate a group of MetaSystem tokens, click the duplicate button and drag the mouse pointer to outline the group to be duplicated. Note that for group duplications, all connecting paths within the group are preserved.

7.2 Viewing a MetaSystem

MetaSystems are viewed in the MetaSystem Window shown in Figure 7.1. The MetaSystem Window can be entered using any of the following:

- Click the View MetaSystem button then click the token to be viewed.
- Right Mouse button click on the MetaSystem and select view MetaSystem.
- Use the **View** Menu with the MetaSystem option.
- Place the dynamic system probe on the MetaSystem,
- Place the mouse arrow over the MetaSystem and double click.

MetaSystems files can also be viewed and edited as stand-alone system view files.


SystemView - C:\Program Files\SystemView 4.5\Examples\Lms.svu
File Edit Preferences View NotePads Connections Compiler System Tokens Tools Help
v4.5.008
m15 LMS (c:\systemvu\examples\lms.mta)
File Edit View NotePads Connections Compiler Tokens
Router
Source
MetaSys
Adder
Meta I/O
Operator
Function
Multiplier
Sink
16
20
23
18
24
21
19
17
22
Speed Optimized
System Time: 0.0 sec
John B. Fourier

LMS, m15, Level 1 (c:\systemvu\examples\lms.mta)

Viewing MetaSystem LMS, m15

Figure 7.1 Viewing the MetaSystem via the MetaSystem Window

7.3 Saving a MetaSystem

MetaSystems are automatically saved with the parent SystemView system file (svu), and may also be saved in a separate file. Any file extension may be used, but MTA is recommended, use the following to save a system.

- From the menu select *Save MetaSystem*, and then click on the token to be saved.
- From the MetaSystem File menu, select *Save System*. The Save dialog box appears.

7.4 Integrate and Dump Example MetaSystem

The finite time integrator, or "integrate and dump" filter, is defined mathematically by:

$$y(t) = \int_{t}^{t+T} x(\tau)\, d\tau$$

The output signal *y* at any time *t*, is the result of integrating the input function *x*(*t*) over a *T* second interval beginning at time *t*. The token arrangement is shown in Figure 7.2, with a delay of one-half second.

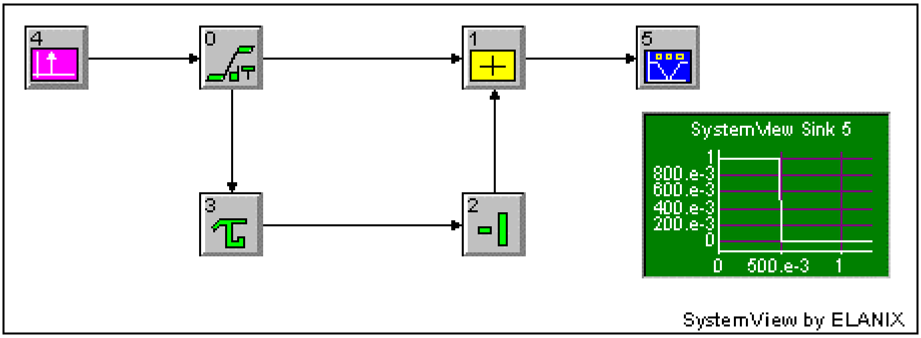


Figure 7.2 Example of a Finite Time Integrator System

The system in Figure 7.2 is equivalent to writing the above equation in the form:

$$y(t) = \int_t^\infty x(\tau)\,d\tau - \int_{t+T}^\infty x(\tau)\,d\tau$$
$$= \int_t^\infty [x(\tau) - x(\tau - T)]d\tau$$

To generate an integrate and dump MetaSystem, perform the following steps:

1. Create the system shown in Figure 7.2 using:
   Token 0 -- Integral Operator, Group Integral/Differential
   Token 1 -- Adder
   Token 2 -- Negate Operator, Group Gain/Scale
   Token 3 -- Delay Operator, Group Delays
   Token 4 -- Impulse Source, Group Aperiodic
   Token 5 -- SystemView Sink, Group Graphic

2. Define System Time and execute the system. Viewing the output in the Analysis window shows the proper T (0.5 sec for this example).

3. Convert a subsystem into a MetaSystem using the create MetaSystem button on the tool bar.

4. Press control, and using the mouse button, surround the following tokens, integrate, adder, delay and negate, then release the mouse button. The resulting screen places the MetaSystem token in place of the operator tokens as shown below:

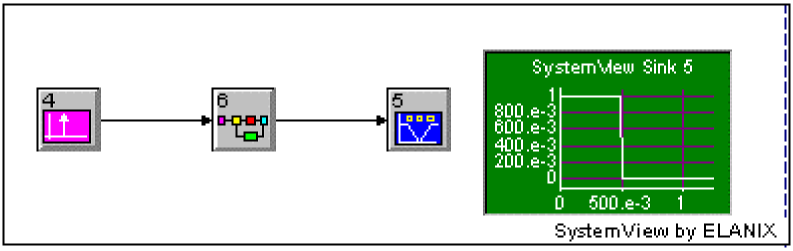


Figure 7.3 Tokens Collapsed to MetaSystem

5. View the resulting MetaSystem, using the View MetaSystem Button.

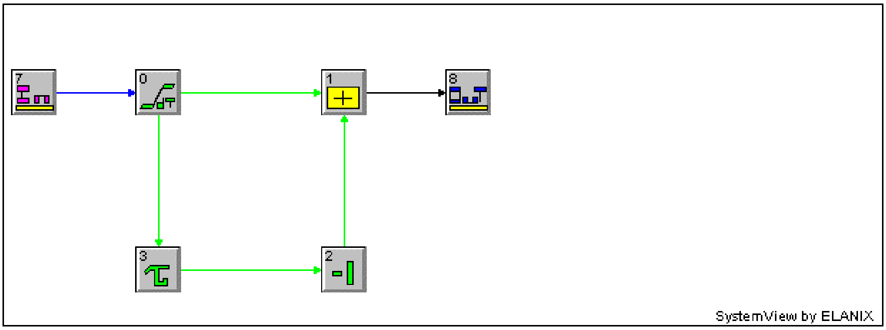


Figure 7.4 The Integrate and Dump MetaSystem

The In token represents a MetaSystem input and the Out token represents a MetaSystem output. The yellow highlight under the In Token (7), and the Out Token (8), indicates that the tokens are connected.

Save the MetaSystem as I&D.mta using the MetaSystem File menu, or duplicate the MetaSystem by clicking on the duplicate button.

Exit the MetaSystem Window by using the return button.

To incorporate a saved MetaSystem into the system simulation, use the I&D MetaSystem just developed. The integrate and dump filter is a linear filter whose impulse response is a unit high step of *T* units in length. To continue the example shown in Figure 7.5, construct the system.

Launch a Source token. Double click and define it as a sawtooth source and use the default parameter values.

Use the duplicated MetaSystem token, or launch a generic MetaSystem from the token Reservoir. Double click on the new token, and the file dialog box will appear. Select the I&D.mta file created above, and then click OK to define the new MetaSystem as I&D.mta. To verify this, place the mouse on the token and read the identification on the information bar.

Now launch a Sink token and define it as SystemView.

To connect the system, click the Connect button and then click the Source token followed by the MetaSystem token.

Complete the connections, click on the MetaSystem token and Sink token. Execute the system. The output in the Analysis window shows the proper *T* (0.5 sec for this example).

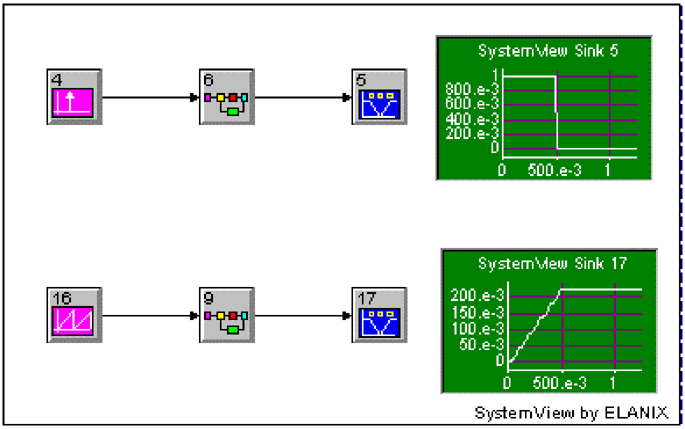


Figure 7.5 Example of two systems using a common MetaSystem token

7.5 MetaSystem I/O Tokens

MetaSystem I/O tokens connect a MetaSystem to the system being designed. All connections to the MetaSystem are made using only these tokens:

- Each MetaSystem output token can be connected to one token within the MetaSystem and any number of tokens in the parent system.
- Each MetaSystem input token can be connected to any number of tokens within the MetaSystem and to one token in the parent system.

MetaSystem I/O tokens are created as part of a MetaSystem, if they are created from an existing system that has already been connected, or by adding them directly to a MetaSystem. To create a MetaSystem I/O token, launch the I/O token located in the token Reservoir. The MetaSystem I/O Library menu will appear and the user then selects Input or Output as shown in figure 7.6.

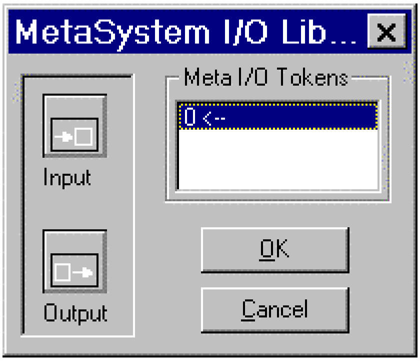


Figure 7.6 The MetaSystem I/O Library

7.6 MetaSystem Screen

The MetaSystem window as shown in Figure 7.7, is similar to the SystemView design window. It can be entered during design, as well as during system execution.

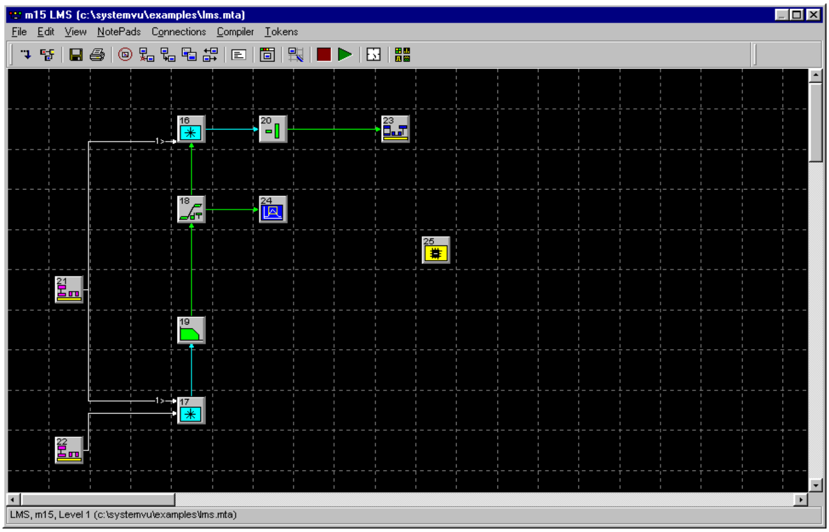

m15 LMS (c:\systemvu\examples\lms.mta)
File Edit View NotePads Connections Compiler Tokens
16
20
23
18
24
25
21
19
17
22
LMS, m15, Level 1 (c:\systemvu\examples\lms.mta)

Figure 7.7 MetaSystem screen

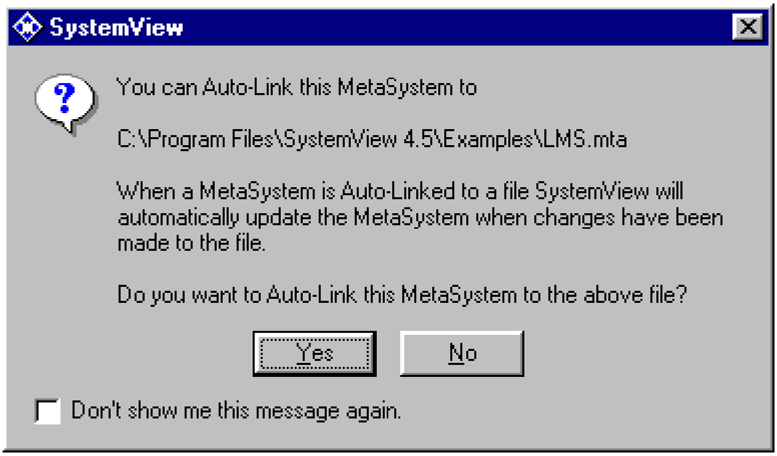
Tokens are added to a MetaSystem from the token reservoir in the same manner as the system design area. The MetaSystem has its own tool bar that is similar to a full design area. The MetaSystem **tool bar** includes two types of returns:

The System button returns directly to the top level of the design.

The Return button on the MetaSystem toolbar returns to the top level of design, one level at a time, by the same route used to get into the MetaSystem (for example, through several nested MetaSystems).

7.7 MetaSystem Auto Links

When a MetaSystem is defined from a file, it can be linked so that all modifications to the system can be saved to the MetaSystem *.mta file. After saving an existing MetaSystem, or defining a new MetaSystem, the following dialogue box is presented:



SystemView

You can Auto-Link this MetaSystem to

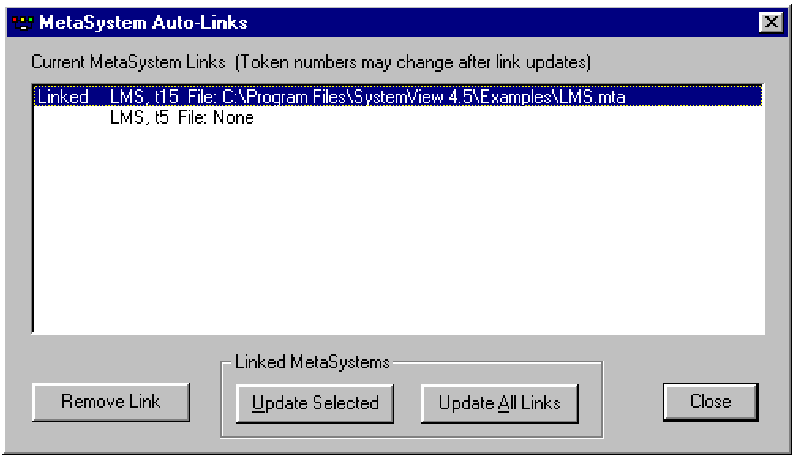
C:\Program Files\SystemView 4.5\Examples\LMS.mta

When a MetaSystem is Auto-Linked to a file SystemView will automatically update the MetaSystem when changes have been made to the file.

Do you want to Auto-Link this MetaSystem to the above file?

□ Don't show me this message again.

Once a MetaSystem is linked, go to the Tools menu and select the MetaSystem Auto Links option, which will present the following:



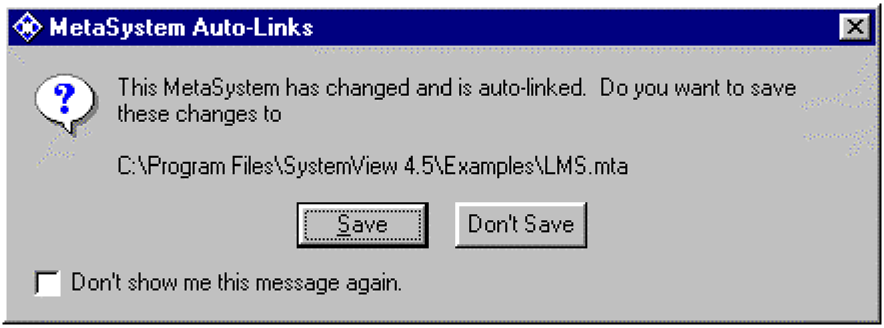
MetaSystem Auto-Links

Current MetaSystem Links (Token numbers may change after link updates)

```
Linked    LMS, t15  File: C:\Program Files\SystemView 4.5\Examples\LMS.mta
          LMS, t5  File: None
```

As an option, a link may be removed, which will cause the MetaSystem to be saved locally. The *.mta will not be updated with any changes to the MetaSystem.

Making changes to the MetaSystem, will present the following screen to save the changes.



MetaSystem Auto-Links

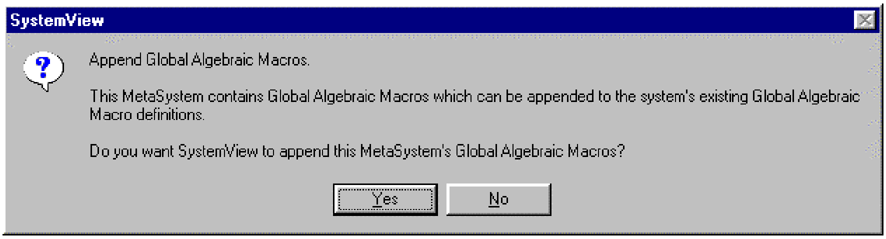
This MetaSystem has changed and is auto-linked. Do you want to save these changes to

C:\Program Files\SystemView 4.5\Examples\LMS.mta

□ Don't show me this message again.

When changes are made to the *.mta file, and it is linked to a MetaSystem, go to Tools\MetaSystem Auto Links and Update Links Now. This will insure that the MetaSystem is in sync with the one saved.

Should a MetaSystem contain any global parameter links, it will ask the user to append or merge with current global parameters.



SystemView

Append Global Algebraic Macros.

This MetaSystem contains Global Algebraic Macros which can be appended to the system's existing Global Algebraic Macro definitions.

Do you want SystemView to append this MetaSystem's Global Algebraic Macros?

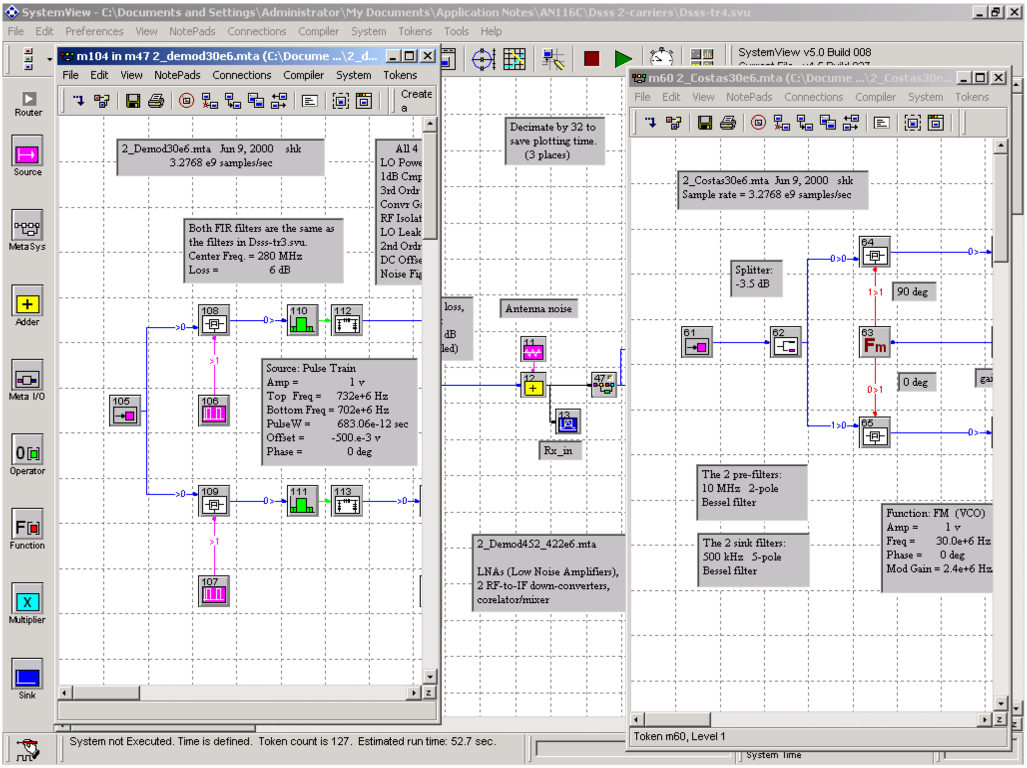
7.8 Guidelines for Using MetaSystems

- MetaSystems may contain other MetaSystems and may be used to add multiple levels of hierarchy to a design.
- Connections and token parameters within MetaSystems can be changed, and new tokens may be added.
- MetaSystems may incorporate multiple I/O tokens.
- When viewing a MetaSystem, place the pointer on an I/O token. The information window will display a description of tokens in the previous level that are connected to the selected I/O token.
- Any changes made to the parameters of tokens within a MetaSystem remain local, unless all links are updated from MetaSystem Auto-Links section. They are not distributed globally to other copies of a MetaSystem within the same design, nor to the original MetaSystem source file.

Warning: Changes to the source file will not affect other systems that have imported the MetaSystem prior to the changes being made. To update existing files with the new version of the MetaSystem, edit the MetaSystem Auto-Links and Update All Links or Selected Links from the Tools menu in SystemView.

The user may open multiple MetaSystems and perform connectivity and copy tokens between the opened MetaSystems. Figure 7.8 below, shows two MetaSystems open in the SystemView design space. Use the mouse to drag and drop tokens between MetaSystems. If a connection is desired from one MetaSystem to another, simply drag the desired connection between MetaSystems. The connection will appear within the SystemView design space.

Once a MetaSystem is opened, the file information is saved in memory. This new change re-opens MetaSystems without delay. Previous versions of SystemView would re-create the MetaSystem each time it was viewed. To release memory, click File I Close, and Release Memory.



SystemView - C:\Documents and Settings\Administrator\My Documents\Application Notes\AN116C\Dsss 2-carriers\Dsss-tr4.svu
File Edit Preferences View NotePads Connections Compiler System Tokens Tools Help
m104 in m47 2_demod30e6.mta (C:\Docume ...\2_d...
File Edit View NotePads Connections Compiler System Tokens
Create a
SystemView v5.0 Build 008
m60 2_Costas30e6.mta (C:\Docume ...\2_Costas30e...
Router
Source
MetaSys
Adder
Meta I/O
Operator
Function
Multiplier
Sink
105
106
107
108
109
110
111
112
113
11
12
13
47
61
62
63
64
65
Fm
Rx_in
0>1
1>1
0>0
1>0
Token m60, Level 1
System Time

2_Demod30e6.mta Jun 9, 2000 shk
3.2768 e9 samples/sec

Both FIR filters are the same as the filters in Dsss-tr3.svu.
Center Freq. = 280 MHz
Loss = 6 dB

All 4
LO Powe
1dB Cmp
3rd Ordr
Convr Ga
RF Isolat
LO Leak
2nd Ordr
DC Offse
Noise Fig

Source: Pulse Train
Amp = 1 v
Top Freq = 732e+6 Hz
Bottom Freq = 702e+6 Hz
PulseW = 683.06e-12 sec
Offset = -500.e-3 v
Phase = 0 deg

Decimate by 32 to save plotting time. (3 places)

loss,
dB
led)

Antenna noise

2_Demod452_422e6.mta

LNAs (Low Noise Amplifiers),
2 RF-to-IF down-converters,
corelator/mixer

2_Costas30e6.mta Jun 9, 2000 shk
Sample rate = 3.2768 e9 samples/sec

Splitter:
-3.5 dB

90 deg

0 deg

gai

The 2 pre-filters:
10 MHz 2-pole
Bessel filter

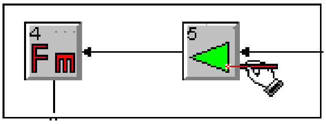
The 2 sink filters:
500 kHz 5-pole
Bessel filter

Function: FM (VCO)
Amp = 1 v
Freq = 30.0e+6 Hz
Phase = 0 deg
Mod Gain = 2.4e+6 Hz

System not Executed. Time is defined. Token count is 127. Estimated run time: 52.7 sec.

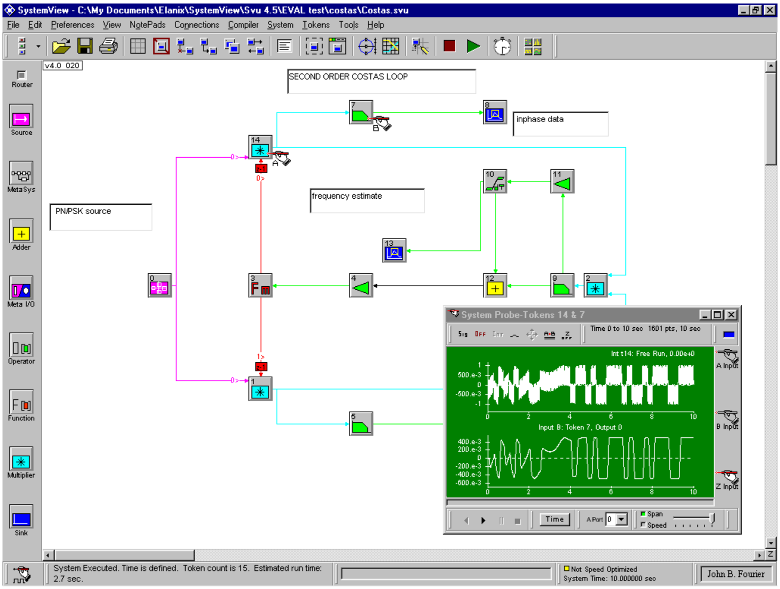
Figure 7.8 Two MetaSystems open

Chapter 8. The Dynamic System Probe



The Dynamic System Probe functions like both an oscilloscope and spectrum analyzer. The three-channel Dynamic System Probe has been integrated into a single tool that is available whenever a system is running. Selection is possible for internal or external trigger sources (tokens), free-run, positive, negative, or positive and negative trigger slopes, and the trigger threshold level.

When the System Probe is active, the user may also *change the parameters of any token in the system dynamically* (i.e., while your system is running) and observe the results in real time. Simply pause the probe and then click the right mouse button on any token and select Edit Parameters from the menu list.


SystemView - C:\My Documents\Elanix\SystemView\Svu 4.5\EVAL test\costas\Costas.svu
File Edit Preferences View NotePads Connections Compiler System Tokens Tools Help
v4.0 020
Router
Source
MetaSys
Adder
Meta I/O
Operator
Function
Multiplier
Sink
SECOND ORDER COSTAS LOOP
inphase data
frequency estimate
PN/PSK source
System Probe-Tokens 14 & 7
Time 0 to 10 sec 1601 pts, 10 sec
Int t14: Free Run, 0.00e+0
Input B: Token 7, Output 0
A Input
B Input
Z Input
Time
A Port 0
Span
Speed
John B. Fourier

System Executed. Time is defined. Token count is 15. Estimated run time: 2.7 sec.

Not Speed Optimized
System Time: 10.000000 sec

Figure 8.1. The Dynamic System Probe as it appears in the System window

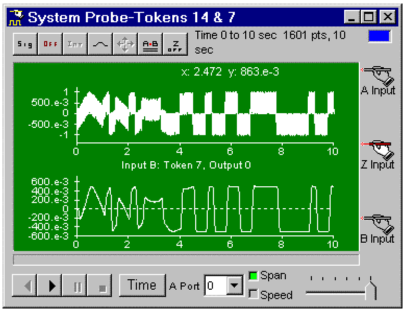


Figure 8.2. (a) System Probe in the A & B time domain mode

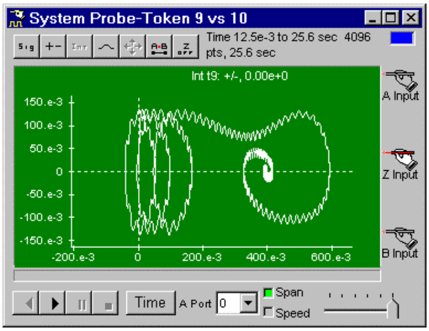


Figure 8.2. (b) System Probe in the A vs. B time domain mode

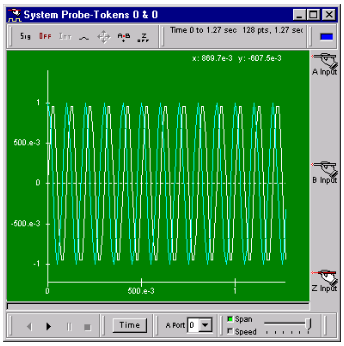


Figure 8.2 (c) System Probe in the overlay of A and B

8.1 Summon Functions

- **Probe Enable** — Click and drag the Probe button to any token within the system. By default this is the A channel input. Click and drag the channel B probe to set the channel B input. To view both channels click the AB button. Each time the AB button is clicked, a different display mode is selected: A vs. B points only, A vs. B connected points, A and B as separate plots, and an overlay of A and B.

- **Tracing a signal through the system** — While the system is running, use the mouse to move the Probe (channel A, or B) to each token in succession.

- **Time/Frequency display** — Click the Probe Time/Frequency button. This button toggles between a Time domain and frequency domain presentation.

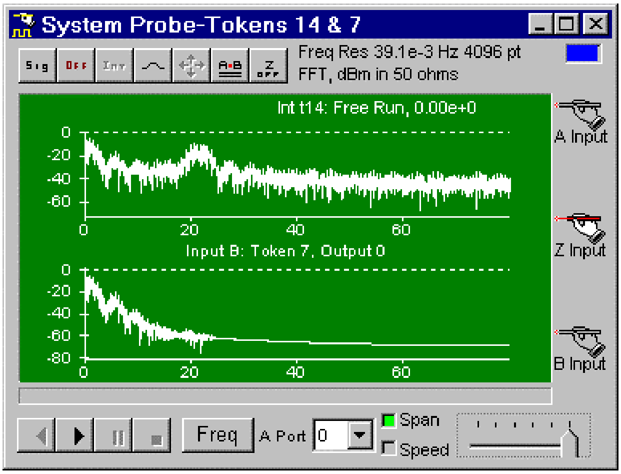
- **Dynamic token parameter modification** — During system execution, click the Probe Pause button, indicated by the symbol (| |), then click any token with the right mouse button and select Edit Parameters.

- **Start/Stop/Pause/End/Restart the run** — Click the appropriate buttons on the probe. While the run is paused, the current Sink data in the Analysis window may be observed and processed.

- **Trigger threshold** — Click the trigger view button (SIG/TRG) and adjust the trigger threshold using the mouse. If the yellow trigger threshold bar is not visible, click the mouse anywhere in the waveform display. Click the button again to return to signal view mode. It is possible to have two independent trigger levels, internal and external trigger modes.

- **Selecting an external trigger source** — While the system is running click the INT button (internal/external trigger) and then select any token in the system to designate the external trigger.

- **Z-Channel Input** — The Z input modulates the probe trace, either by changing the trace color or by changing the trace intensity (select the mode using the Z toolbar button on the probe).



System Probe-Tokens 14 & 7

Freq Res 39.1e-3 Hz 4096 pt
FFT, dBm in 50 ohms

Int t14: Free Run, 0.00e+0

Input B: Token 7, Output 0

Figure 8.3. System Probe in the frequency domain mode

8.2 Dynamic System Probe Controls

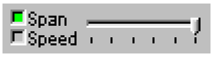
🟦 The LED indicates Probe status. Flashing red means system is paused, constant red means probe is waiting for valid trigger, green means waveform is displayed in real time, blue means the run has finished.

Sig The SIG/TRG button toggles between trigger (internal or external) view and signal view. Use trigger view to adjust the trigger threshold.

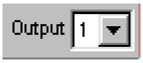
OFF Use the Slope button to select a trigger slope: + (pos), - (neg), + - (pos and neg), or Off (free-run).

Int The INT/EXT button toggles between internal and external trigger modes. While in EXT mode, click the SIG/TRG (trigger view) button to select an external trigger token with the mouse.

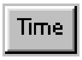
⌒ This button toggles the probe persistence on and off (e.g., to create eye diagrams or overlaid spectra)



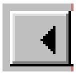
This slider adjusts time or frequency span, or the system run speed. Click Span or Speed before adjusting the slider with the mouse. *Use the arrow keys for fine, and Page Up and Page Down for coarse adjustments.*



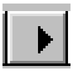
Select a particular output port from tokens having multiple outputs.



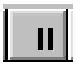
Toggles between a time and frequency (spectrum) display.



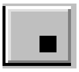
Restart the system run at any time (e.g. during a run).



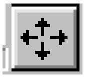
Starts system run, like clicking Run button on System toolbar.



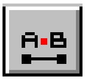
System run pause. While paused, current Sink data in the Analysis window and/or change token parameters can be observed. Click again to resume the system run.



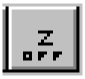
Stops system run, like clicking Stop button on the System toolbar.



Forces the System Probe to re-scale the current display.



Click to step through the AB modes: A vs. B points only, A vs. B connected points, A and B as separate plots, overlay A and B, and Off.



Click to step through Z display modes: color trace modulation (red 0-255, green 256-511, blue 512-767), intensity trace modulation (gray scale: 0-255), and Off.

Chapter 9. The Analysis Window

9.1 The Environment

The Analysis window is the primary tool used to view and process Sink data. There are a variety of options that enhance flexibility and usefulness of the displays. Figure 9.1 shows the Analysis Window.

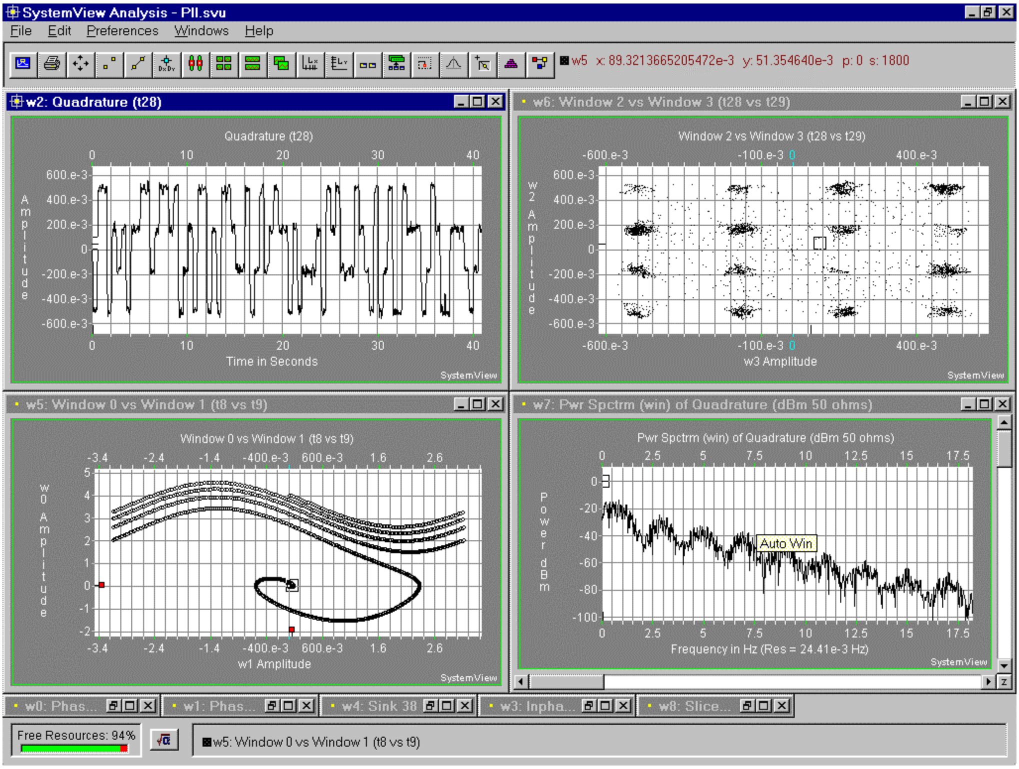

SystemView Analysis - Pll.svu
File
Edit
Preferences
Windows
Help
w5 x: 89.3213665205472e-3 y: 51.354640e-3 p: 0 s: 1800
w2: Quadrature (t28)
Quadrature (t28)
Amplitude
Time in Seconds
w6: Window 2 vs Window 3 (t28 vs t29)
Window 2 vs Window 3 (t28 vs t29)
w2 Amplitude
w3 Amplitude
w5: Window 0 vs Window 1 (t8 vs t9)
Window 0 vs Window 1 (t8 vs t9)
w0 Amplitude
w1 Amplitude
w7: Pwr Spctrm (win) of Quadrature (dBm 50 ohms)
Pwr Spctrm (win) of Quadrature (dBm 50 ohms)
Power dBm
Auto Win
Frequency in Hz (Res = 24.41e-3 Hz)
SystemView
w0: Phas...
w1: Phas...
w4: Sink 38
w3: Inpha...
w8: Slice...
Free Resources: 94%
w5: Window 0 vs Window 1 (t8 vs t9)

Figure 9.1. The Analysis window

The Analysis window includes a powerful Sink Calculator that provides tools for performing block-processing operations within or between, individual plot windows. These block-processing operations may be chained for sequential processing and are automatically re-computed when new Sink data is generated.

9.2 Accessing the Analysis Window and Displays

To access the Analysis window, click the analysis button on the toolbar.

The first time the Analysis window is entered, SystemView will ask if the user wants to perform the block processing sequence as saved with the system file. If no sequence exists then the system Sinks will be displayed as either icons or as open plot windows, depending on the state of the Analysis window when the system file was saved.

9.3 Managing Plot Windows and System Resources

Plot windows may exist in one of four modes:

- **Closed** — Only Sink windows can exist in this mode and still preserve data. This mode uses the least amount of system resources. Select this mode by clicking the Close window button (x) located in the upper-right corner of an open plot window. All screen graphics information is released, so the window must be re-plotted when it is opened again. If a calculated window is closed, and needs to be opened again, it must be redefined using the Sink Calculator.

- **Minimized** — Both Sinks and calculated windows may exist in this mode. This mode uses less system resources than an open plot window. Enter this mode by clicking the Minimize window button (-) located in the upper-right corner of an open plot window.

- **Open Normal** — Double clicking a window icon will open a plot window in the normal mode. To return a maximized plot window to the normal mode, click the Normal window button located in the upper-right corner of the plot window.

- **Open Maximized** — To maximize a plot window, click the Maximize window button located in the upper-right corner of an open plot window. To go from a window icon directly to a maximized plot window, click the icon with the right mouse button and select Maximize.

Free Resources — The percentage of free Windows system resources is displayed in the lower corner of the Analysis window. This is a percent of the total resources, (physical memory and virtual memory) available to SystemView.

The free resources indicator should be greater than 10 percent. SystemView will provide a warning when the percentage becomes low, and may even prevent the opening of a plot window. If this happens, close as many plot windows as necessary to maintain a free memory level above 10 percent.

Virtual Window — The Analysis Windows can create a virtual display area that is larger than the PC monitor's display area. With no plot windows opened in a maximized mode, and at least one plot window opened in the normal mode, move a plot window by clicking and dragging that plot window's title area (at the top of the plot window) until a portion of that window is off-screen. This will cause horizontal and vertical scroll bars to appear at the bottom and right side of the Analysis window. These scroll bars indicate that a virtual viewing area has been created, and are used to position the monitor display area within the virtual display area.

For example, use the scroll bars to position the off-screen plot window such that it is entirely visible. A plot window in the normal display mode can be manually resized by placing the mouse pointer on a border such that the mouse pointer changes to either a left-right arrow or an up-down arrow and then click and dragging the border to a desired position. To undo the virtual viewing area, retile in the Analysis window.

Arranging Plot Windows — There are three green buttons grouped together on the Analysis window toolbar that will automatically arrange your open plot windows, they are: Arrange Vertical, Arrange Horizontal, and Cascade. Try each of these with several plot windows open to view the resulting arrangements.

9.4 Zoom and Scale

A powerful feature of the Analysis window is the ability to easily change the scale of any plot. This is useful for examining detailed plots.

Zoom — Zooming is the process of expanding an area of a plot for a magnified view. To zoom an open plot window, position the mouse in the plot area and click/drag the mouse. A box will appear that outlines the area to be zoomed. When the mouse is released, SystemView will display the outlined area in a magnified view. The user may zoom a previously zoomed plot.

It is possible to simultaneously zoom multiple plot windows in the Analysis window. Press the Ctrl *and* the Alt key together and click/drag the mouse. All open plot windows will be zoomed to the same scale.

Moving around within the plot — Once a plot is zoomed, use the vertical and horizontal scroll bars that are within the plot window to move to different locations within the data. Simultaneously moving the zoomed plot in both dimensions may be done, by clicking the mouse within the plot area and dragging the plot waveform left, right, up or down.

Resetting the Scale — After zooming, or performing any other scale operation, click the Reset Scale button on the toolbar, or click the right mouse button within the plot and select Rescale to return to the default full-scale view.

Log Scales — Click the Log X or Log Y button on the toolbar for a logarithmic axis. Use the Reset Scale button or click the Log X or Log Y button again to return to a linear scale.

Manual scales — Use the Sink Calculator to manually set plot scales to specific values. Click the Sink Calculator and select the Scale tab.

The Scale Axis option will scale an axis algebraically. For example, subtracting a constant *c* on the time axis, shifts the waveform to the right according to *s(t-c).*

The Set Min Max option allows setting the ranges of start and stop values, of either axis.

9.5 Axis Labels, and Plot Titles

The axis labels and plot titles can be customized in any plot window. Double click the label or title and enter the desired text. Press Enter or click outside the text box to apply the new title or label, or press the Esc key to exit without applying the new text.

.9.5.1 Plot Labels

To customize plot labels, click the right mouse button within any plot window and select *New Label*. Position the label using the mouse. Click the label with the right mouse button to customize font, font size, color, etc. Double click the label to change the text.

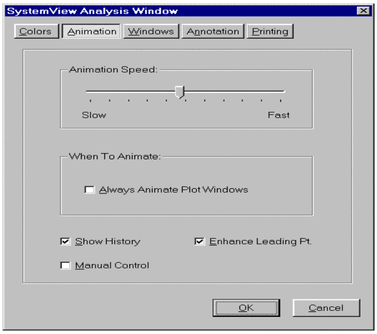
9.6 Copy and Paste to Other Programs

The user can export plot windows as scaleable metafiles or bitmaps, to other Windows applications using Windows copy and paste functions. The user may copy and paste the plot windows into a word document for inclusion as figures in reports or proposals.

To access the copy command, click the Edit menu (or the right mouse button) and select Copy as Bitmap or Copy as Metafile.

9.7 Plot Animation

SystemView provides a plot animation tool for your Analysis, as shown in figure 9.2. Animation is particularly informative when using the scatter plot feature.



Animation Speed:
Slow
Fast

When To Animate:
□ Always Animate Plot Windows

☑ Show History
☑ Enhance Leading Pt.
□ Manual Control

Figure 9.2. The Animation tab in Customize Features

To control or customize the animation, click the Preferences menu in the Analysis window, select Customize, and then click the Animation tab.

- **Animation speed scroll bar** — Use this to set the animation speed.
- **Always Animate** — When selected, all plot windows will be animated when the zoom or Reset Scale button is clicked. Otherwise, a plot window is animated only when the Animate button is clicked.
- **Show History** — When selected, each point plotted stays on the screen. Otherwise, each point is erased before the next is plotted.
- **Manual control** — When selected, the plot animation proceeds in one of two ways. One, the space bar allows stepping through each point in the plot. Or two, holding the space bar, and the points will to plot until released. The animated plot window will be enhanced for increased visibility.

9.8 The Sink Calculator

The Sink Calculator is a tool for performing block operations on plot data. Click the Calculator button √α in the Analysis window to open the Calculator.

Select by clicking the tab on the general category, and click the specific operation to be performed. After choosing the operation, select the window to perform the operation. Click the OK button to perform the operation.

Click on **F** to see the SystemView Parser Function list, a complete list of Boolean operations, random numbers, constants, and arithmetic operations

Click on the **Calculator Icon** to open up the standard built-in Microsoft calculator.

Click on the **Equal Sign Icon** to get to the units converter.

Cascaded operations —Cascaded operations allow the creation of custom processing sequences that can be saved as part of the system file. Each time the system is run, a custom sequence is recalculated when the New Sink Data button is clicked.

9.8.1 Scatter Plots

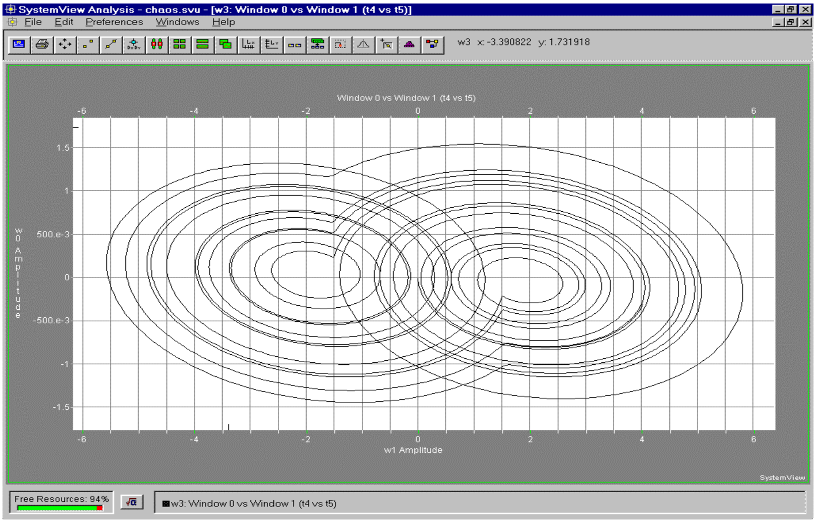
The Scatter plot operation in the Sink Calculator (select the Style tab) allows the plotting of one plot window versus another plot window. The following example will illustrate the procedure.

In the system window, generate a unit amplitude sine and cosine signal, each connected to its own sink.

Enter the Analysis window and click the Sink Calculator, then the Style tab. Select Scatter Plot the select one sink plot from each list, then click OK.

A new plot window will open showing w0 versus w, (the equation $x^2 + y^2 = 1$) f

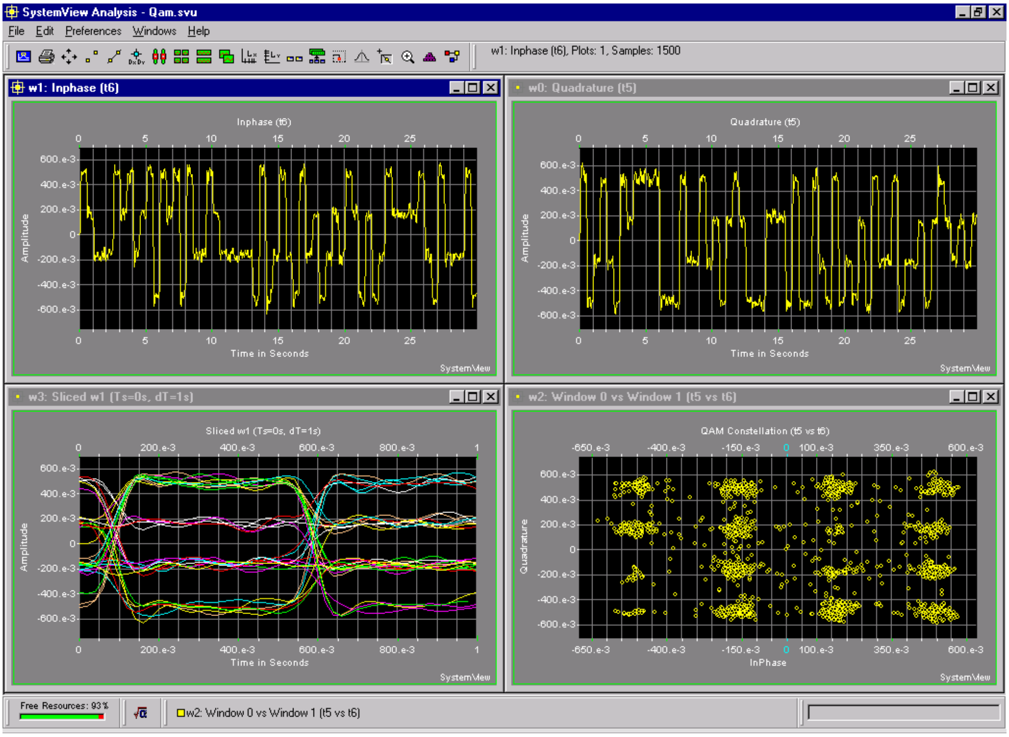
Figure 9.3, shows a system that was designed in SystemView to solve simultaneous differential equations. Figure 9.5 shows the modulation constellation of a QAM signal at baseband and an eye diagram of the quadrature channel these examples are from the example file QAM.svu located in the examples folder.



w3 x: -3.390822 y: 1.731918

w3: Window 0 vs Window 1 (t4 vs t5)

Figure 9.3. Example of the Scatter Plot style using the Sink Calculator



w1: Inphase (t6), Plots: 1, Samples: 1500

w2: Window 0 vs Window 1 (t5 vs t6)

Figure 9.4. Example of Scatter Plot and Time Slice styles

9.8.2 Eye Diagrams and Plot Slicing

The Time Slice operation in the Sink Calculator (select the Style tab) allows you to create an overlapped (folded) plot from a single plot. A common use of this feature is to produce waterfall plots and also the “eye diagrams” used in analyzing distortions in digital communication systems.

The following example will illustrate the use of the Time Slice. Define system time to be 0 to 1 second and the System Sample Rate to be 100 Hz. Connect a 5 Hz sinusoid Source into a Sink, and run the system. Enter the Analysis window and click the Sink Calculator. Select the Style tab and then Time Slice. Plot window w0 has Start Time = 0 sec, and Stop Time = 1 sec

Specifying a Start Time as zero seconds and the slice Length as 0.25 seconds, the original plot w0 will be folded back on itself as follows,

Trace 1 $0 \leq t \leq 0.25$

Trace 2 $0.25 \leq t \leq 0.5$

Trace 3 $0.5 \leq t \leq 0.75$

Trace 4 $0.75 \leq t \leq 1.0$

If the output was a filtered digital waveform having a bit time of *T* seconds, then selecting a slice, Length = *T* seconds, would generate bits folded back on each other, to produce the eye diagram.

9.8.3 Definitions

The Sink Calculator provides various block operations for manipulating and analyzing Sink data. The operations are organized by the tab index in the Sink Calculator window. The definitions, w_n refers to selected plot window *n*, and w_{calc} refers to the new plot window created by the Sink Calculator operation on w_n.

Algebraic

Square — Squares the selected window

$$w_{calc} = w_n^2$$

Square Root — Computes the square root of the selected window,

$$w_{calc} = \operatorname{sgn}\left(w_n\right)\sqrt{\left|w_n\right|}$$

Sin — Computes the sine of the selected window

$$w_{calc} = \sin(w_n)$$

Cosine — Computes the cosine of the selected window

$$w_{calc} = \cos(w_n)$$

Tangent — Computes the tangent of the selected window

$$w_{calc} = \tan(w_n)$$

Arctangent — Computes the arctangent of the selected window

$$w_{calc} = \tan^{-1}(w_n)$$

Max, Min, Avg. — Computes the maximum, minimum, or average of the selected multiple plot window

$$w_{calc,i} = \max\Big|_k, \min\Big|_k, avg\Big|_k \left\{ w_{n,i}^{k} \right\}$$

Where $w_{n,i}$ is the *i*-th sample in w_n and *k* is the *k*-th plot in w_n

Window^a — Raises the selected window to the *a* power (Note the different definition if *a* is fractional)

$$w_{calc} = w_n^a \text{ , if } a \text{ is an integer}$$

$$w_{calc} = \operatorname{sgn}\left(w_n\right) \cdot \left|w_n\right|^a \text{ , if } a \text{ is fractional}$$

a^Window — Raises the constant *a* to the selected window power

(Note the different definition if the window value is fractional.)

$$w_{calc} = a^{\left(w_n\right)} \text{ , If } w_n \text{ is an integer}$$

$$w_{calc} = \operatorname{sgn}\left(a\right) \cdot \left|a\right|^{\left(w_n\right)} \text{ , If } w_n \text{ is fractional}$$

a^{x}Log — Computes the log of the selected window, the specified base *b*

$$w_{calc} = Log_b\left(w_n\right) \text{ , If } w_n > 0$$

$$w_{calc} = \operatorname{sgn}\left(w_n\right) \cdot Log_b\left(\left|w_n\right|\right) \text{ , If } w_n < 0$$

$$w_{calc} = Log_b\left(10^{-300}\right) \text{ , If } w_n = 0$$

Arithmetic

Add Windows — Forms the arithmetic sum of the selected plot windows.

Add Plots — Forms the arithmetic sum of all plots *within* the selected plot window.

Subtract — Subtracts the second selected window from the first.

Negate — Negates all plots within the selected plot window.

Normalize Plots — Normalizes each plot in the window to have maximum amplitude of unity.

Normalize Window — Normalizes all plots in the window relative to the global maximum value in the window.

Multiply Windows — Forms the product of all selected plot windows.

Multiply Plots — Forms the product of all plots *within* the selected plot window.

Divide — Divides the first selected plot window by the second.

Reciprocal — Computes w_n^{-1} (If the denominator is zero then 10^{300} is returned.)

Correlation/Convolution

Auto Correlation — This operation computes the circular auto correlation of all plots within the selected plot window. To avoid "end effects", you should window the plots prior to auto correlation by using Apply Window under the Operators tab.

Cross Correlation — This operation computes the circular cross correlation between all plots within the two selected plot windows. To avoid "end effects", you should window the plots prior to auto correlation by using Apply Window under the Operators tab.

Options

Circular (FFT Based)

Circular (Not FFT Based)

Normalize Lag By Number of samples

Convolution — This operation computes the circular convolution between all plots within the two selected plot windows. To avoid "end effects", you should window the plots prior to auto correlation by using Apply Window under the Operators tab.

Operators

Overlay Plots — Overlays all selected plot windows.

Integrate — Integrates all plots in the selected window.

Differentiate — Differentiates all plots in the selected window.

Cumulative Sum — Keeps a running sum of sample values**.**

Magnitude — Computes the magnitude (i.e., absolute value) of all plots in the selected window.

Decimate By — Decimates, removes samples from all plots within the selected plot window by the specified decimation factor *d*, $w_{calc} = w_n$, for all samples *i* such that *i* mod(*d*) = 0

Moving Average — This operation forms the moving average of all plots within the selected plot window using the relation,

$$w_{calc,i} = \frac{1}{N+1} \sum_{j=i-N/2}^{j=i+N/2} w_{n,j}$$

Where $w_{n,j}$ is the *i*-th sample in w_n and *N*+1 is the time window length in samples (*N*=time window/dT).

Overlay Stats — This operation computes and overlays the specified statistics on the selected plot window. When Show Stats Only is selected, only the selected statistics are plotted. If Detrend Only is selected, then the minimum mean square linear trend is subtracted from the selected plot window.

Apply Window — This operation applies the specified window to the selected plot window. It is used as a pre-processing step prior to spectral analysis, circular convolution, or circular correlation.

Apply Normalized — This operation applies a normalized version of the specified window to the selected plot window. Normalization is defined as the integral of the window function being unity.

Scale

Set Min/Max — This operation allows the manual setting of the max and min values for both axes in the selected plot window. Only data within the specified values will be plotted. (**Tip**: *Use the zoom feature with the mouse to interactively set these values.*)

Scale Axis — Algebraically scales the selected axis as if the plots were functions. Specifying *a* as the Multiply By value and *b* as the Add value for the *x*-axis, and specifying *c* as the Multiply By value and *d* as the Add value for the *y*-axis, the Sink Calculator forms,

$w_{calc}(t) = c \cdot w_n(at + b) + d$ Where *t* is the variable of the *x*-axis.

Spectrum

- **Power Spectrum (dBm in 50 ohms)** — This operation produces the power spectrum for all plots within the selected plot window. The spectrum is in dBm assuming 50-ohm impedance at the input to an analog spectrum analyzer. (**Tip**: *Use Moving Average under the Operator tab to smooth the spectrum*)
- **Power Spectrum (dBm in 1 ohm)** — This operation produces the power spectrum for all plots within the selected plot window. The spectrum is in dBm assuming one-ohm impedance at the input to an analog spectrum analyzer. (**Tip**: *Use the Moving Average under the Operator tab to smooth the spectrum.*)
- **Power Spectral Density (dBm/Hz in 50 ohms)** — This operation produces the power spectral density for all plots within the selected plot window. The spectral density is in dBm/Hz assuming 50-ohm impedance at the input to an analog spectrum analyzer. (**Tip**: *Use Moving Average under the Operator tab to smooth the spectrum.*)

- **Power Spectral Density (dBm/Hz in 1 ohm)** — This produces the power spectral density for all plots within the selected plot window. The spectral density is in dBm/Hz assuming one-ohm impedance at the input to an analog spectrum analyzer. (**Tip**: *Use Moving Average under the Operator tab to smooth the spectrum.*)
- **20 Log FFT (dB)** — This operation produces the magnitude FFT of all plots within the selected plot window and displays the result in dB. The forward transform is multiplied by Δt where $\Delta t = 1/f_S$ and f_S = system sample rate. If the selected plot window represents an FFT of another plot window, then the Sink Calculator automatically performs the inverse FFT (no normalization).
- **|FFT|^2** — This operation produces the magnitude squared FFT of all plots within the selected plot window and displays the results using a linear power scale. The forward transform is multiplied by Δt where $\Delta t = 1/f_S$ and f_S = system sample rate.. If the selected plot window is itself an FFT of another plot window, then the Sink Calculator automatically performs the inverse FFT (no normalization).
- **|FFT|** — This produces the magnitude of the FFT of all plots within the selected plot window. The forward transform is multiplied by Δt where $\Delta t = 1/f_S$ and f_S = system sample rate. If the selected plot window is an FFT of another plot window, then the Sink Calculator automatically performs the inverse FFT (no normalization).
- **Phase** — This operation produces the phase (in degrees) of the spectrum of all plots within the selected plot window. The resulting values are modulo ±180 degrees.
- **Group Delay** — This operation will calculate the overall group delay of cascaded filters. Impulse the chain of cascaded filters. Then select the impulse response plot and choose Group Delay.
- **Phase Unwrapped** — This operation produces the unwrapped phase in degrees, (removes the ±180 degree ambiguity), of the spectrum of all plots within the selected plot window.
- **Phase Deviation From MSE Linear Fit** — This operation produces the unwrapped phase deviation from a MSE (mean square error) straight line fit (removes the linear portion of the phase versus frequency) of the spectrum of all plots within the selected plot window.

Style

Scatter Plot This operation produces a scatter plot between all plots within the two selected plot windows. The first window selected is the vertical axis and the second is the horizontal axis (i.e., w1 vs. w2).

Histogram This operation produces a separate histogram of all plots within the selected window. A larger bin size yields a “smoother” histogram.

BER Plot This operation produces a plot of bit error rate versus SNR and automatically creates the appropriate axis labels and title. The Start SNR parameter is assigned to the first BER value, and all subsequent samples are assigned Start SNR + $n\Delta\mathrm{SNR}$, where n = 1,2… is the sample number and ΔSNR is the Increment parameter. From the list of Sinks, select the Sink containing the bit error data from your system.

Time Slice This operation produces multiple overlaid “slices” of each plot in the selected plot window. By default, the last sample of each time slice is repeated as the first sample of the next time slice. The “+ dT” option forces the Sink Calculator to begin each new slice with the next sample, after the last sample of the previous slice, thus, enabling cascaded time slices. The time slice style can be used to produce “eye diagrams” used in digital communication signal analysis, and is used in creating waterfall plots.

Movie This operation will automatically create a "movie" consisting of two or more frames you select from the plot window list. The playback rate is variable and may be paused during playback. Applications include the dynamic presentation of spectral or time behavior of signals.

Contour — The input for the Contour style must be a multiple plot window. For each plot in the input window, the locus of amplitudes within the specified range (Level ± Height/2) is plotted versus the original *x*-axis values. In Figure 9.5, the contour plot of window w3 is shown.

Waterfall — This operation will overlay all plots in the selected plot window, and offsets each in the x- and y-axis by the specified amounts to produce the waterfall effect. Use the Time Slice under this tab to prepare a single plot for waterfall display.

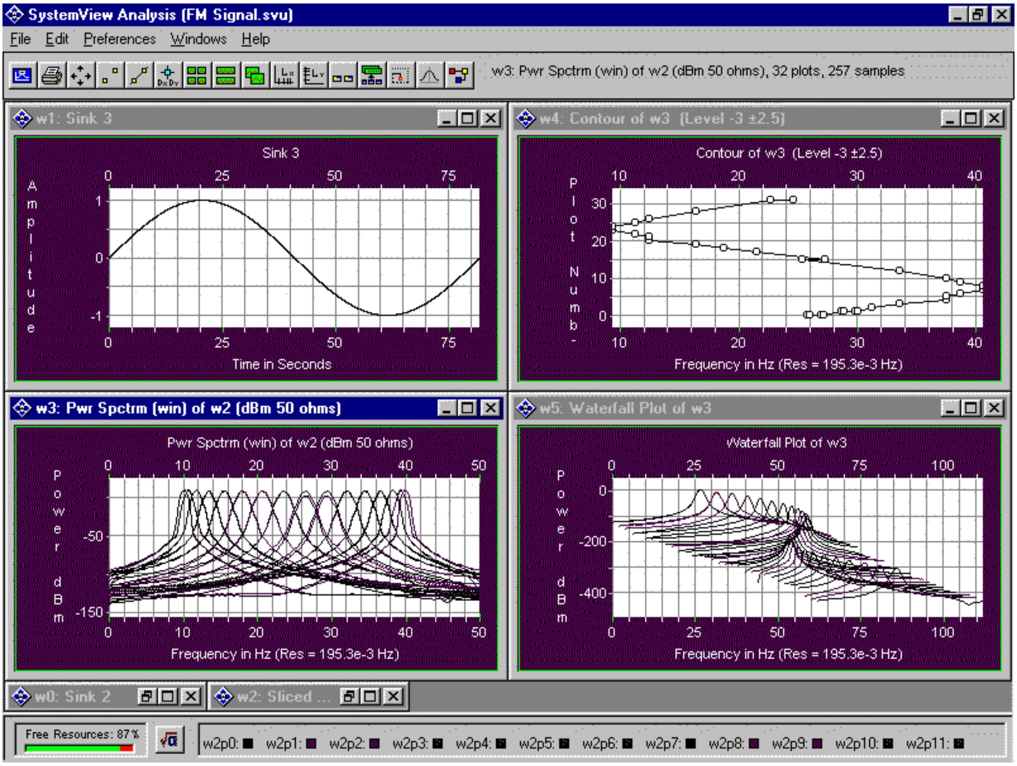

SystemView Analysis (FM Signal.svu)
File Edit Preferences Windows Help
w3: Pwr Spctrm (win) of w2 (dBm 50 ohms), 32 plots, 257 samples
w1: Sink 3
Sink 3
Amplitude
Time in Seconds
w4: Contour of w3 (Level -3 ±2.5)
Contour of w3 (Level -3 ±2.5)
Plot Numb-
Frequency in Hz (Res = 195.3e-3 Hz)
w3: Pwr Spctrm (win) of w2 (dBm 50 ohms)
Pwr Spctrm (win) of w2 (dBm 50 ohms)
Power dBm
Frequency in Hz (Res = 195.3e-3 Hz)
w5: Waterfall Plot of w3
Waterfall Plot of w3
Power dBm
Frequency in Hz (Res = 195.3e-3 Hz)
w0: Sink 2
w2: Sliced ...
Free Resources: 87%
w2p0: w2p1: w2p2: w2p3: w2p4: w2p5: w2p6: w2p7: w2p8: w2p9: w2p10: w2p11:

Figure 9.5. Example of the Waterfall and Contour styles of w3 using the Sink Calculator

Complex FFT

- **FFT** — This operation produces the complex FFT from the selected real and/or imaginary input plot windows. Select the real input from the upper plot window list and the imaginary input from the lower list. The FFT outputs (multiple, if desired) are specified using the output list. The forward transform is multiplied by Δt where $\Delta t = 1/f_s$ and f_s = system sample rate. If the selected real and/or imaginary input plot windows are themselves an FFT of other plot windows, then the Sink Calculator automatically performs the inverse complex FFT.

Data

- **Extract Plots** — This operation separates a plot of overlaid signals into individual plots.
- **Extract Data** — This operation extracts data from the input window beginning at the Start value and ending with the Stop value, and creates a new window.
- **Insert** — This operation inserts the selected window, in the upper plot window list, into the selected window from the lower plot window list. The insertion point is determined by the Start value, and a new window is created.
- **Delete** — This operation deletes data from the input window beginning at the Start value and ending with the Stop value, and creates a new window.
- **Append** — This operation appends the selected window, in the upper plot window list, to the selected window in the lower plot window list. A new window is created.
- **Pad Zeros** — This operation pads (adds) zeros to the selected plot window. The No determines the number of zeros. Zeros value. A new window is created.

Custom

- **Custom Algebraic Window:** This function allows the user to perform any mathematical operation on data. Incorporate trigonometric, algebraic, boolean logic, random variables, and constants (such as pi) in any equation. Also the Q, erf and erfc can be used to create a theoretical BER Curve.

Comm

- **Theoretical BER Plots:** Generate theoretical BER and SER Curves. Click the new Comm button in the Sink Calculator to display a list of available theoretical BER and SER curves. Examples include PSK (coherent), DPSK (coherent and incoherent), and QPSK.

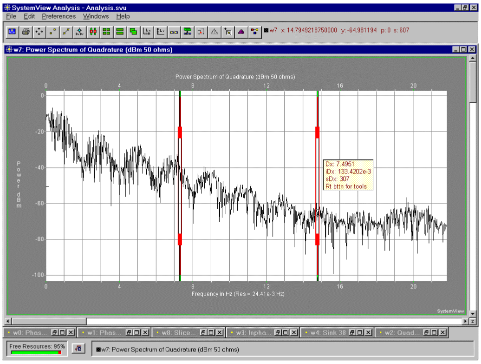
9.9 Special Tools

The Segment Markers, the MicroViewer, and the Differential XY are special tools designed to enhance the data in the Analysis window.

9.9.1 Segment Markers

Toolbar Button:

The Segment markers specify an x-axis range used for subsequent processing. When Segment Markers are positioned in a plot, and the Sink Calculator is used to compute the spectrum, then only the specified segment will be used in the spectral computation.


SystemView Analysis - Analysis.svu
File
Edit
Preferences
Windows
Help
w7: Power Spectrum of Quadrature (dBm 50 ohms)
Power Spectrum of Quadrature (dBm 50 ohms)
0
4
8
12
16
20
0
-20
-40
-60
-80
-100
Power dBm
Frequency in Hz (Res = 24.41e-3 Hz)
SystemView
w0: Phas...
w1: Phas...
w8: Slice...
w3: Inpha...
w4: Sink 38
w2: Quad...
Free Resources: 95%
w7: Power Spectrum of Quadrature (dBm 50 ohms)

w7 x: 14.7949218750000 y: -64.981194 p: 0 s: 607

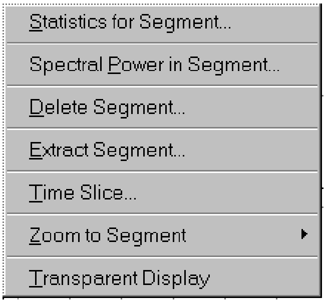
Dx: 7.4951
iDx: 133.4202e-3
sDx: 307
Rt bttn for tools

Figure 9.6 Segment Markers positioned within a plot window.

As either marker is moved with the mouse, the following information is displayed:

- Dx: Distance between markers in the x-axis units.
- iDx: Inverse (1/Dx) distance between markers.
- sDx: Distance between markers in samples.

Click the right mouse button on either Segment Marker to access the following menu:



SystemView will compute or apply the menu selection to the x-axis range specified by the current Segment Marker positions.

9.9.2 MicroView

Toolbar Button:

The MicroView tool shows a magnified view of the plot data as the mouse is moved.

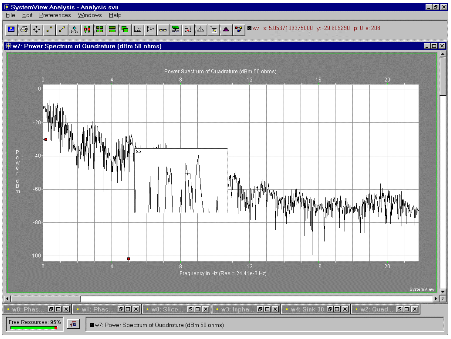

SystemView Analysis - Analysis.svu
File Edit Preferences Windows Help
w7 x: 5.0537109375000 y: -29.609290 p: 0 s: 208
w7: Power Spectrum of Quadrature (dBm 50 ohms)
Power Spectrum of Quadrature (dBm 50 ohms)
0
4
8
12
16
20
0
-20
-40
-60
-80
-100
Power dBm
5.x
Frequency in Hz (Res = 24.41e-3 Hz)
SystemView
w0: Phas...
w1: Phas...
w8: Slice...
w3: Inpha...
w4: Sink 38
w2: Quad...
Free Resources: 95%
w7: Power Spectrum of Quadrature (dBm 50 ohms)

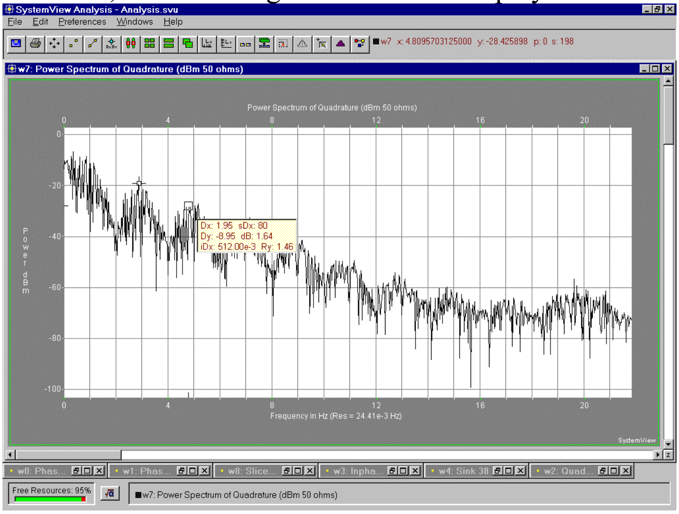
Figure 9.7 Using the MicroView tool to examine plot details.

To resize the MicroView window select the Freeze View entry and click the MicroView window to expose the size blocks. Click and drag any of the size blocks to resize the window. Click on Freeze View again, to return to normal MicroView operation. The text color of the displayed information turns RED when both the mouse and the marker are exactly positioned on a plot sample.

9.9.3 Differential XY

Toolbar Button: Dx Dy

This tool shows the distance between the marker ⌖ and the current mouse position. As the mouse is moved, the following information is displayed:



SystemView Analysis - Analysis.svu

w7 x: 4.8095703125000 y: -28.425898 p: 0 s: 198

w7: Power Spectrum of Quadrature (dBm 50 ohms)

Dx: 1.95 sDx: 80
Dy: -8.95 dB: 1.64
iDx: 512.00e-3 Ry: 1.46

Free Resources: 95%

w7: Power Spectrum of Quadrature (dBm 50 ohms)

Figure 9.8 Using the Differential XY tool to measure relative distances.

Dx: Distance between mouse and marker in the x-axis units.

iDx: Inverse (1/Dx) distance between mouse and marker.

sDx: Distance between mouse and marker in samples.

Dy: Distance between mouse and marker in the y-axis units.

dB: 10 Log/Ymouse/Ymarker|

Ry: Ymouse/Ymarker

Chapter 10. System Note Pads

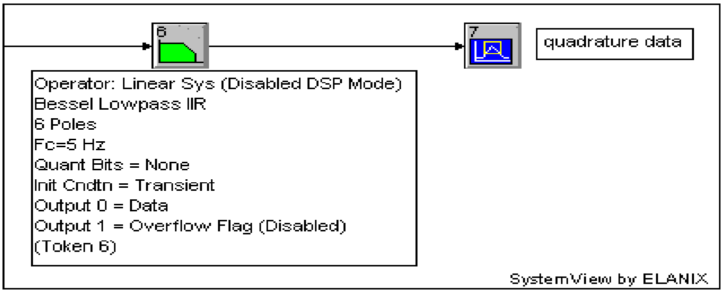
The Note Pads allow annotating the System window with text describing subsystem functionality, parameter values, and other information. To annotate the design, use the New Note Pad button on the toolbar, or the New Note Pad under the Note Pad menu.

Creating a Note Pad — There are three ways to create a note pad:

1. Click the New Note Pad button on the toolbar, and a note pad window will appear near the center of the system screen.. Click in the text area and type the note. To edit the message, use standard Windows keyboard edit commands.
2. Click the NotePads menu above the toolbar, and select New Note Pad. A note pad window will appear.
3. Use the auto-parameter feature described below.

Auto Parameters — It can be useful to annotate the system with note pads, listing the parameter values of the system tokens.

1. Click the NotePads menu above the toolbar.
2. From the menu select Copy-Token Parameters to Note Pad.
3. Select the desired token with the mouse.



Operator: Linear Sys (Disabled DSP Mode)
Bessel Lowpass IIR
6 Poles
Fc=5 Hz
Quant Bits = None
Init Cndtn = Transient
Output 0 = Data
Output 1 = Overflow Flag (Disabled)
(Token 6)

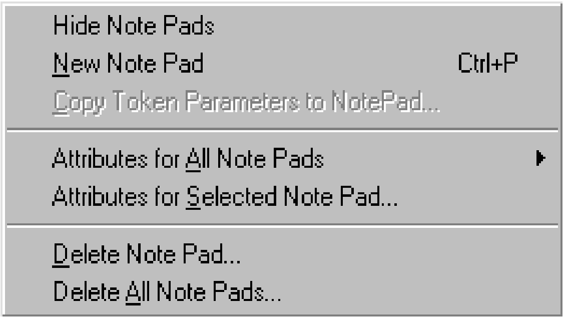
Figure 10.1. Example results of the Parameters-To-Note-Pad operation

The annotated note pad will appear and may be resized or edited as necessary.

Editing/Copying note pad text — To edit or copy text in a note pad:

1. Select the desired note pad.
2. Click and drag to select text.
3. Edit text or copy and paste text into another note pad or any Windows program.

Clicking the Notepads menu, located above the toolbar, to set attributes for selected or for all note pads in the system. Another way to copy text is:


Hide Note Pads
New Note Pad Ctrl+P
Copy Token Parameters to NotePad...
Attributes for All Note Pads ▸
Attributes for Selected Note Pad...
Delete Note Pad...
Delete All Note Pads...

1. Click/drag the mouse to select the text to copy.
2. Press Ctrl + Insert.
3. Click inside another note pad and then press Shift + Insert.

The text from the first pad will be copied into the second note pad.

Moving a note pad — To move a note pad:

1. Click the left mouse button and hold anywhere inside the note pad.
2. Hold down the left mouse button and the perimeter of the note pad will become shaded, indicating it may be moved to another location.

When clicking on text, it is sometimes necessary to click twice before shading appears.

Resizing a note pad — To change the size of a note pad:

1. Click and release the left mouse button anywhere inside the note pad. This produces eight anchor points around the perimeter of the note pad.
2. Place the mouse cursor on the appropriate anchor point.
3. An arrow will appear indicating the direction of the re-sizing.
4. Use the left mouse button to click/drag the anchor point to resize.

Deleting a note pad — There are two ways to delete a note pad:

1. Click the Delete Objects button on the toolbar, and then click on a note pad.
2. The delete feature is also available in the NotePads menu above the toolbar.

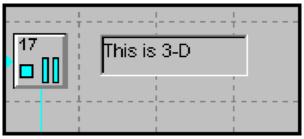
Hiding/Showing all note pads — To show or hide all of the note pads:

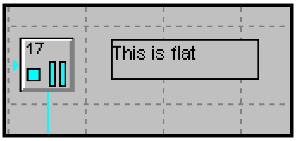
1. Click the Note Pads menu above the toolbar.
2. Select Hide Note Pads or Show Note Pads.

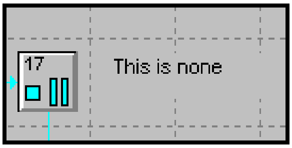
All of the note pads in the System window will be hidden or displayed.

Attributes for Selected/All Note Pads — The attributes (text color, background color, and fonts) for one or more note pads can be changed as desired. Use the NotePads menu and select Attributes for All Note Pads or Attributes for Selected Note Pad. For a single note pad, access the Attributes menu by clicking the right mouse button twice on the note pad. There are no limits on Note Pads.

You can customize the border style of your Notepads (3-D is the default). Select NotePads | Attributes for All NotePads | Border Style. See examples below.







Chapter 11. Printers and Printing

To print the system screen, click the File pull-down menu. There are four print modes:

- Text Tokens
- Symbolic Tokens
- System Summary
- Connection List

11.1 Print System (Text Tokens)

In this mode, the system screen will be printed with “line art” boxes that replace the Symbolic Tokens. The “line art” boxes include text to describe each token.

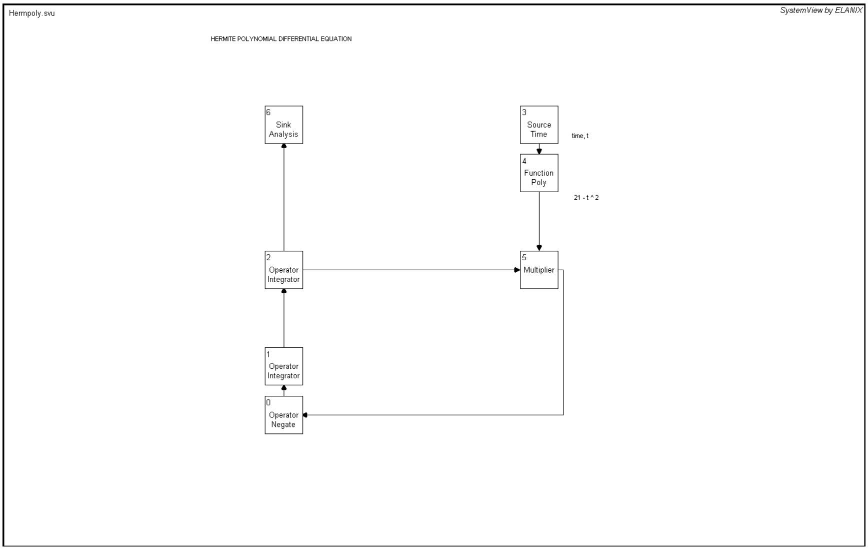


Figure 11.1 Result when using Text Tokens mode to print system screen

11.2 Print System (Symbolic Tokens)

In this mode, the system is printed as it is displayed in the System window.

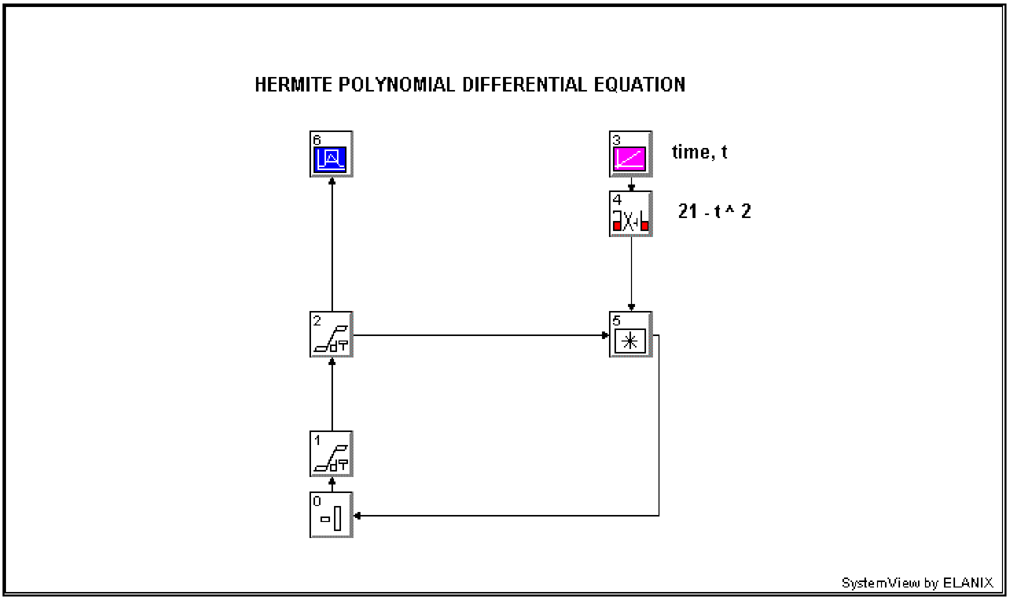
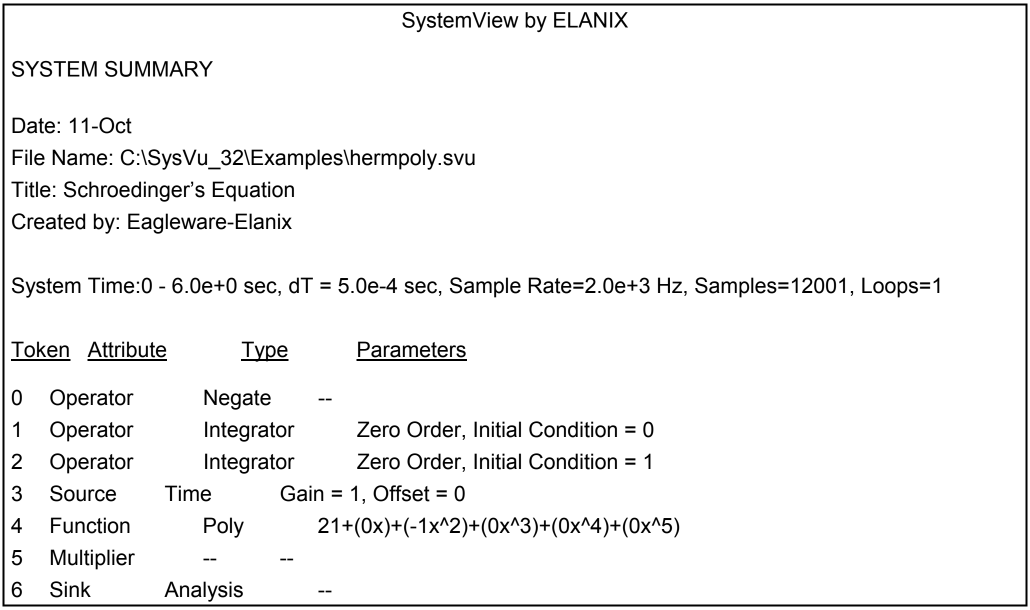


Figure 11.2. Result when using Symbolic Tokens mode to print system screen

11.3 Print System Summary

This mode provides a printing option that will print a description summary of all tokens. A system summary for the Hermpoly.svu example is shown in Figure 11.3.

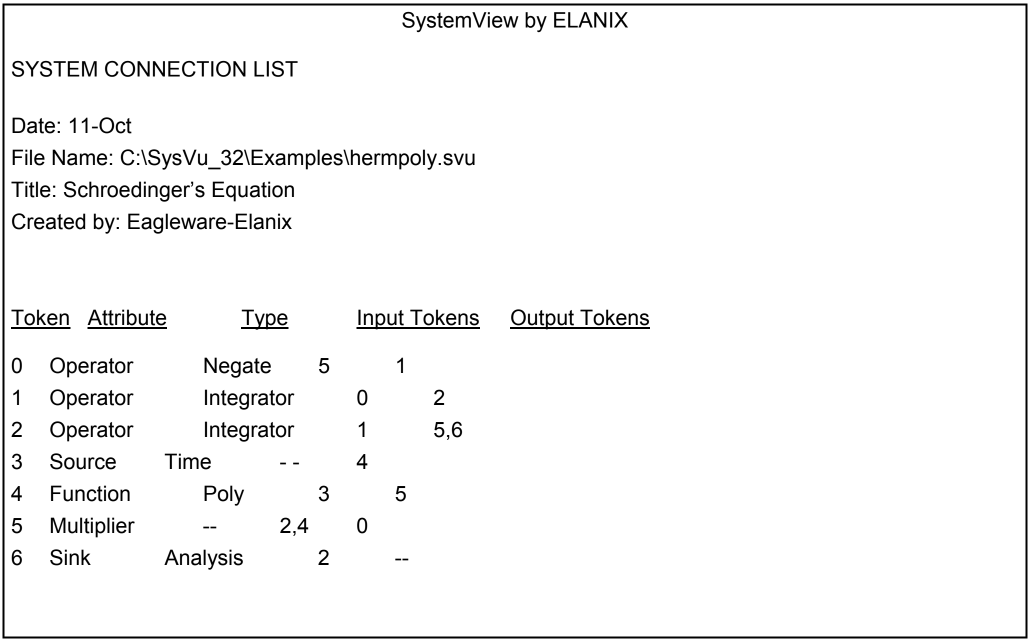


```
                              SystemView by ELANIX

SYSTEM SUMMARY

Date: 11-Oct
File Name: C:\SysVu_32\Examples\hermpoly.svu
Title: Schroedinger's Equation
Created by: Eagleware-Elanix

System Time:0 - 6.0e+0 sec, dT = 5.0e-4 sec, Sample Rate=2.0e+3 Hz, Samples=12001, Loops=1

Token   Attribute       Type        Parameters

0   Operator        Negate      --
1   Operator        Integrator      Zero Order, Initial Condition = 0
2   Operator        Integrator      Zero Order, Initial Condition = 1
3   Source      Time        Gain = 1, Offset = 0
4   Function        Poly        21+(0x)+(-1x^2)+(0x^3)+(0x^4)+(0x^5)
5   Multiplier      --      --
6   Sink        Analysis        --
```

Figure 11.3. Result when using System Summary mode to print system screen

11.4 Print System Connection List

This option will print a list of the tokens, the attributes and the type, in addition to all related input and output tokens.



```
                               SystemView by ELANIX

SYSTEM CONNECTION LIST

Date: 11-Oct
File Name: C:\SysVu_32\Examples\hermpoly.svu
Title: Schroedinger’s Equation
Created by: Eagleware-Elanix


Token   Attribute          Type          Input Tokens     Output Tokens

0   Operator          Negate      5        1
1   Operator          Integrator       0        2
2   Operator          Integrator       1        5,6
3   Source       Time          - -     4
4   Function          Poly          3        5
5   Multiplier        --        2,4      0
6   Sink         Analysis          2        --
```

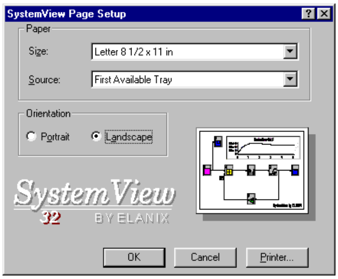
Figure 11.4. Result when using Connection List mode to print system screen

11.5 Printing In Color

To print in color, click the Preferences menu and select Customize. Click the System Colors tab and select Print In Color.

11.6 Printer Setup

Click Printer Setup under the System or Analysis window File menu to open the Print/Page Setup window as shown in Figure 11.5.


SystemView Page Setup

Paper

Size: Letter 8 1/2 x 11 in

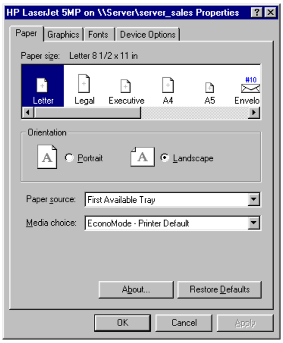
Source: First Available Tray

Orientation

○ Portrait ● Landscape

Figure 11.5. The Printer/Page Setup window

Select either portrait or landscape mode. Clicking the Printer button opens a window that lets you access additional settings to configure your printer. The type and number of options available will depend on your particular printer and may differ from the example shown in Figure 11.6.


HP LaserJet 5MP on \\Server\server_sales Properties

Paper size: Letter 8 1/2 x 11 in

Orientation

○ Portrait ● Landscape

Paper source: First Available Tray

Media choice: EconoMode - Printer Default

Figure 11.6. The Printer Properties window

11.7 Other Methods

The Edit menu allows copying of system block diagrams, waveforms, and other information to the Windows clipboard. This allows incorporation of elements from the design into word processing, desktop publishing, or presentation applications and provides an alternate method of printing.

Chapter 12. Debugging A System Design

After the system design has been completed, click the Execute button, and look at the output. If the output is not what is expected, or doesn't look right, the system may have a "bug" in the design that must be tracked down and fixed. While there are no hard and fast rules for debugging a system, the following techniques may provide some guidance.

12.1 Guidelines for Debugging Systems

- Don't implement a complex system all at once. Build parts of the system one subsystem at a time, and ensure that each subsystem is working correctly before proceeding to the next step.
- Use the Dynamic System Probe. This is a powerful and efficient debugging tool. It can be used to trace signals through the system in both time and frequency, to locate the point where the output is incorrect or looks suspicious.
- Use MetaSystems extensively to help reduce the visual complexity of your system. Test each MetaSystem separately to verify its performance.
- It may be possible to simplify the design to a configuration that is easier to analyze by adjusting a single token value, or temporarily bypassing portions of the system.
- Use the Data List Sink token. Use Sink tokens liberally; although it may be tedious it is sometimes the only way to debug a system, by observing outputs on a point-by-point basis.
- Until the system has been debugged, reduce the number of samples in the simulation (i.e., the run time). This will allow faster debugging of the system, especially when there are multiple problems that must be fixed.

12.2 Using the Dynamic System Probe

SystemView includes a Dynamic System Probe that is a very powerful debugging tool. It allows signals to be observed at the output of a token during system execution. Both the A and B probes can be used to monitor two different signals in the SystemView simulation. To enable the probe, click the probe button located in the lower left-hand corner of the SystemView screen. Drag it to any token and then run the system. In addition, drag Probe B to another token to monitor the output at that point in the system. A small screen will overlay the tokens. Figure 12.1 illustrates the use of the probe in the example file Costas.svu, located in \Program Files\SystemView\examples.

The probe is attached to gain token number five. The execution time can be adjusted using the speed-adjusting dial on the probe screen. A system run button is also included. The probe can be moved from token to token at any time. By moving the probe along a path of tokens, system bugs can be identified faster. See Chapter 8 for additional information on the Dynamic System Probe.

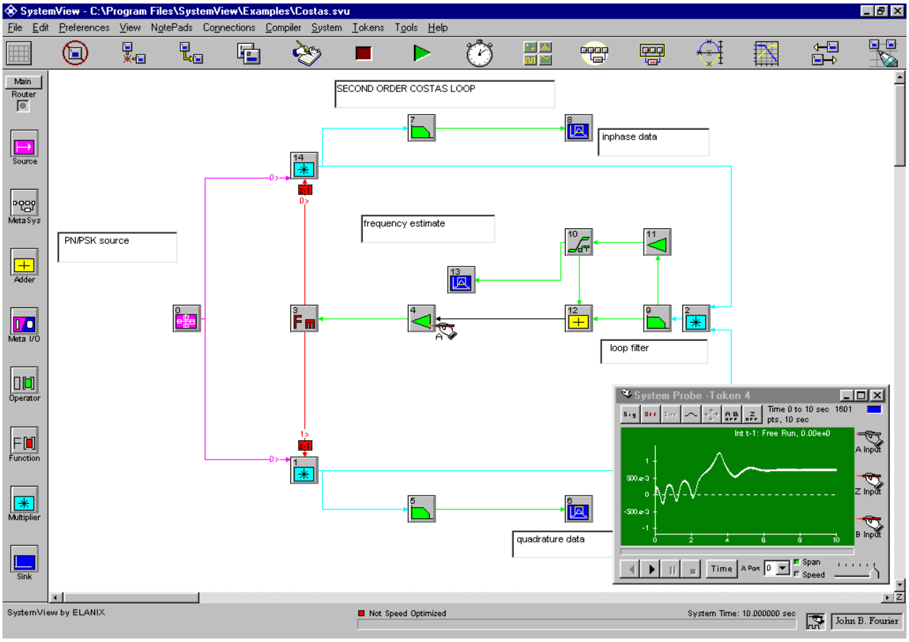


Figure 12.1. Example showing the use of the Dynamic System Probe

12.3 Additional Tools

- Select the Check Connections option from the Connections menu and let SystemView locate unconnected tokens, or tokens that do not have the proper number of inputs.

- The Fix Random Seed option, under the Preference Customize menu, allows numeric patterns from the random number generator to be repeated.

- Under the Preference menu, ensure that the warning message option is enabled.

- Under the System menu, the Single Step option will allow display of each clock cycle separately (use the Data List or Current Value Sink).

- If any feedback loops affect execution, the Find Implicit Delay token under the Tokens menu will locate them.

- The execution sequence can be shown visually using Animate Exe Sequence located in the Compiler menu. This is especially valuable for systems with feedback loops and delays.

Chapter 13. Time Delays In Feedback Paths

This chapter outlines the important timing issues that can affect the results when using feedback loops and delays as part of a system model.

13.1 Theory and Practice

Consider the simple analog feedback system shown in Figure 13.1.

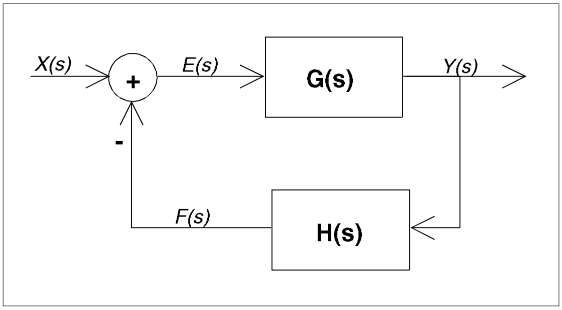


Figure 13.1 Continuous feedback system

The basic Laplace equations describing this system are,

$$Y(s) = G(s)E(s)$$
$$E(s) = X(s) - F(s)$$
$$F(s) = H(s)Y(s)$$

Resulting in the loop transfer function, $\dfrac{Y(s)}{X(s)} = \dfrac{G(s)}{1 + G(s)H(s)}$

Now consider the same loop, only this time we implement the equations as a time-discrete sampled data system.

The equations now become,

$$
\begin{aligned}
y(t_k) &= g(t_k) \otimes e(t_k) \\
e(t_k) &= x(t_k) - f(t_k) \\
f(t_k) &= h(t_k) \otimes y(t_k)
\end{aligned}
$$

Where $h(t_k)$ and $g(t_k)$ are the time impulse responses of the forward and feedback loop transfer functions, and $\otimes$ indicates convolution. To calculate $y(t_k)$, we must first calculate $e(t_k)$. To do this, we need to use the second equation. But to calculate $f(t_k)$, we need to use the last equation. However, to compute $f(t_k)$ we must know $y(t_k)$, which brings us back to where we started. In this example, we find that the equations cannot be implemented as they are. The basic problem is the fact that the feedback loop value is dependent on its own value used elsewhere in the loop. We cannot calculate the value until we calculate the value!

There are two points to consider in this dilemma. The first relates to the Laplace equations used at the beginning of this example. The time problem does not come into play here, since we are working with analytical not physical quantities. Secondly, a real analog loop always has an inherent delay in the feedback connection due to the time required to propagate signals through the system. In a SystemView simulation, the solution to the computational dilemma is to insert a delay of one sample for one of the parameters in the feedback path. The mathematical representation of this delay is realized when we replace the second equation above with $e(t_k) = x(t_k) - f(t_{k-1})$

The loop now includes a one-sample delay in the feedback path. This delay is referred to as the *implicit* delay. SystemView automatically flags the location of the implicit delay with a small box enclosing the [z] indicator, as shown in figure 13.2. This delay is incorporated in every feedback path in every SystemView simulation. The indicator will always appear at an input of a multi-input token. An exception to this occurs in a feedback path, when using the sample delay token in the active mode.

While the indicator [z] does not always appear, a one-sample delay is present in all physically realizable time-discrete (i.e., digital) feedback systems.

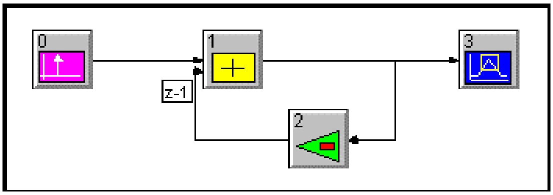


Figure 13.2 The Implicit one-sample delay, in a feedback loop**.**

In any system where the sample rate is large, compared to the inverse of the loop bandwidth, the loop will provide the correct physical response. In some instances, the sample rate may be slow relative to the loop bandwidth; this may produce unexpected results if it is not carefully considered. Likewise the initial conditions associated with the implicit delay can influence the loop behavior.

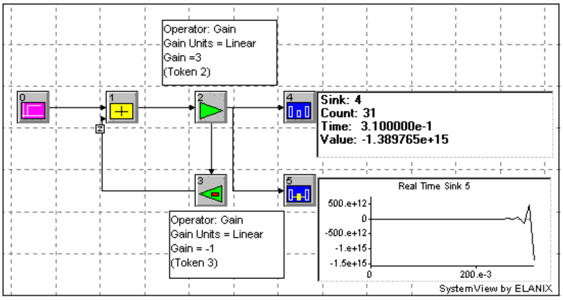
Algebraic loops — Consider the feedback system with feed-forward and feedback paths that contain *fixed constants* with values $G = 3$ and $H = -1$, as shown in figure 13.3. The transfer function in this example is a constant value independent of time (i.e., algebraic loops),

$$\frac{Y(s)}{X(S)} = \frac{G}{1 - G \cdot H} = \frac{3}{4}$$

$$\frac{Y(s)}{X(s)} = \frac{G}{1 - G \cdot H} = \frac{3}{4}$$

Note that this linear system has an infinite bandwidth. Thus, the reciprocal bandwidth is zero, which means that *any* delay will cause instability.

Now, implement this simple loop in SystemView. Drive the loop with a constant step function of unit amplitude, set the Time control to a 100 Hz sample rate, and set the number of samples to 32 and execute the loop. Now examine the output of the loop. The Laplace transfer function illustrated previously, predicts that the output should be a constant value 3/4. But as seen, this is not the case, and in fact the loop is unstable.



Operator: Gain
Gain Units = Linear
Gain =3
(Token 2)

Sink: 4
Count: 31
Time: 3.100000e-1
Value: -1.389765e+15

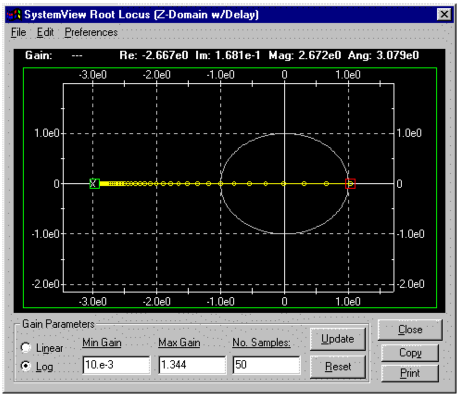
Operator: Gain
Gain Units = Linear
Gain = -1
(Token 3)

Figure 13.3 Feedback / Feed-Forward System

The reason is the 0.01 sec delay (one sample) introduced into the feedback loop as discussed above. This delay, no matter how small, is large with respect to the reciprocal bandwidth (which is zero because the bandwidth is infinite). The actual transfer function of the loop is then,

$$\frac{Y(z)}{X(z)} = \frac{3}{1+3z^{-1}}$$

Note that the transfer function in figure 13.4 has a pole outside the unit circle (z = -3) and is therefore unstable.



Gain: --- Re: -2.667e0 Im: 1.681e-1 Mag: 2.672e0 Ang: 3.079e0

Gain Parameters
○ Linear
● Log
Min Gain: 10.e-3
Max Gain: 1.344
No. Samples: 50

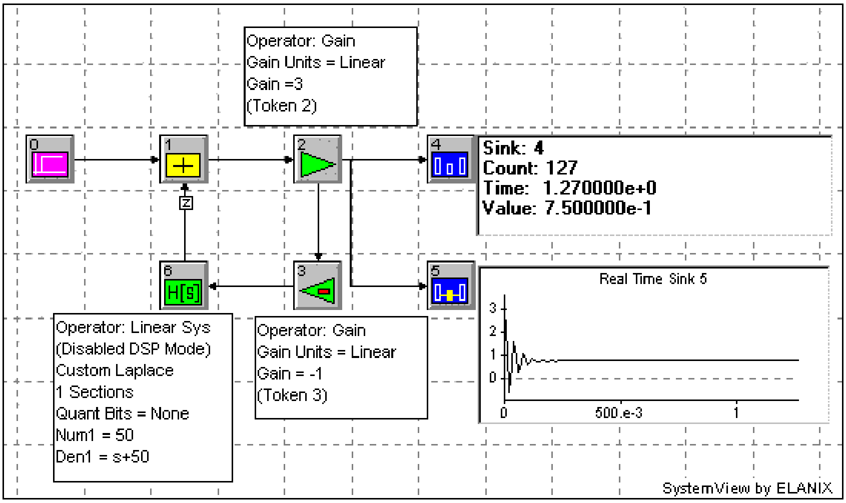
Figure 13.4 Root Locus Plot (Z-Domain w/Delay)

To verify the above assertion, launch an Operator token on the same screen as shown in figure 13.5, defined as a Linear System. Implement the transfer function by entering the above coefficients in the z-domain coefficient windows, and drive this token with the same input. Compare the two outputs and verify that they are identical and unstable.

Creating stable algebraic loops. To force the previous algebraic loop to be stable we must have a finite bandwidth. To do this, simply introduce a pole at a large frequency. That is, multiply *H*(*s*) above by *a/(s+a),* where *1/a* is "large" relative to 0.01 sec (the implicit delay).

$$\frac{Y(S)}{X(S)} = \frac{G}{1+G\cdot H\cdot\dfrac{50}{s+50}}$$

Note that this system has the desired 3/4 gain for frequencies < 8 Hz (~50/2π), and is now stable.



Operator: Gain
Gain Units = Linear
Gain =3
(Token 2)

Sink: 4
Count: 127
Time: 1.270000e+0
Value: 7.500000e-1

Operator: Linear Sys
(Disabled DSP Mode)
Custom Laplace
1 Sections
Quant Bits = None
Num1 = 50
Den1 = s+50

Operator: Gain
Gain Units = Linear
Gain = -1
(Token 3)

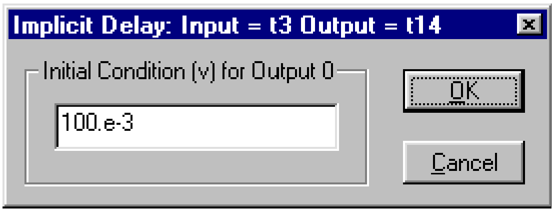
Figure 13.5 Linear feedback System

IIR filter considerations — If a system requires a delay in a feedback path (e.g., an IIR filter), implement the delay with the desired value minus one sample time. Therefore, if the system sample rate is changed, this delay must be adjusted accordingly, or use the Global Parameter Link feature under the Tokens menu, and link the delay to the system dT. A SystemView simulation of a digital system should use the Sample Delay Operator token which has a delay that is always exactly an integral number of samples, that are independent of the system sample rate.

13.2 Initial Conditions for Implicit Delays

When a simulation is initialized, it will zero the value of a feedback sample to the input of the multiple input token in the feedback loop. Ordinarily, this presents no problem, but in some simulations it can cause the simulation to produce an error message or produce incorrect results.   To deal with this, SystemView provides a convenient way to specify the initial conditions for any one or all of the implicit delays that appear in a feedback system.

Click the right mouse button (or double-click the left mouse button) on the implicit delay to open the parameter window shown below.


Implicit Delay: Input = t3 Output = t14

Initial Condition (v) for Output 0

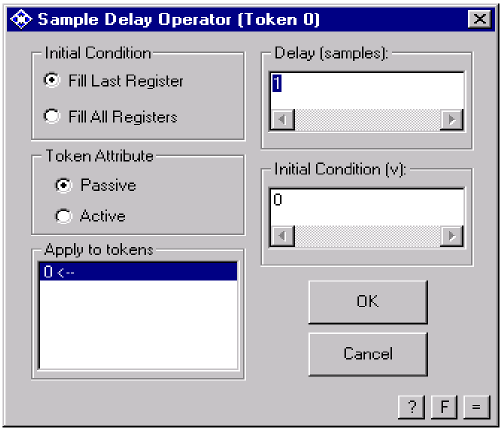
100.e-3

The value you enter here will be the implicit delay output voltage at the start of the system run (i.e., the value at t = 0). It is important to remember that the implicit delay is *not* a SystemView token, but is an implicit computational delay as previously described.

13.3 Using SystemView's sample delay token

The Sample Delay token operates in either the Passive or Active mode. In the Passive mode, the sample delay has two output options. Option one, is an output at a sample of exactly the specified delay. Option two, is an output at a sample of a time that is exactly one sample less than the specified delay. The latter is provided in order to compensate (when required) for the one sample implicit delay present in a feedback loop. The implicit delay indication, [z], will always appear somewhere in a feedback loop when the Passive mode of the Sample Delay is used.

The Active mode has only one output. This output automatically compensates for the one sample implicit delay in a feedback loop. It outputs a sample value exactly one sample less than the specified delay. This feature is used in developing certain IIR structures. Note that the Active mode operates only at one rate for the duration of a simulation. Caution--Any attempt to change the sample rate in any part of a simulation containing a Sample Delay token in the Active mode will lead to incorrect results.



Sample Delay Operator (Token 0)

Initial Condition
● Fill Last Register
○ Fill All Registers

Token Attribute
● Passive
○ Active

Apply to tokens
0 <--

Delay (samples): 1

Initial Condition (v): 0

Choosing the active mode gives the user another means of controlling the placement of implicit delays in feedback loops. The difference in active and passive modes is shown in figure 13.3.

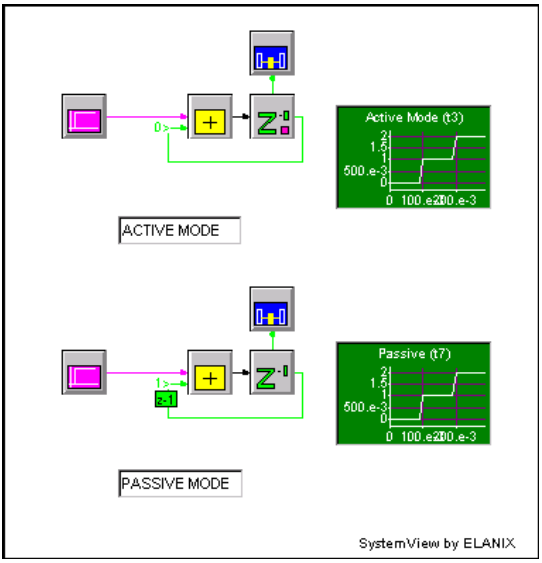


Figure 13.6 Active / Passive mode comparison

In the active mode, the upper channel delay token is clearly marked. “0” marks the delay output connection from the delay token to the adder. In the lower channel, the delay token is in the passive mode.

Note: an implicit delay is placed in the feedback path and that the delay – dt output port is used to compensate for the implicit delay. (Indicated by a “1”on the feedback connection). The overall results are the same. If the active delay token is used in an IIR design, the delay output port would be used for both the feed-forward and feedback paths.

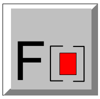
Chapter 14. The Token Libraries

The standard SystemView tokens are organized into the following libraries:

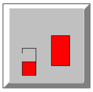
- Function Library
- Operator Library
- MetaSystem I/O Library
- Source Library
- Sink Library
- Token Reservoir (Adder and Multiplier)

The following tables contain an alphabetical list of the standard tokens found in each of the libraries listed above. Unless otherwise indicated, the quantity, x(t), refers to the input to the token. For tokens with multiple inputs, these quantities will have subscripts. The quantity, y(t), refers to the output of a token. For tokens with multiple outputs, these quantities will have subscripts. Control Voltages are defined as c(t) and angle or phase inputs defined by θ(t) or ϕ(t), etc.

14.1 SystemView Function Library



This symbol represents a generic (undefined) Function token.



Name: a^X

Group: Algebraic

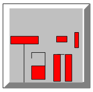
Parameters: (1) Constant a (default = *e)*

Inputs: x(t)

Outputs: y(t)

Description: Raises the specified base (a) to a input power according to,

$$y(t) = a^{x(t)}$$



Name: Arc Tan

Group: Functions

Parameters: (1) Output Gain (G)

Inputs: x(t)

Outputs: y(t)

Description: Produces the inverse tangent of input according to:

$$y(t) = GTan^{-1}(x(t))$$

$$-\pi / 2 \le y(t) \le \pi / 2$$

Name: Arc Tan 4

Group: Functions

Parameters: (1) Select output as modulo or unwrapped; (2) Output Gain

Inputs: $x_1(t), x_2(t)$

Outputs; y(t)

Description: Produces the four-quadrant inverse tangent of the two inputs, with results in radians.

$$y(t) = \mathrm{GTan}^{-1}(x_2(t)/x_1(t))$$

Name: Block

Group: Non-Linear

Parameters: (1) Min Input (v) (2) Max Input (v) (3) Output Gain (G)

Inputs: x(t)

Outputs: y(t)

Description: Produces output according to,

$$\begin{aligned} y(t) &= G,\ x_{\min} \le x(t) \le x_{\max} \\ &= 0, \textit{Otherwise} \end{aligned}$$

Where $x_{\min}$, *and* $x_{\max}$ are the specified min and max input threshold levels.

Name: Coulomb

Group: Non-Linear

Parameters: (1) Slope; (2) +Y-intercept

Inputs: x(t)

Outputs: y(t)

Description: Produces its output according to

$$y(t) = a \cdot x(t) + b \cdot sign(x(t))$$

Where *a,* is the specified slope and *b,* is the +y-intercept.

Name: Cmltv Avg.

Group: Functions

Parameters: (1) Gain (G)

Inputs: x(t)

Outputs: y(t)

Description: Produces the cumulative average of the input according to,

$$y(t) = G \cdot \frac{1}{t} \int_0^t x(\alpha) d\alpha$$

Name: Crt-Plr

Group: Complex

Parameters: None

Inputs: $x_1(t)$, $x_2(t)$

Outputs: r(t), θ(t).

Description: Converts Cartesian coordinates to Polar coordinates in radians.

$$r = \sqrt{x_1^2 + x_2^2} \quad \theta = \tan^{-1}\left(\frac{x_2}{x_1}\right)$$

Name: Cx Add

Group: Complex

Parameters: None

Inputs: $x_{1r}(t)$, $x_{1i}(t)$, $x_{2r}(t)$, $x_{2i}(t)$

Outputs: $y_r(t)$, $y_i(t)$

Description: Performs complex addition:

$$(y_r, y_i) = (x_{1r} + x_{2r}, x_{1i} + x_{2i})$$

Name: Cx Multiply

Group: Complex

Parameters: (1) Multiple Type: Conjugate or Normal

Inputs: x_{1r} (t), x_{1i} (t), x_{2r} (t), x_{2i} (t)

Outputs: y_r (t), yi (t)

Description: Performs complex multiply or conjugate multiply.

$$(y_r, y_i) = \quad (x_{r1}x_{r2} - x_{i1}x_{i2}, x_{r1}x_{i2} + x_{i1}x_{r2}) \text{ Normal}$$

$$(x_{r1}x_{r2} + x_{i1}x_{i2}, -\ x_{r1}x_{i2} + x_{i1}x_{r2}) \text{ Conjugate}$$

Name: Cx Rotate

Group: Complex

Parameters: (1) Phase Gain (G); (2) Phase Offset in degrees (α)

Gain of 1 = 2PI radians/V

Inputs: x_1(t), x_2 (t), θ(t)

Outputs: y_1(t), y_2(t)

Description: This token performs the rotation:

$$y_1(t) = x_1(t)\cos\,(G\theta(t)+\alpha) - x_2(t)\sin\,(G\theta(t)+\alpha)$$

$$y_2(t) = x_2(t)\cos\,(G\theta(t)+\ \alpha) + x_1(t)\sin\,(G\theta(t)+\ \alpha)$$

Note: x_1(t), x_2(t), and $\theta(t)$ are required inputs.

Name: Custom

Group: Functions

Parameters: (1) Number of Inputs; (2) Algebraic Expression

Description: Produces output based on user-defined expression in terms of inputs, such as p (0), p (1)… and system variables such as current time (ct), current sample (cs), etc.

Name: Dead Band

Group: Non-Linear

Parameters: (1) dead band limit

Inputs: x(t)

Outputs, y(t)

Description: Produces output according to,

$$\begin{aligned} y(t) &= \max\bigl(0, x(t) - z\bigr),\ x(t) \geq 0 \\ &= \min\bigl(0, x(t) + z\bigr),\ x(t) < 0 \end{aligned}$$

Where *z* is the specified positive dead band limit.

Name: Divide

Group: Algebraic

Parameters: (1) Output Gain, G

Inputs: $x_1(t)$, $x_2(t)$

Outputs: y(t)

Description: Produces its output according to,

$$y(t) = x_1(t) / x_2(t),\ x_2(t) \neq 0,$$

$$= x_1(t - \Delta t) / x_2(t - \Delta t) x_1(t),\ x_2(t) = 0$$

Where $x_1(t)$ and $x_2(t)$ are the specified numerator and denominator.

Name: Extract

Group: Multiplex

Parameters: (1) Threshold (v)

Inputs: x(t)

Outputs: y(t)

Description: When the control threshold goes high, the token allows the data to pass.

Name: Freq. Mod

Group: Phase / Frequency

Parameters: (1) Carrier amplitude (v); (2) Carrier freq. (Hz); (3) Carrier phase (deg); (4) Modulation Gain Output Selection: Quadrature (sin), and In-Phase (Cos)

Inputs: x(t)

Outputs: $y_1(t)$, y_2(t)

Description: Freq. modulates a carrier with input signal according to,

$$y_1(t) = A\sin\left\{2\pi\left[f_c t + G\int_{t_{start}}^{t} x(\alpha)d\alpha\right] + \theta\right\}$$

$$y_2(t) = A\cos\left\{2\pi\left[f_c t + G\int_{t_{start}}^{t} x(\alpha)d\alpha\right] + \theta\right\}$$

Where *A* is the specified amplitude, f_c is the carrier frequency, *G* is the modulation gain, and θ is the carrier phase offset.

Name: Gen Mux

Group: Multiplex

Parameters: (1) Slot time (optional); (2) Number of Segments, up to 20.

Inputs: x(t)

Outputs: y(t)

Description: This allows the multiplexing of frames of data. With the Gen Mux, the user can determine the number of input channels, (up to 20), and the number of samples for each channel to be multiplexed.

Name: Gen De-Mux

Group: Multiplex

Parameters: (1) Slot time (optional); (2) Number of Segments, up to 20; (3) Slot Boundary Offset.

Inputs: x(t)

Outputs: y(t)

Description: The generalized de-multiplexor token can reverse the operation of the multiplexor by extracting samples from an input frame. The number of output channels (up to 20, is defined by segments, and the number of bits being sent to each output channel is by segment sample entry. The “slot boundary offset entry allows starting the operation after a specified number of samples.

Name: Half Rectify

Group: Non-Linear

Parameters: (1) Zero point

Inputs: x(t)

Outputs: y(t)

Description: Half-wave rectifies the input according to,

$$\begin{aligned} y(t) &= x(t) - z,\ x(t) \geq z \\ &= 0,\ elsewhere \end{aligned}$$

Were *z* is the specified zero point.

Name: Hysteresis

Group: Non-Linear

Parameters: (1) Bandwidth; (2) Backlash; (3) Slope

Inputs: x(t)

Outputs: y(t)

Description: Provides a hysteresis transfer function with characteristics specified by the bandwidth and gain parameters. Very small bandwidths (relative to the system sample rate) result in a 'sluggish' hysteresis (i.e., a lowpass filter), whereas large bandwidths (e.g., ~0.3 x sample rate) result in a 'stiff' hysteresis.

Name: Limit

Group: Non-Linear

Parameters: (1) Input max. (⌖); (2) Output max. (⌖)

Inputs: x(t)

Outputs: y(t)

Description: The output is limited to ± the Output Maximum when the input exceeds ± the Input Maximum.

$$\begin{aligned} y(t) &= \left(\frac{OutMax}{InMax}\right) x\ (t\ ),\ 0 < |x(t)| \le InMax \\ &= OutMax \cdot sign(x(t)),\ \ |x(t)| > InMax \end{aligned}$$

Hard limiting is accomplished by setting InMax = 0.

Name: Log

Group: Functions

Parameters: (1) Log base (default is *e*)

Inputs: x(t)

Output: y(t)

Description: Produces the logarithm of the input using the specified base.

$$y\,(t) = \mathrm{Log}_{\,b}\,(x\,(t))$$

Name: 2ch Multiplex

Group: Multiplex

Parameters: (1) No. Samples from input A (N_A); (2) No. Samples from input B (N_B)

Inputs: x_1 (t), x_2 (t)

Outputs: y (t)

Description: Multiplexes (interleaves) the two token inputs by taking N_1 samples from input 1, followed by N_2 samples from input 2. The token output rate is R_{out} times the maximum input rate, where

$$R_{out} = \frac{(N_1 + N_2)}{T} = R_1 + R_2$$

$$\textit{Where } R_1 = N_1 / T \textit{ and } R_2 = N_2 / T$$

Note: The sum of input rates 1 and 2 cannot exceed the system rate.

Name: Phase Mod

Group: Phase / Frequency

Parameters: (1) Carrier amp (v); (2) Carrier freq. (Hz); (3) Carrier phase (deg); (4) Mod Gain Output Selection: Quad (sin), In-Phase (Cos)

Inputs: x(t)

Outputs: y_1(t), y_2(t)

Description: This token Phase modulates a carrier with the input signal according to, $y_1(t) = A\sin\left\{2\pi\left[f_c t + G\cdot x(t)\right] + \theta\right\}$

Or

$$y_2(t) = A\cos\left\{2\pi\left[f_c t + G\cdot x(t)\right] + \theta\right\}\cdot$$

Where A is the specified amplitude, f_c is the carrier frequency, G is the modulation gain, and θ is the carrier phase offset.

Name: Plr-Crt

Group: Complex

Parameters: None

Inputs: x(t), ϕ(t)

Outputs: y_r(t), y_i(t)

Description: Converts polar (magnitude and phase) inputs to Cartesian (real and imaginary) coordinates, where *x (t)* is the magnitude input and $\phi(t)$ is the phase input in radians. According to,

$$y_{\mathbf{r}}(t) = x(t)\cos(\phi(t))$$
$$y_{\mathbf{i}}(t) = x(t)\sin(\phi(t))$$

Name: Polynomial

Group: Algebraic

Parameters: (1) Polynomial Coefficients (fifth order, six coefficients)

Inputs: x(t)

Outputs: y(t)

Description: Creates a fifth-order polynomial function of the input using the specified a_i,

$$y(t) = a_5 x^5(t) + a_4 x^4(t) + a_3 x^3(t) + \\ a_2 x^2(t) + a_1 x(t) + a_0$$

Name: Quantize

Group: Non-linear

Parameters: (1) Number of binary bits; (2) Input max (±); (3) Floating point or signed integer output

Inputs: x(t)

Outputs: y(t)

Description: Quantizes the input amplitude, x (t), to the specified number of levels,

$$\text{levels} = 2^{\text{bits}}$$

Saturating at the specified input maximum. Zero is always an output level, so there is one less positive level than negative when a signed integer is selected; the output is:

$$-2^{n-1} \leq y\ (t) \leq Z^{n-1} - 1$$

Name: Rectify

Group: Non-Linear

Parameters: (1) Zero point

Inputs: x(t)

Outputs: y(t)

Description: Rectifies the input according to

$$y(t) = \left| x(t) - z \right|$$

Were *z* is the specified zero point.

Name: Sigmoid

Group: Functions

Parameters: (1) Shape factor

Inputs: x(t)

Outputs: y(t)

Description: The sigmoid transfer function is defined as,

$$y(t) = \frac{1}{1 + \exp\left(-2\beta x(t)\right)}$$

Where β is the specified shape factor.

Name: Sine

Group: Functions

Parameters: (1) Phase offset (deg)

Inputs: x(t)

Outputs: y_1(t), y_2(t)

Description: This generates two sinusoidal functions defined as,

$$y_1(t) = \sin(x(t) + \theta)$$
$$y_2(t) = \cos(x(t) + \theta)$$

Were, θ is the specified phase offset in degrees.

Name: Tangent

Group: Functions

Parameters: (1) Phase offset (deg)

Inputs: x(t)

Outputs, y(t)

Description: This produces the tangent of the input according to

$$y(t) = \tan(x(t) + \theta)$$

Were, θ is the specified phase offset in degrees.

Name: Tanh

Group: Functions

Parameters: (1) Shape factor

Inputs: x(t)

Outputs; y(t)

Description: The hyperbolic tangent transfer function is defined as,

$$y(t) = \frac{1 - \exp\left(-2\beta x(t)\right)}{1 + \exp\left(-2\beta x(t)\right)}$$

Were β is the specified shape factor.

Name: Vector Fct

Group: Algebraic

Parameters: (1) Select output as Average, Order Statistics, Modulus, or Geometric Mean; (2) Output gain or Percentile.

Inputs: $x_{i(t)}$

Outputs; y(t)

Description: Produces the output according to:

$$\begin{aligned} y(t) &= \frac{G}{N}\sum_{i=0}^{N-1} x_i(t),\ \textit{Average} \\ &= x_i(t), \textit{the input having the rank } OS \\ &= G\sqrt{\sum_{i=0}^{N-1} x_i^2(t)},\ \textit{Modulus} \\ &= G\left(\prod_{i=0}^{N-1} x_i(t)\right)^{\frac{1}{N}},\ \textit{Geometric Mean} \end{aligned}$$

Name: X^a

Group: Algebraic

Parameters: (1) Exponent value

Inputs: x(t)

Outputs; y(t)

Description: Raises the input to the specified power,

$$y(t) = x^a(t)$$

Name: Xtrnl Fct

Group: Non-Linear

Parameters: (1) File name for the transfer function.

Inputs: x(t)

Outputs: y(t)

Description: Produces a custom transfer function that you specify in an external file. The file data must be text, in one of the following formats:

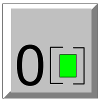
X, Y pairs (no header), separated by a space, tab, or comma. The X's should be in ascending order as shown:

```
-3, 14.2
0, 18.7
2.2, 16.4
```

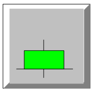
Equally spaced data (with header). “Samples” refers to the number of values in the table. Xmin and Xmax define the input ranges that are valid for the output values. The maximum number of samples is 8.192.

```
Samples = 3,

Xmin = -10,

Xmax = 29

-11.3

-13.1
```

14.2 SystemView Operator Library



This symbol represents a generic (undefined) Operator token.



Name: Average

Group: Filters / Systems

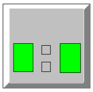
Parameters: (1) Time window (T sec)

Inputs: x(t)

Outputs: y(t).

Description: Outputs the moving average of the input,

$$y(t) = \frac{1}{T}\int_{t-T}^{t} x(\alpha)d\alpha$$



Name: Compare

Group: Logic

Parameters: (1) True Output (v); (2) False Output (v); (3) Comparison

Inputs: x_0(t), x_1(t)

Outputs: y(t).

Description: Compares the two inputs

$$x_0(t)\ vs.\ x_1(t)$$

According to selected comparison rule: (i.e., =, <>, <=, <, >=, >).

The output = 1 if compare is True, and output = 0 if compare is False.

Name: Decimate

Group: Sample / Hold

Parameters: (1) Decimation factor (integer)

Inputs: x(t)

Outputs: y(t)

Description: Down-samples the input signal by the specified factor,

$$y_n = x_n \text{, } n \bmod(N) = 0$$

Where *N* is the decimation factor.

Name: Delay

Group: Delays

Parameters: (1) Time delay (sec); (2) Select delay type, Non-Interpolating or Interpolating (Linear).

Inputs: x(t)

Outputs: y(t)

Description: The input is delayed according to,

$$y(t) = x(t - \tau)$$

Where *t* is the specified delay. Will linear interpolate to create delayed output if this option is selected. This time-active token has two outputs, the first is the delay specified by the input parameter. The second, (labeled τ - dT in the connection dialog), is delayed by one System Time interval less than the specified delay. This second output compensates for the inherent one-system-tick delay in a feedback loop when adjusting the feedback delay for a precise value.

Name: Derivative

Group: Integral / Differential

Parameters: (1) Gain (G)

Inputs: x(t)

Outputs: y(t)

Description: Forms the derivative of the input,

$$y(t) = G \cdot \frac{d}{dt} x(t), \quad t > t_{start}$$

Where t_{start} = simulation start time.

Name: Digital Scale

Group: Gain / Scale

Parameters: (1) Input word size (bits) (2) Retained significant bits

Inputs: x(t)

Outputs, y(t)

Description: Extracts the specified number of significant bits from the input word (sample). The input must be integer valued. For example, with integer 13, (1101) b, input and 2 as the specified retained bits, the output is 3, (11) b.

Name: FFT

Group: Filters / Systems

Parameters: (1) Select FFT direction, Forward / Inverse; (2) FFT size (samples)

Inputs: x(t)

Outputs: y(t).

Description: This token performs sequential FFTs on the input. Output begins one sample after the FFT size, for an FFT size of 32; a minimum of 64 samples is needed, and is continuous thereafter. The DC component of the output spectrum will occur at sample number [(FFT size) + (FFT size/2)] (e.g., for an FFT size of 32, a DC signal will occur at sample 48). Given a frequency resolution that is an integer multiple of the frequency of interest, then each sample represents a (F_s/FFT size) Hz step where F_s is the overall sampling rate. (e.g., if F_s is 16Hz and the FFT size is 32, then each sample represents a 0.5Hz step). The magnitude of the FFT peak is the amplitude of the signal times half the FFT size. The user can select from one to five outputs, the real or imaginary component, the magnitude, the phase, or phase unwrapped.

Name: Fraction

Group: Gain / Scale

Parameters: (1) Retain the Fractional and/or Integer parts; (2) Output Gain.

Inputs: x(t)

Outputs: y(t)

Description: This token extracts the fractional, or integer part of the input (e.g., for the fractional option, *G* is the specified gain. An input of 3.01 becomes *G* x 0.01.)

Name: Gain

Group: Gain / Scale

Parameters: (1) Gain value; (2) Select Gain Units, Linear or dB power

Inputs: x(t)

Outputs: y(t)

Description: This token multiplies the input by the specified value. One of two outputs can be selected; the first is linear, given by

$$y(t) = G \cdot x(t)$$

The second, in dB, is given by

$$G = 10^{G_{dB}/20}$$

Name: Hold

Group: Sample / Hold

Parameters: (1) Select hold last value or hold zero between samples; (2) Gain

Inputs: x(t)

Output: y(t)

Description: Used to return signal input to the system sample rate (e.g., the time spacing between samples is brought to $1/F_s$, where F_s is the overall sampling rate defined in the System Time Specification Window). This token is used after a sampler or decimator, to return the data to the sampling rate specified in the system window.

Name: Integral

Group: Integral / Differential

Parameters:(1) Integration Order (zero or first); (2) Initial condition, default is zero.

Inputs: x(t)

Outputs: y(t)

Description: Forms the integral of the input,

$$y(t) = \int_{t_{start}}^{t} x(\alpha)d\alpha + I_0, t > t_{start}$$

Where t_{start} = simulation start time.

Name: Linear Sys Filters

Group: Filters / Systems

Parameters: Several, depending on selection of Linear System type.

Inputs: x(t)

Outputs: y(t)

Description: FIR, IIR, Laplace, and analog filter system design and specification. Chapter 6, Filters and Linear Systems, contains detailed information regarding this token.

Name: Pulse

Group: Logic

Parameters: (1) Threshold (v); (2) True Output (v); (3) False Output (v); (4) Pulse Width (s)

Inputs: x(t)

Outputs: y(t)

Description: This token produces an output pulse with an amplitude that is equal to either of the following: the amplitude equal to the False Output parameters as a default condition, or the True Output based on the parameters for the specified duration (Pulse Width) each time the input crosses the Threshold parameter (from a low to high level).

Name: P I D (Proportional Integrator Differentiator)

Group: Integral / Differential

Parameters: (1) Proportional Gain; (2) Integral Gain (v); (3) Derivative Gain

Inputs: x(t)

Outputs: y(t).

Description: This token produces its output according to:

$$y(t)=G_P x(t)+G_I\int_0^t x(\alpha)d\alpha+G_D\frac{d}{dt}x(t)$$

Name: Max-Min

Group: Logic

Parameters: (1) Output Gain; (2) Output Offset

Inputs: x(t)

Outputs: y(t)

Description: This token can accommodate up to 19 inputs. At each sample instant, the token selects the maximum and minimum among the inputs according to:

Y (0) = input port number with maximum data value*Gain + Offset

Y (1) = input port number with minimum data value*Gain + Offset

Name: Modulo

Group: Gain / Scale

Parameters: (1) Modulus base

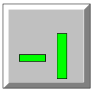
Inputs: x(t)

Outputs: y(t)

Description: This token performs the modulo operation,

$$y(t) = x(t) - bas \cdot Int\{x(t)/bas\}, x(t) \geq 0$$

$$y(t) = bas + x(t) - bas \cdot Int\{x(t)/bas\}, x(t) < 0$$

**Name**: Negate

Group: Gain / Scale

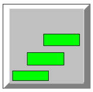
Parameters: None

Inputs: x(t)

Outputs: y(t)

Description: This token forms the negative of the input,

$$y(t) = -x(t)$$

**Name**: OSF

Group: Filters / Systems

Parameters: (1) Time window (sec). (2) Output rank (%)

Inputs: x(t)

Outputs: y(t)

Description: Performs Order Statistics Filtering (OSF). The output is that input sample, having the specified rank within the current time window, (e.g., Rank = 50% is the median filter, Rank = 100% outputs the maximum value in the window).

Name: Peak-Hold

Group: Sample / Hold

Parameters: (1) Hold Zero or Last Peak; (2) Reset Threshold (v)

Inputs: x(t)

Outputs: y(t)

Description: Outputs the maximum value, minimum value, maximum location and minimum value depending on what option was picked, the Hold Last Peak or Hold Zero.

Name: ReSample

Group: Sample / Hold

Parameters: (1) Sample Rate (Hz)

Inputs: x(t)

Outputs: y(t)

Description: This token performs a sampling of input data at a rate other than the system rate. The input data has been sampled or decimated prior to the token input. The output samples are at a rate typically less than the system rate. The token operates by holding the last value of the input data and then samples the result at the specified rate. The token does not interpolate the sampled data.

Name: Sampler

Group: Sample / Hold

Parameters: (1) Sample Rate (Hz); (2) Sample Aperture duration (s); (3) Aperture Jitter (s); (4) Select: Interpolating Linear, Non-Interpolating, Non-Interp Look Right, and Non-Interp Look Left.

Inputs: x(t)

Outputs: y(t)

Description: This token performs a true sampling operation at the specified rate (interpolation is performed if selected). The output sample is a linear combination of the input samples falling within the sample time aperture. The aperture start time is uniformly distributed over the aperture jitter time.

Name: Sample Hold

Group: Sample / Hold

Parameters: (1) Control Threshold: Input Signal, Control Signal

Inputs: x(t), c(t).

Outputs: y(t)

Description: This token samples, tracks and holds. When the Control signal is above threshold, the Output tracks the Input on a per sample basis. When the Control signal is below threshold, the Output is held at the last sampled value when the Control signal was above threshold.

Name: Select

Group: Logic

Parameters: (1) Threshold (v)

Inputs: x(t)

Outputs: y_0 , y_1

Description: This token has two outputs. Each one selects a value based on the control signal (input 1) and the input signal (input 0) according to: y_0 = signal, for control => threshold = 0 Otherwise, y_1 = 0, for control => threshold = signal. Otherwise y_0 and y_1 are complementary.

Name: Switch

Group: Logic

Parameters: (1) Min threshold (v); (2) Max threshold (v)

Inputs: $X_{\text{n (t), n up to 20 inputs}}$

Outputs: y(t)

Description: One of the inputs *n* (0<=*n*<=*N*-1) is selected at time *t* depending on the control signal *c(t)*, y(t) = $X_{n\ selected}$ (t)

$$n_{selected} = Int\left[\frac{N(c(t) - C_{min})}{C_{max} - C_{min}}\right]$$

Where *N* (N<20) is the number of inputs (excluding control). The switch can be used as a commutator. If the control voltage has a saw tooth waveform with appropriate threshold settings, then the switch consecutively selects each of the n inputs. After completing selection (through Nth switch), the cycle repeats. Thus the inputs on each of the N data inputs are sampled and multiplexed.

Name: AND

Group: Logic

Parameters: (1) Threshold (v); (2) TRUE output; (3) FALSE output

Inputs: Up to 20.

Outputs: y(t)

Description: Performs the logical AND operation on the inputs,

$$\begin{aligned} y(t) &= T,\ \ x_i(t) \geq Thr \ \ all\, i \\ &= F \ \ Otherwise \end{aligned}$$

Name: OR

Group: Logic

Parameters: (1) Threshold (v); (2) TRUE output; (3) FALSE output

Inputs: Up to 20

Outputs; y(t)

Description: Performs the logical OR operation on the inputs,

$$\begin{aligned} y(t) &= T,\ \ x_i(t) \geq Thr \ \ any\, i \\ &= F \ \ Otherwise \end{aligned}$$

Name: XOR

Group: Logic

Parameters: (1) Threshold, (2) TRUE output, (3) FALSE output

Inputs: Up to 20

Outputs: y(t)

Description: Performs the logical XOR (Exclusive Or) operation on inputs,

$$\begin{aligned} y(t) &= T, x(t) \geq Thr, N_i odd \\ y(t) &= F, x(t) \geq Thr, N_i even \end{aligned}$$

Where N_i is the number of inputs exceeding threshold *Thr*.

Name: NAND

Group: Logic

Parameters: (1) Threshold, (2) TRUE output, (3) FALSE output

Inputs: Up to 20

Outputs: y(t)

Description: Performs the logical NAND operation on the inputs,

$$\begin{aligned} y(t) &= T,\ \ x_i(t) < Thr \ \ all\, i \\ &= F \ \ Otherwise \end{aligned}$$

Name: NOT

Group: Logic

Parameters: (1) Threshold, (2) TRUE output, (3) FALSE output

Inputs: x(t)

Outputs: y(t)

Description: Performs the logical NOT operation on the input,

$$\begin{gathered} y(t) = T, x(t) < Thr \\ y(t) = F, otherwise \end{gathered}$$

Name: Sample Delay

Group: Delays

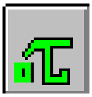
Parameters: (1) Delay samples, Active or Passive mode.

Inputs: x(t)

Outputs: y(t)

Description: This token delays the input by the specified number of *samples*. Unlike the Delay Operator, the sample delay has a delay *time* that is dependent on the sample rate provided by the input token data, usually the system sample rate F_s. In multi-rate systems, the input sample rate can be any value that is less than the system sample rate. Note: the sample delay works as follows. After the delay buffers are filled, a delayed sample is released from the delay token for every new sample input to the delay token.

The delay is implemented as $y_n = x_{n-k}$ where *k* is the specified sample delay. The sample delay token has two output ports, delay, as well as delay – dt. Where dt = $1/F_s$. The delay – dt port should be used to compensate for implicit delays that are placed in feedback loops.

**Name**: Variable Delay

Group: Delays

Parameters: (1) Delay Type: Non-interpolating or Interpolating; (2) Min Delay (s), $\tau_{\mathrm{min;}}$ (3) Max Delay (s), $\tau_{\mathrm{max,;}}$ (4) Min Control (v), Cmin; (5) Max Control (v), Cmax.

Inputs: x(t)

Outputs: y(t)

Description: Delays input according to,

$$y = X(t-\tau)$$

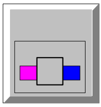
$$\tau = \frac{(\tau_{\max} - \tau_{\min})}{(C_{\max} - C_{\min})} \cdot (C_{in} - C_{\min}) + \tau_{\min}$$

$$\tau = \tau_{\max}, \ C_{in} \geq C_{\max}$$

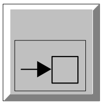
$$\tau = \tau_{\min}, \ C_{in} \leq C_{\min}$$

Where $t_{\min}$ and $t_{\max}$ are the specified delays chosen by providing the appropriate value to the token control input and C_{in} is the input control voltage. The token will interpolate to create the delayed output if this option is selected.

14.3 SystemView MetaSystem I/O Library

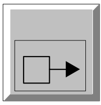

This symbol represents a generic (undefined) MetaSystem I/O.

Each MetaSystem requires these tokens as input and output connections to the next higher-level connection in a simulation. Refer to chapter 7 for details.


Name: Input

Group: Meta System I/O

Description: Provides an input port to a MetaSystem.


Name: Output

Group: Meta System I/O

Description: Provides an output port to a MetaSystem.

14.4 *SystemView* Source Library

This symbol represents a generic (undefined) Source token.

Source tokens generate the waveforms or input data needed to drive a SystemView simulation. Source tokens have no input connections. Several source tokens, however, have multiple outputs. For example, the sinusoid token has two outputs in quadrature. This provides a convenient way for generating quadrature components of a complex, baseband signal.

Name: Custom

Group: Aperiodic

Parameters: (1) Number of Output Ports; (2) P(n)=expression where n is the number of the nth port and P(x) is the mathematical relationship associated with that port. The number of output ports is not limited.

Description: A custom source allows the definition of one or more mathematical descriptions of up to N sources in terms of system variables and parameters. For example:

P (0)=sin (2*pi*ct)

Name: Ext 1ch

Group: Import

Parameters: (1) File name; (2) Data format selected from: 8-bit Integer, 16-bit integer, IEEE single or double precision, Binary, or ASCII text.

Description: A 1-channel (Monaural) source from an external file, allows SystemView to process any data desired. The file format may be an 8-bit integer (0-255), 16-bit integer, and IEEE single or double precision, Binary or ASCII in a standard text format.

Name: Ext 2ch

Group: Import

Parameters: (1) File name; (2) Data format selected from: 8-bit Integer, 16-bit integer, IEEE single or double precision, Binary, or ASCII text.

Description: A 2-channel (Stereo) source from an external file, allows SystemView to process any data desired. The file format may be an 8-bit integer (0-255), 16-bit integer, and IEEE single or double precision, Binary or ASCII in a standard text format.

Name: Freq. Sweep

Group: Periodic

Parameters: (1) Amplitude (v); (2) Start freq. (Hz); (3) Stop freq. (Hz); (4) Period (sec); and (5) Phase (deg.)

Description: This token generates a swept frequency sinusoid (chirp) according to,

$$y(t) = si2\pi \int_{sta} t + \pi R(tm(t)^2 + \theta)$$

$$R = \frac{\int sto \int sta}{T}$$

Name: Gauss Noise

Group: Noise / PN

Parameters: (1) Std. Deviation (or power spectral density); (2) Mean; (3) Constant Parameter, Std. Deviation, 1 ohm density, 50 ohm density

Description: This token generates a random signal having a Gaussian (Normal) distribution (or adjusted to power density).

Name: Impulse

Group: Aperiodic

Parameters: 1) Gain (v); (2) Start time (s); (3) Voltage Offset (v)

Description: This token generates a single, *unit area*, (1/dt by dt) pulse at the specified time t according to:

$$y(t) = G \cdot \delta\left(t - t_{start}\right) + offset \, .$$

If the offset is non-zero and the start time is different than zero, then both the step and impulse response of a system may be observed.

Name: PN Sequence

Group: Noise / PN

Parameters: (1) Amplitude (v); (2) Symbol rate (Hz); (3) Number of levels, N; (4) Offset (v); and (5) Phase (deg)

Description: This token generates a pseudo-random (PN) sequence of multilevel pulses at the specified rate. The token generates output levels in the voltage range of (-A to A). The voltage levels are separated by 2A/(N-1). Use the PN Generator in the Communications Library for maximal length PN sequences.

Name: PSK Carrier

Group: Periodic

Parameters: (1) Amplitude (v); (2) Frequency (Hz); (3) Carrier phase (deg); (4) Symbol rate (Hz); (5) Number of symbols.

Description: This token generates an m-*ary* phase modulated carrier,

$$y(t) = A \cdot Sin\left(2\pi f_c t + \phi_T(t) + \theta\right)$$

Where $\phi_T(t)$ is a PN sequence of m-*ary* phase values (between 0 and 2π), *T* is the specified symbol period (1/symbol rate), θ is the carrier phase offset, and *A* is the carrier amplitude.

Name: Pulse Train

Group: Periodic

Parameters: (1) Amplitude (v); (2) Frequency (Hz); (3) Pulse width (s); (4) Offset; (5) phase (deg); (6) Square wave option.

Description: This token generates a periodic sequence of pulses having the specified amplitude as measured relative to zero volts. The pulse width determines the duration of time, during a pulse period, that the signal remains at amplitude, A. The pulse width must be less than the reciprocal of the pulse frequency. A square wave is generated when the pulse width is 1/2 the pulse period.

Click the Square wave button to automatically convert to a square wave. When Square wave is selected, SystemView ignores the pulse width setting and automatically sets the offset to 1/2 the amplitude.

Name: Sawtooth

Group: Periodic

Parameters: (1) Amplitude (v); (2) Frequency (Hz); (3) Offset (v); and (4) Phase (deg)

Description: This token generates a periodic sawtooth waveform.

Name: Sinusoid

Group: Periodic

Parameters: (1) Amplitude (v); (2) Frequency (Hz); (3) Phase (deg)

Description: This token generates two, quadrature sinusoid outputs:

$$y_1(t) = A \cdot \sin(2\pi f_c(t) + \theta)$$

$$y_2(t) = A \cdot \cos(2\pi f_c(t) + \theta)$$

Name: Step Function

Group: Aperiodic

Parameters: (1) Amplitude (v); (2) Start time (s); (3) Offset (v)

Description: Generates a step function. Note that setting the bias equal to the negative of the amplitude may create a single pulse or an impulse.

Name: Time

Group: Aperiodic

Parameters: (1) Gain (v); (2) Offset (v)

Description: Generates a scaled replica of the system time,

$$y(t) = G \cdot t + Offset$$

Where *t* is the defined system time.

Name: Thermal Noise

Group: Noise / PN

Parameters: (1) Resistance (ohms); (2) Temperature (K)

Description: Generates noise whose distribution is related to:

$$\sqrt{4kTR}$$

K = Boltzman's Constant, R = Resistance, and T = Temperature

Name: Uniform Noise

Group: Noise / PN

Parameters: 1) Minimum value; (2) Maximum value

Description: Generates noise with amplitude values that are uniformly distributed between the specified maximum and minimum.

Name: WAV 1ch

Group: Import

Parameters: File Name

Description: Generates a monaural (1 channel) audio signal from a special Windows WAV-formatted disk file. You specify the file name and SystemView automatically displays the format (8 or 16 bit, and the data rate in Hz). Click the right mouse button on the Source and select Play Audio.

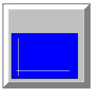
Name: WAV 2ch

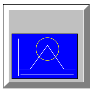
Group: Import

Parameters: File Name

Description: Generates a stereo (2 channel) audio signal from a special Windows WAV-formatted disk file. You specify the file name and SystemView automatically displays the format (8 or 16 bit, and the data rate in Hz). Click the right mouse button on the Source and select Play Audio.

14.5 SystemView Sink Library

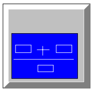

This symbol represents a generic (undefined) Sink token.


Name: Analysis

Group: Analysis

Parameters: None

Description: The basic SystemView sink. No signals are displayed on the System screen. *To view and analyze the sink data, click the Analysis button located at the top of the System screen.* The Analysis window opens for detailed signal analysis. See chapter 9 for a detailed discussion of this window and how sink data is displayed.


Name: Averaging

Group: Analysis

Parameters: None

Description: This sink averages the input waveform. When used with the System Loop time parameter, the Averaging sink computes the cumulative average waveform over all system loops. The output may be observed in the Analysis window.

Name: Current Value

Group: Numeric

Parameters: None

Description: This sink displays the current system time and Sink input value in real-time as the system executes. The output may be observed in the Analysis window.

Name: Data List

Group: Numeric

Parameters: None

Description: This token generates a list of the data received by the sink and displays the list on the System screen. Press the Ctrl key and drag the mouse to enlarge the data list display. The output may be observed in the Analysis window.

Name: Ext 1ch

Group: Export

Parameters: (1) File name; (2) Data format selected from: 8-bit Integer, 16-bit integer, IEEE single or double precision, Binary, or ASCII text.

Description: This token is used to record the results of a SystemView simulation for use in other applications. For example, single channel monaural test data, generated by SystemView is saved in such files. The SystemView output is sent to the disk file defined by the user. The user specifies the file name and format (1 bit, 8 or 16 bit integer, IEEE single or double precision, or ASCII text). Output may also be observed in the Analysis window.

Name: Ext 2ch

Group: Export

Parameters: (1) File name; (2) Data format selected from: 8-bit Integer, 16-bit integer, IEEE single or double precision, Binary, or ASCII text.

Description: This token is used to record the results of a SystemView simulation for use in other applications. For example, two-channel Stereo test data, generated by SystemView is saved in such files. The SystemView output is sent to the disk file defined by the user. The user specifies the file name and format (1 bit, 8 or 16 bit integer, IEEE single or double precision, or ASCII text). Output may also be observed in the Analysis window.

Name: Final Value

Group: Numeric

Parameters: None

Description: This token will display the final value received at the end of each system loop upon completion. *This Sink retains only one sample per system loop*. The output is presented numerically, and may be observed in the Analysis window.

Name: Real Time

Group: Graphic

Parameters: None

Description: Displays a plot of the Sink input value in real-time as the system executes. Output may be observed in the Analysis window, or the SystemView probe can be used to view waveform development as the system executes.

Name: Statistics

Group: Numeric

Parameters: None

Description: Computes and displays the mean, variance, standard deviation, adjacent sample correlation coefficient, maximum, and minimum of the input data. Press the Ctrl key and drag the mouse to enlarge the list. The output may be observed in the Analysis window.

Name: Stop Sink

Group: Analysis

Parameters: (1) Threshold (v)

Description: This token controls the system simulation when the input is greater or equal to the specified threshold value. Select the Halt Simulation, Prompt Before Halt, Alert Only (alarm), or Go To Next Loop option. With the latter, you can have this Sink retain only the last sample of each system loop (e.g., a BER curve generation). Also refer to the Final Value Sink. The output may be observed in the Analysis window.

Name: SystemView

Group: Graphic

Parameters: None

Description: This token will display a plot of a token's output in the System window as the simulation executes. Press the Ctrl key and drag the mouse to enlarge the plot display. The output may be observed in the Analysis window.

Name: WAV 1ch

Group: Analysis

Parameters: (1) 8 or 16 bit Monaural; (2) Data Rate (Hz)

Description: SystemView monaural (1 channel) output is sent to a special Windows WAV-formatted disk file. You specify the file name and format (8 or 16 bit, and the data rate in Hz). Click the right mouse button on the Sink and select Play Audio. The output may be observed in the Analysis window.

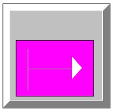
Name: WAV 2ch

Group: Analysis

Parameters: (1) 8 or 16 bit Stereo; (2) Data Rate (Hz)

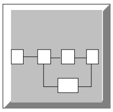
Description: SystemView stereo (2 channel) output is sent to a special Windows WAV-formatted disk file. You specify the file name and format (8 or 16 bit, and the data rate in Hz). Click the right mouse button on the Sink and select Play Audio. The output may be observed in the Analysis window, that displays the sum of the two input channels.

14.6 SystemView Token Reservoir

Token Symbol: 

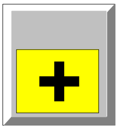
Name: Source

Description: Generic (undefined) Source token

Token Symbol: 

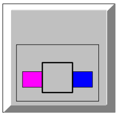
Name: MetaSystem

Description: Generic (undefined) MetaSystem token

Token Symbol: 

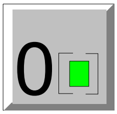
Name: Adder

Description: The Adder forms the sum of up to 20 inputs. Multiple inputs are allowed, if only one input, then the output equals the input.

Token Symbol: 

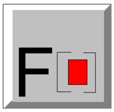
Name: MetaSystem I/O

Description: Generic (undefined) MetaSystem I/O token

Token Symbol: 

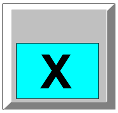
Name: Operator

Description: Generic (undefined) Operator token

Token Symbol: 

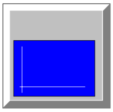
Name: Function

Description: Generic (undefined) Function token

Token Symbol: 

Name: Multiplier

Description: The Multiplier forms the product of up to 20 inputs. Multiple inputs are allowed, if there is only one input, the output is zero.

Token Symbol: 

Name: Sink

Description: Generic (undefined) Sink token

14.7 Optional Libraries

In addition to the standard token libraries, Eagleware-Elanix offers optional libraries for use in specialized applications. The libraries include:

Communications Library: This library includes error correcting encoders and decoders (BCH, RS, Golay, Viterbi, Gray), channel models (Jakes, Rayleigh, Rician, Rummler, Multipath), modulators, demodulators, bit-to-symbol, symbol-to-bit, BER counters, frequency dividers, Phase Lock and Costas loops.

DSP Library: This library includes C4x standard and extended math, conventional or IEEE standard and extended math, mixed radix FFTs, Cosine, Sine, and Hadamard transforms, mean and trend removal, NxN multiply-accumulate, FIR and IIR filter design, fixed bit adders and multipliers, variable-rate data buffers, and more.

Logic Library: This library includes Analog-to-Digital Converters ADCs, Digital-to-Analog Converters DACs, flip-flops (D, JK), counters, dividers, encoders and decoders, and more.

RF/Analog Library: This library has variable log-gain-control and RF amplifiers, op-amp circuits, passive and active mixers, power splitters and combiners, diodes (including Zener), resistor-capacitor-inductor circuits (filters), and more.

Control Logic Scheduler: This software allows the designer to control when different portions of the system process data, as a function of an algebraic/logical expression. A single scheduler may control any number of tokens, including MetaSystems. Each scheduler token has an algebraic/logical expression that is evaluated at each system sample. If the expression is true (=1), then all tokens and MetaSystems under control will be called. If the expression is false (=0), then they are not called.

M-Link Option — This software links SystemView to MATLAB and

Simulink. This library allows the incorporation of existing MATLAB functions into SystemView designs, to include blocks from MATLAB third party libraries, or write custom blocks in MATLAB in addition, you may use both SystemView and MATLAB analysis tools to examine simulation results.

CDMA/PCS Library - This contains a complete set of tools to develop systems based on the IS-95/97-A and J-STD-008 specification. It has models required to implement the forward and reverse link receiver, the transmitter, and the traffic, pilot, sync, access, and paging channels. Individual models are provided for elements making up the channel. Other models represent the full channel in a functional block, and both rate sets are supported.

DVB Library - The DVB Library has a comprehensive set of tools for rapid development of systems based on the ETS 300 744 specification. Complete sets of models representing the various blocks used for signal generation and demodulation have been provided. The key elements of the DVB system include the [204, 188, 8] shortened Reed Solomon coder, the [17, 12] convolutional interleaver, the rate ½ k = 7 punctured convolutional encoder, code polynomial (133, 171) octal, and the DVB modulator consisting of Demux, Bit and Symbol Interleaver, Symbol Mapper, and OFDM Modulator.

Real Time DSP Architect -(RTDA) TI TMS320C6000 / 5000- This tool provides an interface between SystemView simulations and DSP software designs, by providing a link to Texas Instrument Code Composer Studio™. The RDTA option allows for real time signal generation, data acquisition, analysis and hardware-in-the-loop simulation using TI hardware.

FPGA Architect - Xilinx XC400/Spartan - This option enables a flow from SystemView simulations to FPGA implementation by providing a seamless link to Xilinx LogiCORES, CoreGEN and the Alliance tool suite from Xilinx. This allows the implementation of bit-true DSP designs into Xilinx FPGAs.

14.8 Alphabetical Listing of Tokens

Name	Library	Group
a^X	Function	Algebraic
Adder	Token Reservoir	
Analysis	Sink	Analysis
ArcTan	Function	Functions
ArcTan 4	Function	Functions
Average	Operator	Filters / Systems
Averaging	Sink	Analysis
Block	Function	Non Linear
Cmltv Avg	Function	Functions
Compare	Operator	Logic
Coulomb	Function	Non Linear
Crt-Plr	Function	Complex
Current Value	Sink	Numeric
Custom	Function	Functions
Custom	Source	Aperiodic
Cx Add	Function	Complex
Cx Multiply	Function	Complex
Cx Rotate	Function	Complex
Data List	Sink	Numeric
Switch	Operator	Logic
Dead Band	Function	Non Linear
Decimator	Operator	Sample / Hold
Delay	Operator	Delay
Derivative	Operator	Integral / Differential
Digital Scalar	Operator	Gain / Scale
Divide	Function	Algebraic
Ext 1ch	Sink	Export
Ext 2ch	Sink	Export
Ext 1ch	Source	Import
Ext 2ch	Source	Import
Extract	Function	Multiplex
FFT	Operator	Filters / Systems
Final Value	Sink	Numeric
Fractional Part	Operator	Gain / Scale
Freq. Mod	Function	Phase / Frequency

Alphabetical Listing of Tokens (Continued)

Name	Library	Group
Gain	Operator	Gain / Scale
Gauss Noise	Source	Noise / PN
Half Rectify	Function	Non Linear
Hold	Operator	Sample / Hold
Hysteresis	Function	Non Linear
Impulse	Source	Aperiodic
Input	MetaSystem	MetaSystem I/O
Integrator	Operator	Integral / Differential
Limit	Function	Non Linear
Linear Sys	Operator	Filers / Systems
Log	Function	Functions
Logical AND	Operator	Logic
Logical NAND	Operator	Logic
Logical NOT	Operator	Logic
Logical OR	Operator	Logic
Logical XOR	Operator	Logic
Max-Min	Operator	Logic
Meta System	MetaSystem	MetaSystem
Modulo	Operator	Gain / Scale
Multiplex	Function	Multiplex
Multiplier	Token Reservoir	
Negate	Operator	Gain / Scale
OSF	Operator	Filters / Systems
Output	MetaSystem	MetaSystem I/O
Phase Mod	Function	Phase / Frequency
Peak-Hold	Operator	Sample / Hold
PID	Operator	Integral / Differential
Plr-Crt	Function	Complex
PN Sequence	Source	Noise PN
Polynomial	Function	Algebraic
PSK Carrier	Source	Periodic
Pulse	Operator	Logic
Pulse Train	Source	Periodic
Quantize	Function	Non Linear

Name	**Library**	**Group**
Real-Time	Sink	Graphic
ReSample	Operator	Sample / Hold
Sample Delay	Operator	Delays
Sample Hold	Operator	Sample / Hold
Sampler	Operator	Sample / Hold
Sawtooth	Source	Periodic
Select	Operator	Logic
Sigmoid	Function	Functions
Sine	Function	Functions
Sinusoid	Source	Periodic
Statistics	Sink	Numeric
Step Function	Source	Aperiodic
Stop Sink	Sink	Analysis
SystemView	Sink	Graphic
Tangent	Function	Functions
Tanh	Function	Functions
Thermal Noise	Source	Noise / PN
Time	Source	Aperiodic
Uniform Noise	Source	Noise / PN
Variable Delay	Operator	Delays
Vector Fct.	Function	Algebraic
Wav 1ch	Source	Import
Wav 2ch	Source	Import
Wav 1ch	Sink	Export
Wav 2ch	Sink	Export
X^a	Function	Algebraic
Xtrnl Fct	Function	Non Linear

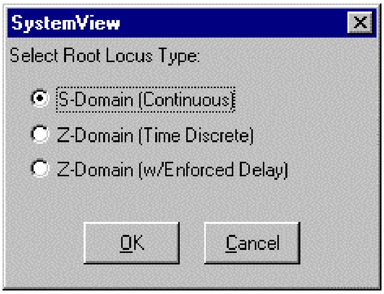
Chapter 15. Root Locus and Bode Plots

SystemView will *algebraically* (i.e., not by numerical simulation), compute Root Locus and Bode plots for a linear system, as it appears in the System window, when the Root Locus or Bode Plot button is clicked on the toolbar. This feature is available in the Linear System window and in the Laplace window, as outlined in Chapter 6.

15.1 Root Locus

To compute the root locus, SystemView must first determine the open-loop transfer function of the system.

1. Disconnect the output of the last token in the feedback path of the system (e.g., disconnect token 9 from token 1 as shown in Figure 15.2).
2. Click the Root Locus button in the System window.
3. In the dialog box, shown in Figure 15.1, select S-Domain, Z-Domain, or Z-Domain w/enforced delay, (has added poles at z=0 resulting from the implicit sample delays).
4. Click OK.



SystemView

Select Root Locus Type:

● S-Domain (Continuous)

○ Z-Domain (Time Discrete)

○ Z-Domain (w/Enforced Delay)

Figure 15.1. Domain selection menu for the Root Locus computations

SystemView will perform the conversions necessary to compute the specified root locus, Regardless of the type of system. For example, the bilinear transform is applied to all continuous Linear System tokens if either Z-Domain is selected.

The Root Locus is the locus of the poles (as a function of the loop gain *k*) in a closed-loop feedback system, whose open-loop transfer function is defined in the System window. Thus, it is a plot of the poles of, $\dfrac{H(s)}{1+kH(s)}$ or equivalently, the roots of $D(s)+kN(s)=0$, where $H(s)=\dfrac{N(s)}{D(s)}$ is the transfer function, that SystemView computes from the linear system block diagram. If the system is defined entirely in the z-domain then the above definitions would be in terms of *H*(*z*)

To compute the open-loop transfer function, SystemView assumes the loop at the feedback point, token 9 and 1, was opened prior to clicking the Root Locus button. For an ambiguous situation, SystemView may ask which token in the system is the last token in the open loop.

An example is shown in Figure 15.2, for the third-order system. The open-loop transfer function for this system is, $H(s)=\dfrac{-5}{s(s+1)(s+5)}$

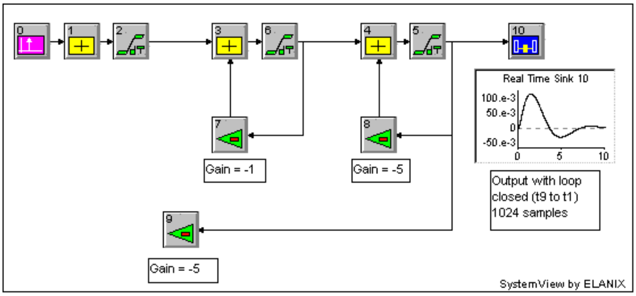
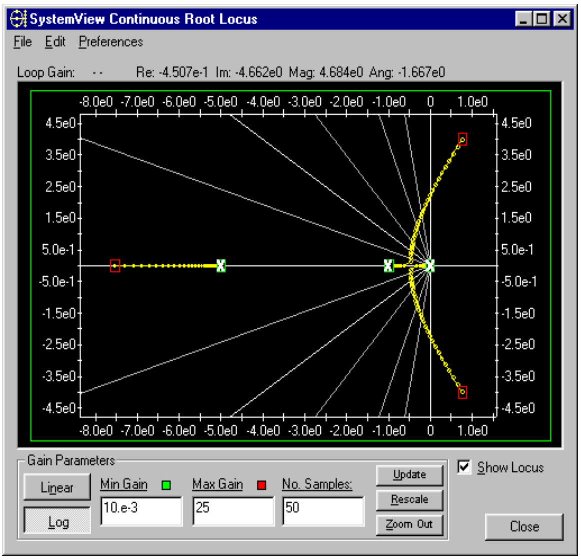


Figure 15.2. A third-order system for Root Locus and Bode computation

Figure 15.3 shows the corresponding Root Locus window. Move the mouse along the root locus and read the loop gain value displayed in the window. This is the *additional* gain factor for the system. If the system already has feedback gain, (e.g., token 9 in Figure 15.2), then the total feedback gain corresponding to a root location is the displayed gain value *times* the existing gain.

The system in Figure 15.2 has poles in the right half plane for an additional gain of six or greater and will be unstable for a total loop gain exceeding approximately 30 (6.027 x 5, token 9 has a gain of 5). Note the root locus in the Z-Domain with implicit delays, shows instability for an additional gain of 6.3, or total of 31.5. The radial lines show critical damping ratios, 0.9, 0.707, 0.5, 0.25, and 0.1



Loop Gain: - -   Re: -4.507e-1 Im: -4.662e0 Mag: 4.684e0 Ang: -1.667e0

Min Gain 🟩 10.e-3

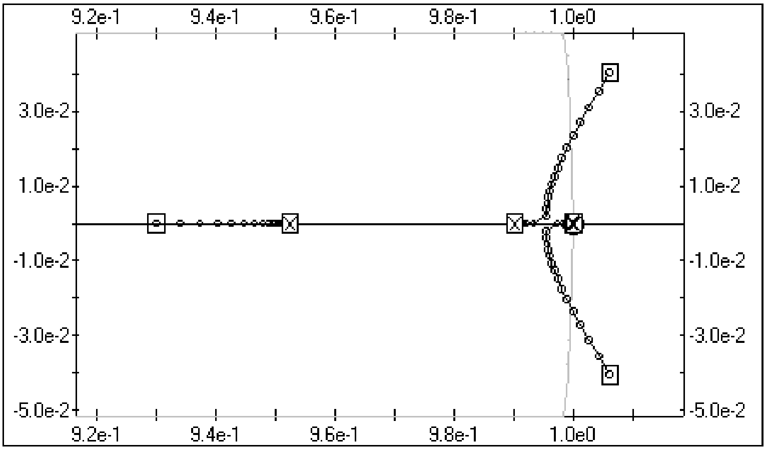
Max Gain 🟥 25

No. Samples: 50

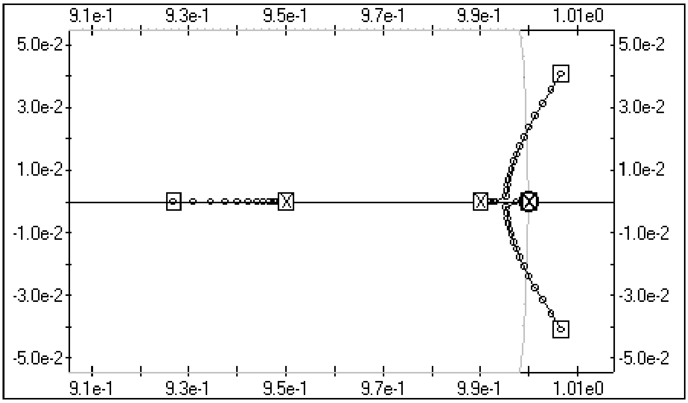
☑ Show Locus

Figure 15.3.The Root Locus window.

For viewing system Poles and Zeros, system Poles are indicated in the display with the symbol “X”, and Zeros are displayed using the symbol “0”. The roots associated with the minimum gain sample (i.e., the start of the locus) are indicated with a green box, and the roots associated with the maximum gain sample are indicated with a red box.



Z-Domain Root Locus for the system in Figure 15.2 (zoomed)



- Z-Domain Root Locus (with implicit delays) for system in Figure 15.2 (zoomed)

The Root Locus window is interactive; below are descriptions of analysis tools.

Moving Poles and Zeros — Movement is only enabled when Root Locus is launched, while in the Linear System design window. When the mouse is over any pole or zero the pointer will change to a multi-pointed arrow, allowing it to be moved to a new location in the s- or z-domain.

Real Time Bode Plot — When movement of poles and zeros is enabled, the results are seen in the Bode plot window within the root locus plot (if not visible, click the right mouse button on the root locus plot and select View Bode Plot). Click the right mouse button within the Bode plot to select linear or log frequency scale. Click the left mouse button to show the size blocks and change the Bode plot dimensions.

Zoom Out Button — Click to zoom out by factors of 2x.

Zoom — Press the Ctrl key and click and drag the mouse to zoom a selected area.

Gain, Real and Imaginary Values — Place the mouse anywhere on the root locus curve and the corresponding loop gain is displayed, along with the real and imaginary coordinates of the mouse.

Linear/Log — Allows the selection of either linear distribution of gain samples (from Min Gain to Max Gain), or a logarithmic distribution.

Min Gain, Max Gain — Allows the entry of minimum and or maximum gain used in the root locus computation.

No. Samples — Allows the entry of the desired number of gain samples, to be used in the computations.

Update — Forces re-computation of the root locus, after changes are made in the parameter-entry boxes.

Reset — This option allows the return to the default settings.

Close — Allows closure of the root locus window and return to the system.

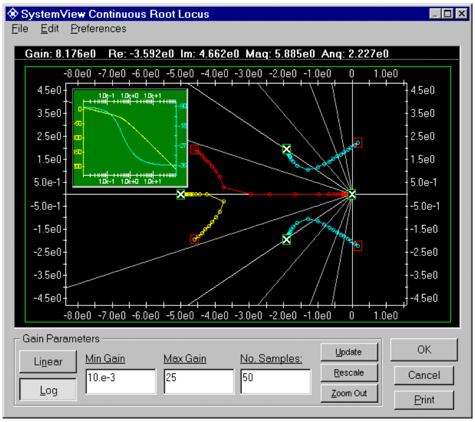
Copy — Allows the copy of the root locus plot to the Windows clipboard. If Use Color Printer has been selected, the copy will be in color.

Print — Allows the printing of the root locus plot. If Use Color Printer has been selected in the Preferences menu, the print will be in color.

15.2 Moving Poles and Zeros

There are two ways to directly edit poles and zeros: 1) using the mouse in the Root Locus window, and 2) manual editing in the Laplace window

Using the Mouse: Position the mouse over a pole (X) or zero (O). When the mouse pointer changes to a multi-pointed arrow, click and drag it to the desired location in the s- or z-domain (Figure 15.4). SystemView will automatically update H(s) or H(z) to reflect the change. Note that a strictly real pole or zero can be moved off the real axis. SystemView will automatically create a complex conjugate pair.

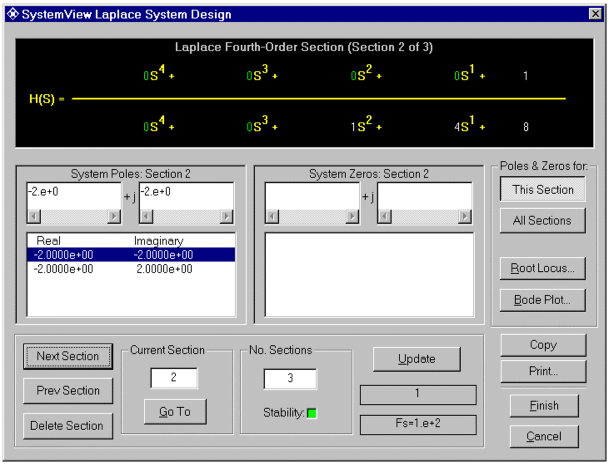


Gain: 8.176e0 Re: -3.592e0 Im: 4.662e0 Mag: 5.885e0 Ang: 2.227e0

Min Gain: 10.e-3
Max Gain: 25
No. Samples: 50

Figure 15.4. Using the mouse to move the Pole at (-1+j0) in Figure 15.3 to (-2 ±j2).

Manual Edit: In the Laplace window, select the option titled "This Section" located above the Root Locus button. (For editing filters from the Analog Filter design window, click "Convert to Laplace" to enter the Laplace window). Next move to the Laplace fourth-order section to be edited. (Use the Next Section button to move between sections) Now use the mouse to select the pole or zero, in the list to be changed. Edit the pole or zero, by typing the new value for the real and imaginary parts and clicking the Update button (Figure 15.5). SystemView will automatically update H(s) or H(z) to reflect the change. Note that a strictly real pole or zero can be moved off the real axis. SystemView will automatically create a complex conjugate pair.



Laplace Fourth-Order Section (Section 2 of 3)

$$H(S) = \frac{0s^4 + 0s^3 + 0s^2 + 0s^1 + 1}{0s^4 + 0s^3 + 1s^2 + 4s^1 + 8}$$

System Poles: Section 2

-2.e+0 + j -2.e+0

Real	Imaginary
-2.0000e+00	-2.0000e+00
-2.0000e+00	2.0000e+00

System Zeros: Section 2

+ j

Current Section: 2

No. Sections: 3

Stability: 🟩

1

Fs=1.e+2

Figure 15.5. Manually editing the Pole at (-1+j0) in Figure 15.3 to (-2 ±j2).

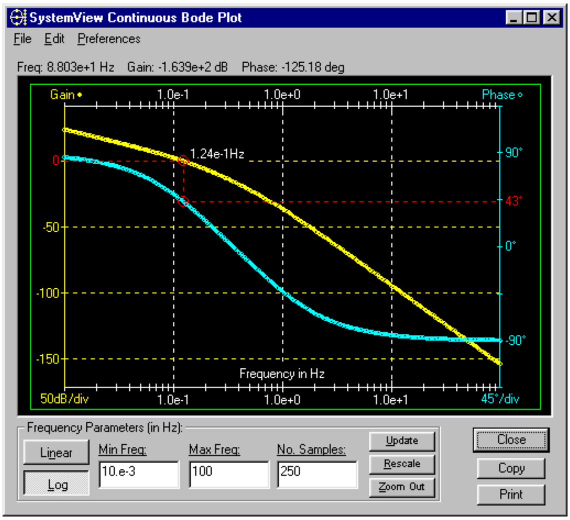
15.3 Bode Plots

The Bode plot shows the magnitude and phase of $H(s)$ as a function of the frequency f in Hz (with $s = j2\pi f$). An example is shown in Figure 15.6 for the *open-loop* system discussed in Section 15.1, and shown in Figure 15.2. The *closed-loop* (i.e., connecting Gain token 9 to Adder token 1) response for the same system is shown in Figure 15.7.

- For the open-loop Bode plot, disconnect the output of last token in the feedback path of the system (e.g., disconnect token 9 from token 1 as in Figure 15.2).
- Click the Bode Plot button in the System window,
- Select S-Domain, Z-Domain, or Z-Domain with added poles at z=0 (resulting from the implicit sample delays as described in Chapter 13).
- Click OK.

While in either the Root Locus or Bode Plot window, you can zoom the display. Press the Ctrl key and click/drag the mouse to outline the area to be zoomed. The number of samples, the gain or frequency range, and linear or log (gain or frequency) spacing can also be specified.

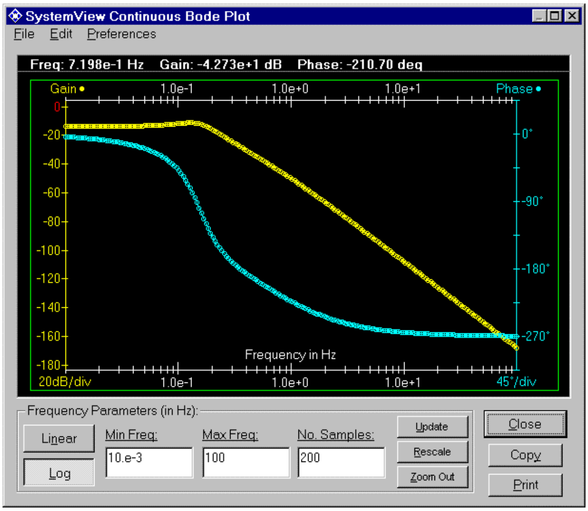
The system phase margin is computed from the system phase at the frequency where the system gain is 0dB. This point is automatically indicated in the display as shown in Figure 15.6.



Freq: 8.803e+1 Hz Gain: -1.639e+2 dB Phase: -125.18 deg

Min Freq: 10.e-3 Max Freq: 100 No. Samples: 250

Figure 15.6 The SystemView Bode plot window for the open-loop system



Freq: 7.198e-1 Hz Gain: -4.273e+1 dB Phase: -210.70 deg

Min Freq: 10.e-3 Max Freq: 100 No. Samples: 200

Figure 15.7. The SystemView Bode plot window for the closed-loop system

Chapter 16. Editing the Execution Sequence

16.1 Introduction

SystemView provides the tools required to customize the order in which the tokens are executed during the simulation. This is useful in systems that contain multiple feedback connections, or when the feedback structure is inherently ambiguous due to the location of implicit computational delays. (See Chapter 13 for information on implicit delays.) To edit an execution sequence, use any of the following techniques.

16.2 Using the Mouse

The easiest method for editing the execution sequence is to simply click and drag an implicit delay to another location. As an example, the implicit delay [z-1] is shown at the input to token 1 in Figure 16.1. Remember that implicit delays can only exist at the input to a multiple-input token, and SystemView will notify the user by indicating the selection of an “illegal” location.

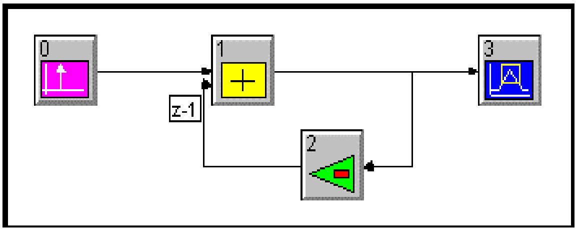
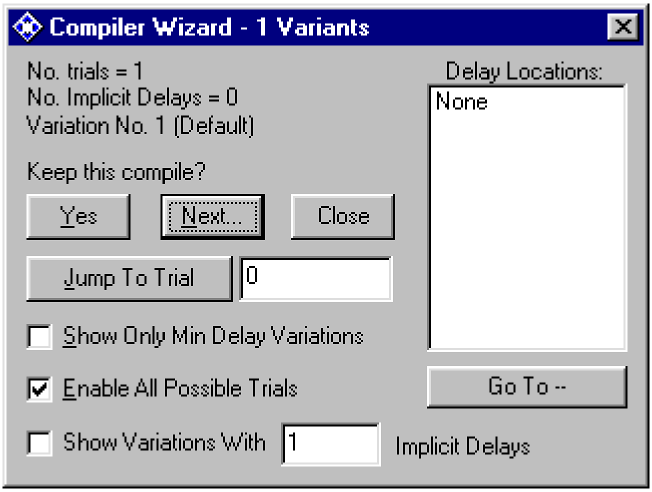


Figure 16.1. The SystemView implicit delay

16.3 Using the Compiler Wizard (Auto Sequence Editor)

The Compiler Wizard is an automated method for editing the execution sequence. Select Compiler Wizard in the Compiler menu to open the window shown in Figure 16.2.



Compiler Wizard - 1 Variants

No. trials = 1
No. Implicit Delays = 0
Variation No. 1 (Default)

Keep this compile?

Delay Locations: None

Jump To Trial: 0

□ Show Only Min Delay Variations

☑ Enable All Possible Trials

□ Show Variations With 1 Implicit Delays

Figure 16.2.The SystemView Compiler Wizard window.

The Compiler Wizard will attempt to find all valid variations of the token execution sequence for the system. For each sequence, the Compiler Wizard displays the location of all implicit delays and lists every token that has an implicit delay at its input. Double click any listed token, and SystemView will display that token.

Click the Next button to have the Compiler Wizard find the next valid execution sequence. Click the Yes button to keep the compiler variation that is acceptable. The option titled Enable Finer Compiler Variations, forces the Compiler Wizard to perform a deeper search for possible sequence variations and may take longer to execute.

16.4 Using the System Tokens

Selecting “Use Sys Tokens” provides a simple way to edit the system execution sequence graphically. Click the system or MetaSystem tokens in the order they are to be executed. Begin with Source tokens, or choose a location anywhere in the system to edit only a portion of the sequence. Click Cancel Last Edit in the Compiler menu to cancel the last token selected. Click Cancel Edit Operation to cancel the entire operation.

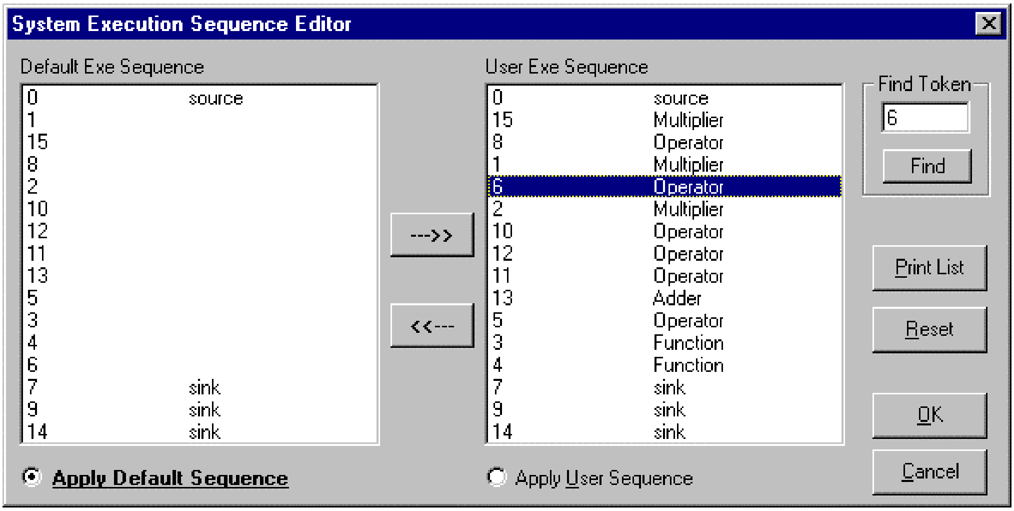
When finished with the sequence edit, click End Edit in the Compiler menu. To view the results, click Use the Sequence Editor or click Animate Exe Sequence in the Compiler menu.

16.5 Using the Exe List

When the Use Exe List is selected, the Sequence Editor shown in Figure 16.3 appears. (The entries shown in the example are from the example file Costas.Svu located in the \sysvu_32\examples directory. This system has a user-defined custom execution sequence.)

The default execute sequence is listed on the left side of the window. This is the ordered execution list for each token, as compiled. An empty list (except for Sources and sinks, which may not be changed), or the previously defined custom sequence that was created, will be displayed on the right side of the window.

When starting with an empty list, edit the sequence by selecting the first token or group of tokens from the default sequence. Select the token from the user list to be inserted, after the selection from the default sequence. Click the transfer button (--->>) to move the selection to the specified insertion point in the user list. Continue this process until the new sequence is complete. Use the Find Token button to locate any specific token in long lists. A token or a group can be removed from the user list by first selecting and then clicking the reverse transfer button (<<---). When editing an existing User Sequence list, remove the token from the Use Exe Sequence before inserting it into its new position.



Default Exe Sequence

0	source
1	
15	
8	
2	
10	
12	
11	
13	
5	
3	
4	
6	
7	sink
9	sink
14	sink

User Exe Sequence

0	source
15	Multiplier
8	Operator
1	Multiplier
6	Operator
2	Multiplier
10	Operator
12	Operator
11	Operator
13	Adder
5	Operator
3	Function
4	Function
7	sink
9	sink
14	sink

Find Token: 6

● Apply Default Sequence

○ Apply User Sequence

Figure 16.3.The SystemView Execution Sequence Editor.

After an execution sequence has been created, select the Apply User Sequence option located under the user sequence list, and then click OK to return to the system.

Verification of the new sequence may be done visually, by selecting the Animate Exe Sequence located in the Compiler menu.

The default sequence can be selected, or the custom sequence, by selecting either Use Default Exe Sequence or Use Custom Exe Sequence, located in the Compiler menu.

Chapter 17. Global Parameter Links

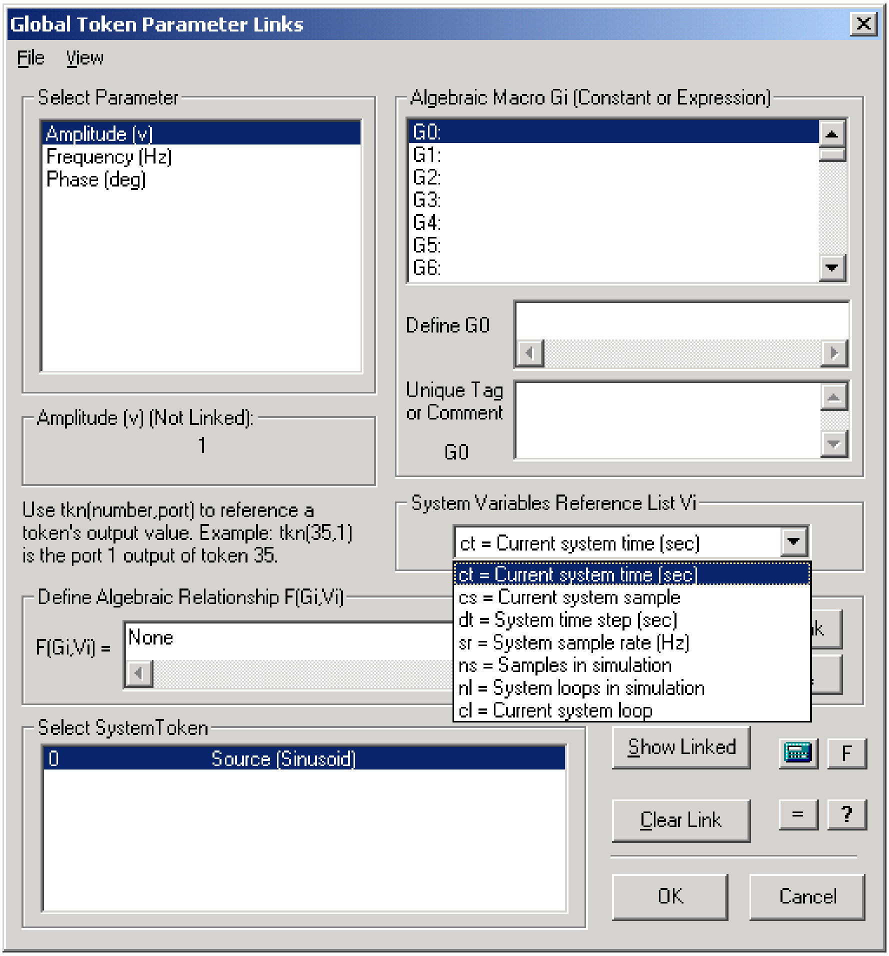
17.1 Introduction

The Global Parameter Links tool is designed to simplify the task of creating and modifying related parameters within a system model. This tool provides the ability to algebraically link token parameters to selected system-level parameters (e.g., the system sample rate) and/or global constants that are defined. Global links are applied only at the beginning of each system loop, as they are defined in the System Time window. A feature of SystemView allows the designer to determine the parameter for a given token in terms on a runtime value.

Users are able to define token parameters in terms of system variables through the Global Parameter Links window. Users can define token parameters (such as Amplitude, Frequency, and Phase) in terms of such system variables including system time step (dt), system sample rate (sr), samples in simulation, (ns), system loops in simulation, (nl), and current system loop (cl).

Users can also allow token parameters to depend on runtime system variables, such as current system time (ct) and current system sample (cs); these variables can be found in the System Variables Reference List Vi in the Global Parameter Link Window, as shown in figure 17.1.

In addition, the user is able to define token parameters in terms of a specific token's output, giving the designer unlimited control of the runtime definition of token parameters. This dynamic control of token parameters comes in the form of the tkn (*token ID, output ID*) function, where tkn (1,0) represents the output from the 0th port of Token 1.



Global Token Parameter Links

Select Parameter

- Amplitude (v)
- Frequency (Hz)
- Phase (deg)

Algebraic Macro Gi (Constant or Expression)

- G0:
- G1:
- G2:
- G3:
- G4:
- G5:
- G6:

Define G0

Unique Tag or Comment G0

Amplitude (v) (Not Linked): 1

Use tkn(number,port) to reference a token's output value. Example: tkn(35,1) is the port 1 output of token 35.

System Variables Reference List Vi

ct = Current system time (sec)

- ct = Current system time (sec)
- cs = Current system sample
- dt = System time step (sec)
- sr = System sample rate (Hz)
- ns = Samples in simulation
- nl = System loops in simulation
- cl = Current system loop

Define Algebraic Relationship F(Gi,Vi)

F(Gi,Vi) = None

Select SystemToken

0 Source (Sinusoid)

Figure 17.1. The Global Parameter Links window

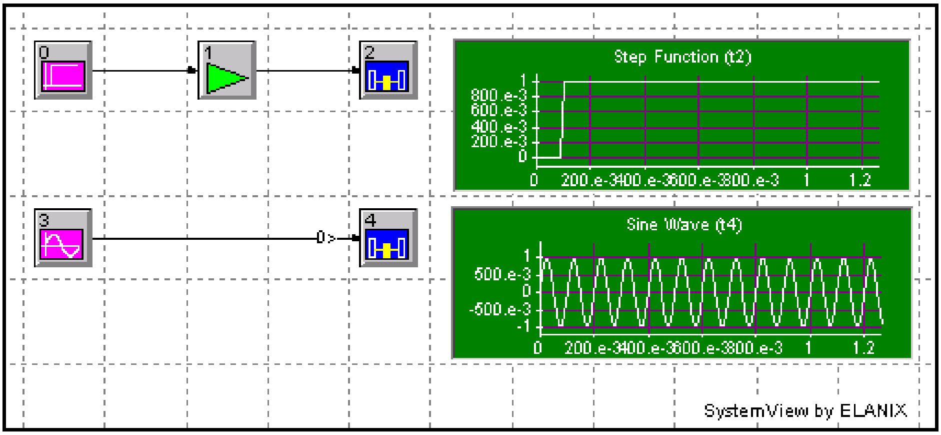
17.2 Operation

The procedure for specifying links follows.

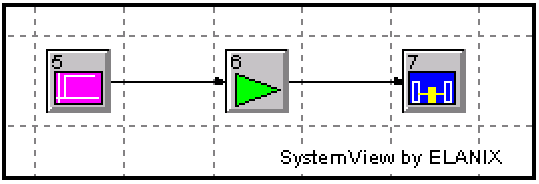
1. Click the Tools menu and select Global Parameter Links.
2. Select the token whose parameters are to be linked, from the list of system tokens.
3. Select the parameter for the token to be linked, from the Select Token Parameters.
4. To define a Global Algebraic Macro, select one from the Global Constants list and enter the desired expression or value in the Set Gi text box.
5. Enter any reminder or definition notes for the global constant in the Notes box. *The first ten characters of the note will appear on the same line as the global constant for reference.* (Click the Global Constant or Update button to view them.)
6. Specify the algebraic link by entering the algebraic relationship in the Define Algebraic Relationship box. The expression can contain global macros (e.g., *G0*, *G1*, etc.) that have defined, and system variables from the System Variable list (e.g., dt, the system time increment). For example *G0*dt + 2*G1*.
7. To use system-level variables in the link, click the down arrow in the System Variables Reference List and click on the desired system variable. The variable can also be typed directly in the algebraic expression.
8. Click the Update Link button to activate the link.
9. Click the Show Linked button for a list of system tokens defined links.
10. Click OK to set the links and return to the system.

17.3 Example Application

Let us develop a simple example to display the use of the tkn () function. Token 0 is a step function source, starting at sample time 0.1 seconds passing into a gain token (Token 1) with a gain of 1.0, and ending in a sink (Token 2). Token 3 is a sine wave source with amplitude of 1volt and frequency of 10 Hz being sent to another sink (Token 4). With standard use of SystemView, the data is generated as seen below:

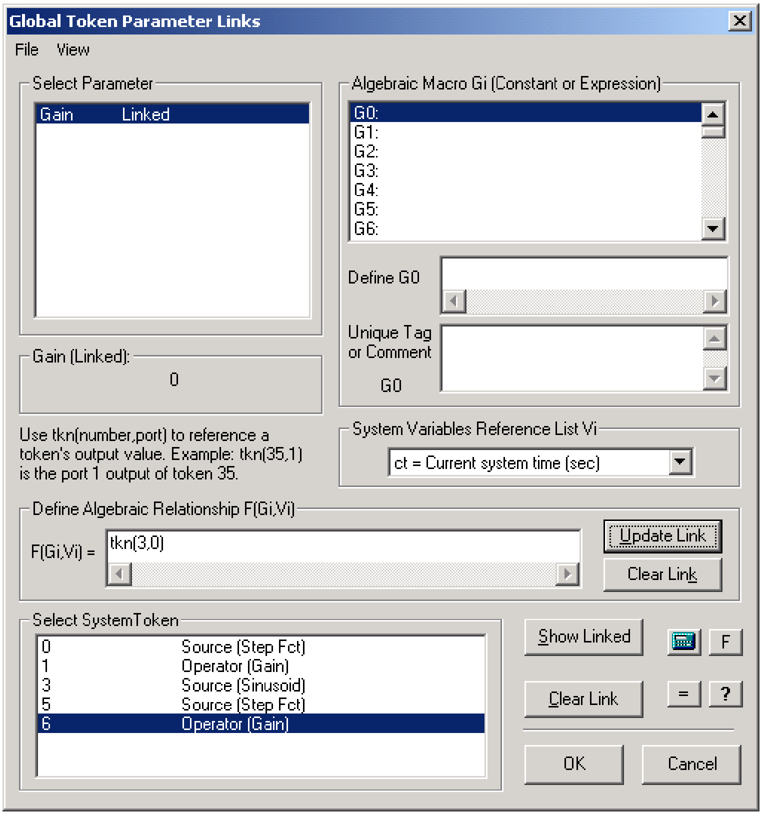


In order to illustrate dynamic parameter control, let us duplicate Tokens 0, 1, and 2, creating Tokens 5, 6, and 7 as shown below:



Now globally link the gain value of Token 6 in terms of the output of our sine wave source Token 3. Open the Global Parameter Links dialog window shown in figure 17.2, from the Tools menu and select the Gain token in the "Select System Token" window. In SystemView, we can define the Algebraic

Expression F (Gi, Vi) to be "tkn (3,0)" that is equal to the data from the 0th port of Token 3 (our sine wave source token):



Select Parameter

Gain Linked

Gain (Linked):

0

Use tkn(number,port) to reference a token's output value. Example: tkn(35,1) is the port 1 output of token 35.

Algebraic Macro Gi (Constant or Expression)

G0:
G1:
G2:
G3:
G4:
G5:
G6:

Define G0

Unique Tag or Comment

G0

System Variables Reference List Vi

ct = Current system time (sec)

Define Algebraic Relationship F(Gi,Vi)

F(Gi,Vi) = tkn(3,0)

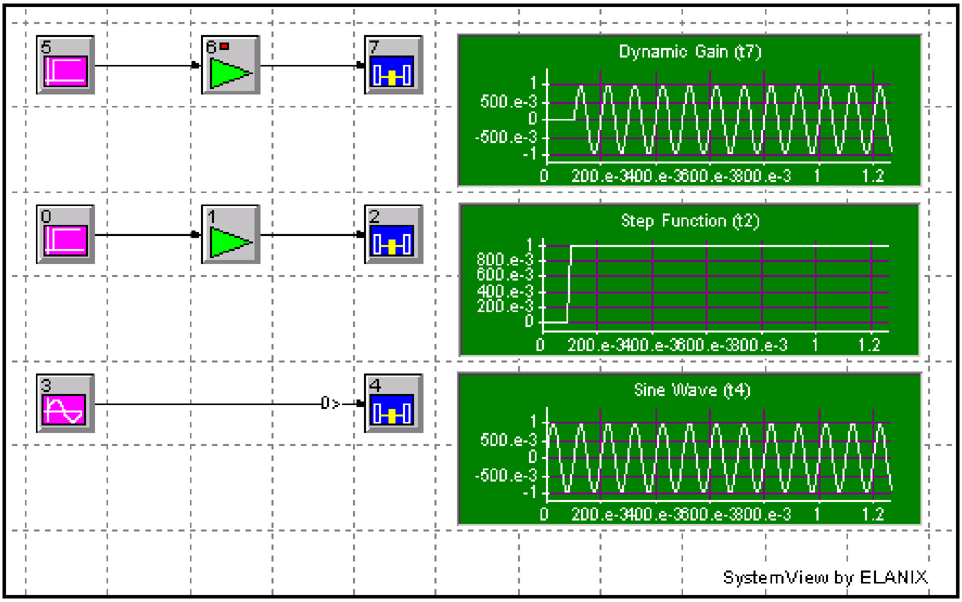
Select SystemToken

0	Source (Step Fct)
1	Operator (Gain)
3	Source (Sinusoid)
5	Source (Step Fct)
6	Operator (Gain)

Figure 17.2

If the system is re-run, we obtain the desired behavior from the globally linked token, giving us a step function source multiplied by a time-varying gain, as seen below – note the desired output of Token 7:

In addition to the Global Parameter Links window, the tkn () function is available



17.4 Additional Information

The **Vi: System Variables Reference List** includes the following variables.

Name	Description
sr	System sample rate (Hz)
dt	System time step (sec)
ns	Number of samples in simulation
nl	Number of loops in simulation
cl	Current system loop index
ct	Current system time (sec)
cs	Current system sample

The **Define Algebraic Relationship F (Gi, Vi)** Update Link button, updates all defined parameter links. The Clear Link button clears the link only for the selected parameter.

The **Select System Token** window, lists all tokens in the system that have linkable parameters. If the Show Linked button is clicked, only linked tokens are displayed. The Clear Link button clears all the parameter links for the selected token.

Chapter 18. Variable Parameter Editing

18.1 Introduction

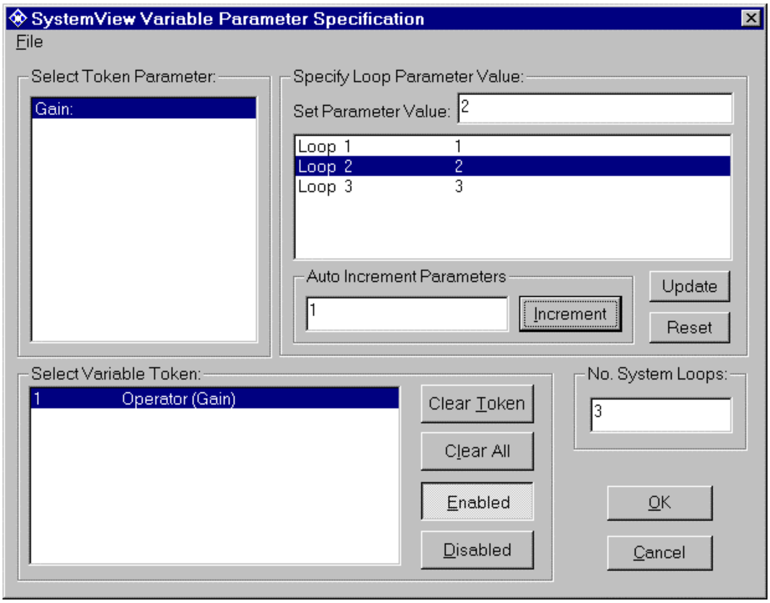
The variable Token and Dynamic Parameter Editing function is used in conjunction with the system loop feature (see Chapter 4, *System Time*), the Variable Token Parameters window provides the ability to set up token parameters that vary over system loops.

18.2 Setting Variable Token Parameters

Operation
To set variable token parameters; first specify the number of system loops in the System Time window. Then follow the procedure outlined below and refer to Figure 18.1.

1. Click the Tokens menu and select New Variable Token.
2. Click the system token whose parameters are to be varied..
3. Select the parameter in the Token Parameter list that will be made a variable.
4. Use the mouse to select a loop or group of loops. Enter the desired value for that loop or loops, in the Set Parameter Value box. Press Enter or click the Update button. Continue this process until the token's parameter value is defined for each of the system Loops.
5. Use the Auto Increment Parameters box if the parameter value is to increase by a fixed amount on each Loop. Enter the increment and press Enter or click the Update Increment button.
6. Click the OK button to activate the entry, and repeat this process for each token that requires variable parameters.

To edit or change the parameter variations, click the Tokens menu and select Edit Parameter Variations.



Select Token Parameter: Gain:

Specify Loop Parameter Value:

Set Parameter Value: 2

```
Loop 1        1
Loop 2        2
Loop 3        3
```

Auto Increment Parameters: 1

Select Variable Token: 1 Operator (Gain)

No. System Loops: 3

Figure 18.1. The Variable Parameter window

Examples

Set the system time to the default values (128 samples, 100 Hz sample rate) and specify three loops. Launch a Sinusoidal token from the Sinusoid/Periodic Source Group (default parameters of 10 Hz), a Gain token from the Gain/Scale Operators Group (Click OK to set the parameters to default values), and a Real Time Sink token from the Graphical Sinks Group.

Connect the Sinusoid Source (sine output) to the Gain Operator, and then connect the Gain Operator to the Real Time Sink. Now click the Tokens menu and choose Select New Variable Token, then click the Gain token.

Enter “1” in the Auto Increment box and click the Increment button. Click OK. Run the simulation and observe the results in the Analysis window. The waveform in Figure 18.2 should be seen.

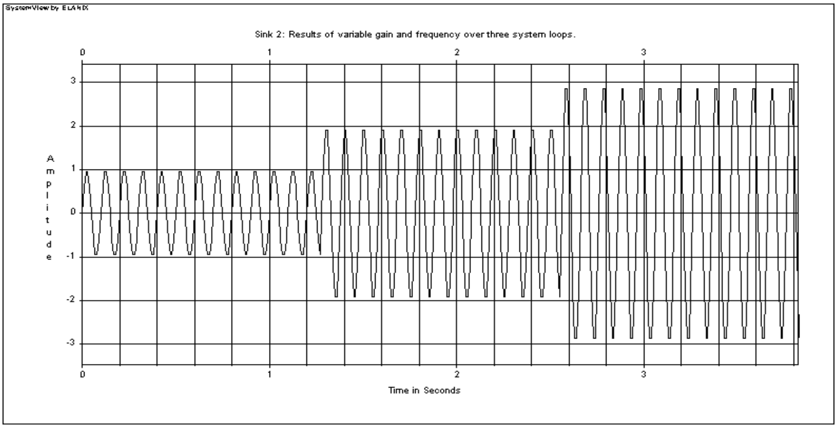
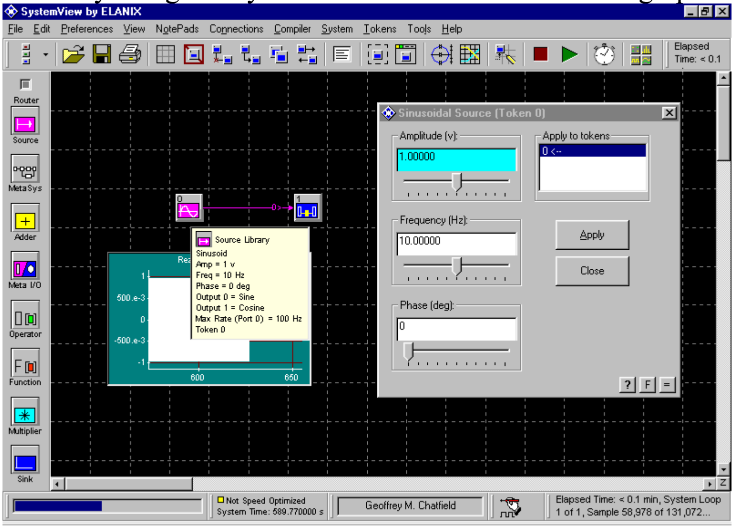


Figure 18.2. Results of the variable parameter example

18.3 Dynamic Parameter Editing

SystemView allows the modification of parameters during the time the system is executing. Use the slide bars shown in figure 18.3, to modify parameters and see the results immediately using the System Probe and/or the real-time graphic display sinks.



Source Library
Sinusoid
Amp = 1 v
Freq = 10 Hz
Phase = 0 deg
Output 0 = Sine
Output 1 = Cosine
Max Rate (Port 0) = 100 Hz
Token 0

Amplitude (v): 1.00000

Apply to tokens: 0 <--

Frequency (Hz): 10.00000

Phase (deg): 0

Not Speed Optimized
System Time: 589.770000 s

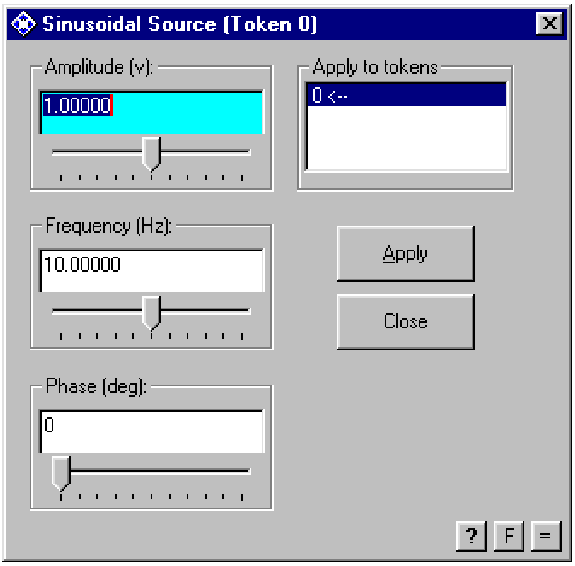
Geoffrey M. Chatfield

Elapsed Time: < 0.1 min, System Loop 1 of 1, Sample 58,978 of 131,072...

Figure 18.3. Right mouse click on a token during system execution to dynamically edit token parameters

The steps listed below, should be followed to dynamically edit the token parameters during system execution:

1. Right mouse click on the desired token while the system is running and select Edit Parameters. The following screen will be displayed:



2. Select a slide bar and move it right or left to increase or decrease the desired parameter value.

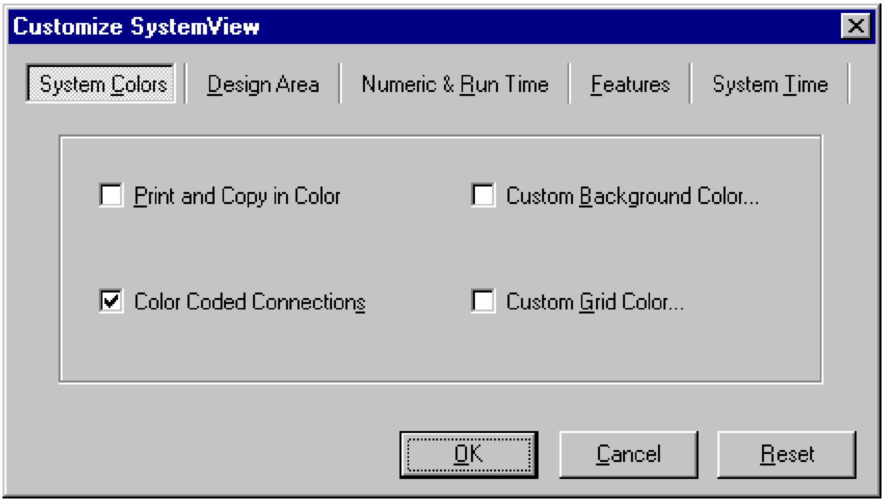
3. The results of the change will be immediately reflected in any graphic display, numerical display or in the Dynamic System Probe window.

Chapter 19. Customizing SystemView

SystemView has many properties and features that can be customized for particular needs. Click the Preferences menu in either the System or Analysis window and select *Properties.* Descriptions follow.

19.1 System Window

System Colors



☐ Print and Copy in Color

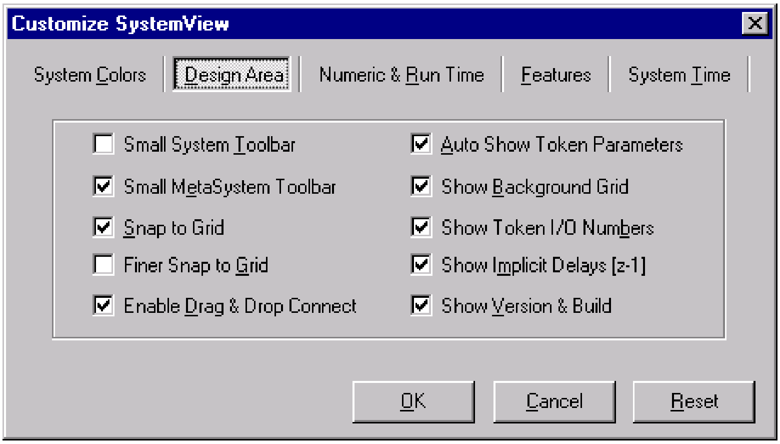
☐ Custom Background Color...

☑ Color Coded Connections

☐ Custom Grid Color...

- **Print and Copy In Color (Default Off)** — Select this option *only* if a color printer is available, and printing or copying SystemView data in color is desired. Both printing and copying to the Windows clipboard are affected by this option.
- **Color Coded Connections (Default On)** — When selected, all connections between SystemView tokens are color coded to match the library from which the originating token was drawn (e.g., Operator token outputs are green). When this option is off, all connections are either black or white, depending on the System window background color.
- **Custom Background Color (Default Off)** — Select this option to change the color of the System window background color.
- **Custom Grid Color (Default Off)** — Select this option to change the color of the grid displayed in the System window design area.

Design Area



☐ Small System Toolbar

☑ Small MetaSystem Toolbar

☑ Snap to Grid

☐ Finer Snap to Grid

☑ Enable Drag & Drop Connect

☑ Auto Show Token Parameters

☑ Show Background Grid

☑ Show Token I/O Numbers

☑ Show Implicit Delays [z-1]

☑ Show Version & Build

- **Small System Toolbar –** This option decreases the size of the System toolbar.

- **Small MetaSystem Toolbar –** This option decreases the size of the MetaSystem toolbar.

- **Snap To Grid (Default On)** — When selected, all tokens are automatically positioned relative to the System window grid. Turn this option off to position tokens anywhere on the grid.

- **Finer Snap To Grid (Default Off)** — This option doubles the grid resolution from the default value.

- **Enable Drag & Drop Connect (Default On)** — Allows quick connect and disconnect of tokens. Place mouse on token until up arrow appears, hold left mouse button down, and drag towards next token until they connect. To disconnect, place mouse on token to be disconnect, until up arrow (with a break in it) appears. Connections to that token will appear as dashed lines. Hold left mouse button down, and click on the token to be disconnected.

- **Auto Show Token Parameters (Default On)** — This option displays SystemView token parameters when the mouse cursor is pointed at a token.

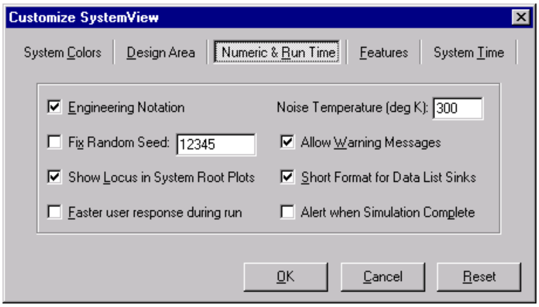
- **Show Background Grid (Default On)** — This option displays or hides the System window grid.

- **Show Token I/O Numbers (Default On) —** This option displays or hides the output to input port numbers

- **Show Implicit Delays [z-1] (Default On)** — This option displays or hides the location of implicit delays (if any) in the system. See Chapter 13 for more information on implicit delays.

- **Show Version and Build (Default On) —** This option shows the current SystemView Version and Build in the upper left hand corner of the SystemView design area.

Numeric & Run Time



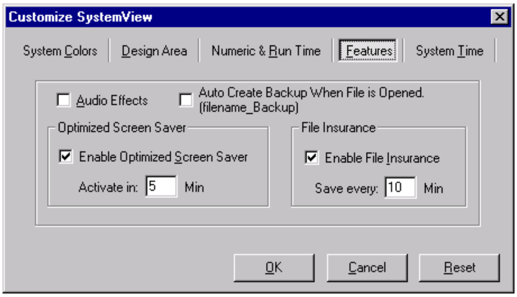
☑ Engineering Notation
Noise Temperature (deg K): 300
☐ Fix Random Seed: 12345
☑ Allow Warning Messages
☑ Show Locus in System Root Plots
☑ Short Format for Data List Sinks
☐ Faster user response during run
☐ Alert when Simulation Complete

- **Engineering Notation (Default On)** — When this is on, SystemView will display all numeric information in engineering notation using multiples of 3 for powers of 10, such that 12345 Hz is displayed as 12.345e+3 Hz. When Engineering Notation is off, 12345 Hz is displayed as 1.2345e+4 Hz.

- **Fix Random Seed (Default Off)** — When selected, SystemView will use the random seed specified, for all random number generation. This allows the running of systems that contain random elements (e.g., a Noise Source) while repeating the same "random" sequences *exactly* for each run. When not selected, SystemView uses the time of day (in msec) to generate a new seed for each run.

- **Show Locus in System Root Plots** — When selected, SystemView will show the locus in System root locus plot (see chapter 15).
- **Faster user response during run (Default On) —** When selected, SystemView releases control of the Windows operating system more frequently, allowing a quicker response time for pending tasks such as keyboard input.
- **Noise Temperature (deg K) (Default 300 K)** — This is the environment temperature specified in degrees Kelvin, used in thermal noise calculations in the RF\Analog library).
- **Allow Warning Messages (Default On)** — When selected, SystemView warns of unusual conditions that may be present in the system, such as input tokens with two or more inputs at different sample rates. It is recommended that this option remain enabled.
- **Short Format For Data List Sinks (Default On)**. — SystemView *always* performs its numerical computations using full double precision (64 bits). When the short format display option is selected, the double precision data is displayed in a shortened format (to conserve space) in Data List Sinks.
- **Alert When Simulation Complete (Default Off)**. When selected, SystemView will provide notification, by using an audible alarm and a message, when the simulation run is complete.

Features



☐ Audio Effects

☐ Auto Create Backup When File is Opened. (filename_Backup)

Optimized Screen Saver

☑ Enable Optimized Screen Saver

Activate in: 5 Min

File Insurance

☑ Enable File Insurance

Save every: 10 Min

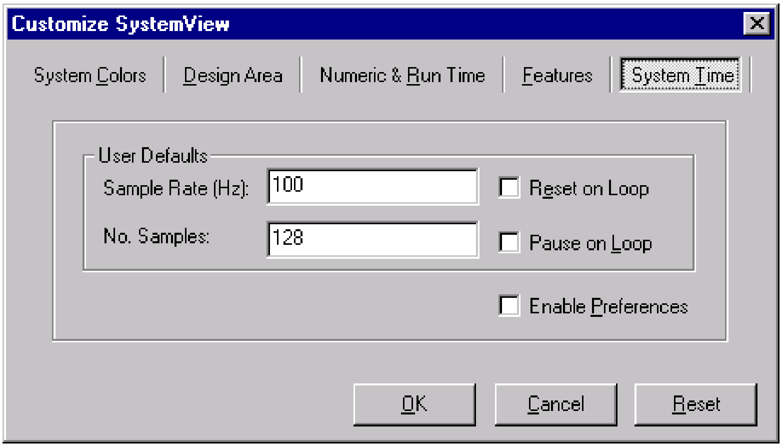
- **Audio Effects (Default On)** — When disabled, start up music is shut off.

- **Auto Create Backup When File is Opened** — When enabled, SystemView will automatically create a backup file of a system when it is first opened.

- **Enable Optimized Screen Saver (Default On)** — When executing a long simulation, the existing commercial screen saver will activate. Screen savers are set for times when the system is idle, and use a lot of CPU time. This can slow down a simulation. To avoid this interference, SystemView provides a built-in screen saver that is optimized to be compatible with long simulations. It also displays the progress of the simulation. The activate time is the time that passes without activity of keyboard/mouse, before screen saver activates.

- **Enable File Insurance (Default On)** — When enabled, SystemView will automatically back up the current system at the interval specified in the Save Every box. *SystemView never overwrites an original file*. If PC power is lost, or a less than graceful exit from Windows occurs (i.e., a crash), the next time SystemView is run, it will ask if you want to recover the last system file. It is recommended that this option remain enabled.

System Time



User Defaults

Sample Rate (Hz): 100

No. Samples: 128

□ Reset on Loop

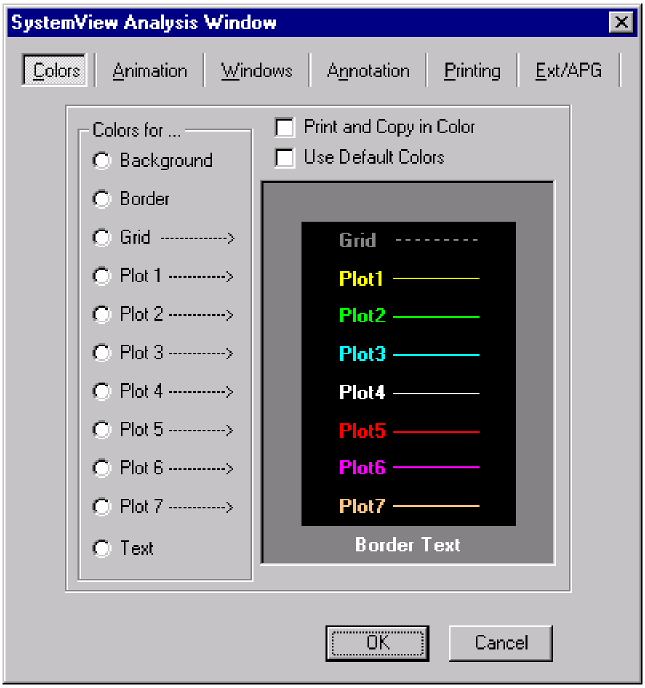
□ Pause on Loop

□ Enable Preferences

- **Sample Rate (Default 100 Hz)** — See Chapter 4 for more information on the sample rate.
- **No. Samples (Default 128)** — See Chapter 4 for more information on the sample rate.
- **Reset On Loop (Default Off)** — See Chapter 4 for more information on the sample rate.
- **Pause On Loop (Default Off)** — See Chapter 4 for more information on the sample rate.
- **Enable Preferences (Default Off)** — When enabled, this option automatically sets the default system time values when you first click the Time button in the System window toolbar.

19.2 Analysis Window

Colors



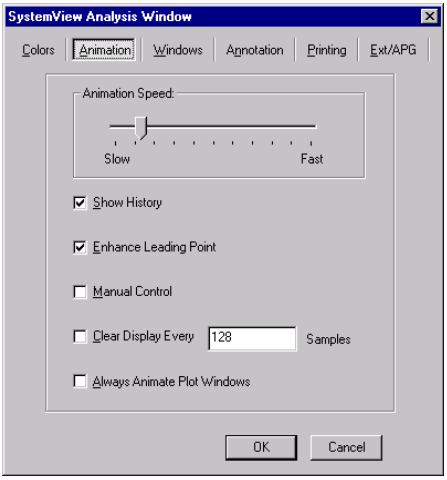
□ Print and Copy in Color

□ Use Default Colors

Select the item to be changed, from the list on the left. A color selection palette will open. Choose the desired color and click the OK button.

- **Print and Copy In Color (Default Off)** — Select this option *only* if a color printer is available and printing or copying SystemView data in color is desired. Both printing and copying to the Windows clipboard are affected by this option.
- **Use Default Colors (Default On)** — Select this option to return to the default colors.

Animation



☑ Show History

☑ Enhance Leading Point

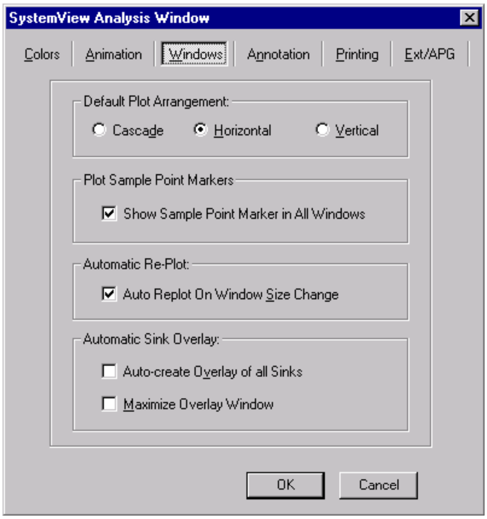
☐ Manual Control

☐ Clear Display Every 128 Samples

☐ Always Animate Plot Windows

- **Animation Speed Scroll Bar** — Use for animation run speed.
- **Show History (Default On)** — When this option is selected, each point plotted will remain on the screen. Otherwise, each point is erased before the next one is plotted.
- **Enhance Leading Pt. (Default ON)** — When selected, the leading point of the plot is made larger, to allow following plot animation
- **Manual Control (Default Off)** — When this option is selected, the plot animation proceeds in one of two ways. By tapping the space bar. It will step through each of the points in the plot. Alternatively, holding the space bar down will cause points in the animation to continue to plot, until the bar is released.
- **Clear Display Every number of samples** — The display is cleared every number of samples selected by the user.
- **Always Animate Plot Windows (Default Off)** — When selected, all plot windows will be animated whenever they are redrawn, when the zoom or Reset Scale button is clicked. Otherwise, a plot window is animated only when the animate button is clicked.

Windows



Default Plot Arrangement:

○ Cascade ● Horizontal ○ Vertical

Plot Sample Point Markers

☑ Show Sample Point Marker in All Windows

Automatic Re-Plot:

☑ Auto Replot On Window Size Change

Automatic Sink Overlay:

□ Auto-create Overlay of all Sinks

□ Maximize Overlay Window

- **Plot Sample Point Markers - Show Sample Point Marker in All Windows (Default On)** — Displays sample point markers for each data point in each plot.

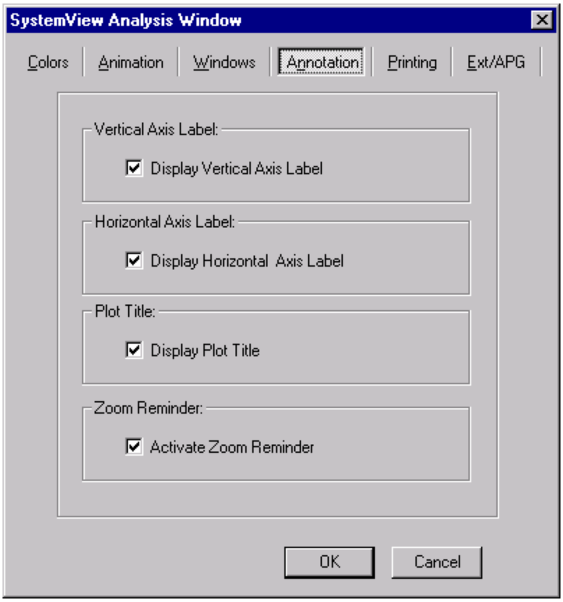
- **Cascade, Horizontal, or Vertical Plot Arrangement (Default Vertical)** — This selection determines the window arrangement on the screen when the Open All toolbar button is clicked, and in other situations when SystemView is ordering the open plot windows.

- **Auto Re-plot On Window Size Change (Default On)** — A plot window will be automatically redrawn whenever the window size is changed (press the Esc key to halt any re-plot). If this option is not selected, click the Reset Scale toolbar button to fit the plot to the new window size.

- **Auto Create Overlay of all Sinks (Default Off)** — When selected, the Sink Calculator will automatically create a single plot window displaying all Sink data. Disabling this option will save time and free system resources.

- **Maximize Overlay Window (Default Off)** — If the Auto Create Overlay option above is selected, this option will maximize the overlay plot window.

Annotation



Vertical Axis Label:

☑ Display Vertical Axis Label

Horizontal Axis Label:

☑ Display Horizontal Axis Label

Plot Title:

☑ Display Plot Title

Zoom Reminder:

☑ Activate Zoom Reminder

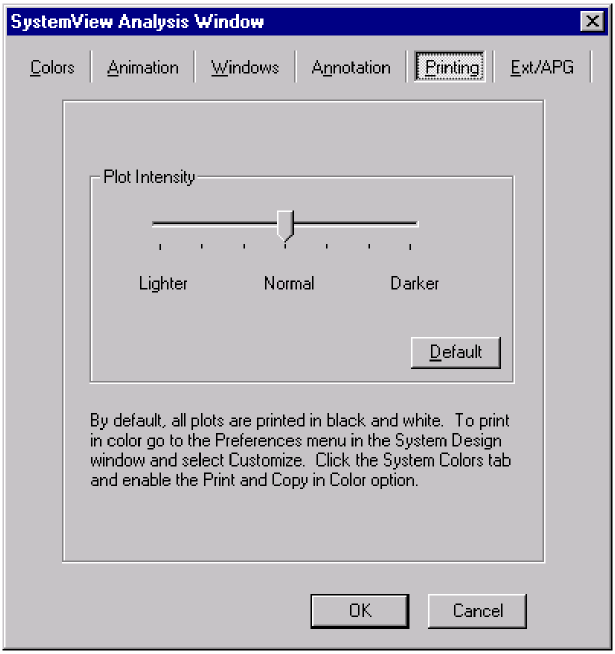
- **Display Vertical Axis label (Default On)** — When selected, the vertical axis (*y*-axis) label will be displayed. (**TIP**: *Double click the axis label to enter a custom label*.)

- **Display Horizontal Axis label (Default On)** — When selected, the horizontal axis (*x*-axis) label will be displayed. (**TIP**: *Double click the axis label to enter a custom label*.)

- **Display Plot Title (Default Off)** — When selected, the plot title will be displayed. (**TIP**: *Double click the title to enter a custom title*.)

- **Activate Zoom Reminder (Default On)** — When selected, a pop-up reminder appears in any scaled plot when a simulation is executed and re-plotted.

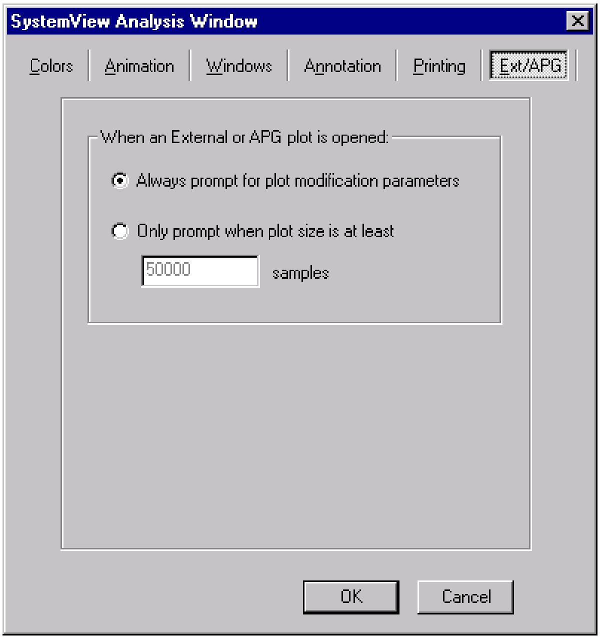
Printing



By default, all plots are printed in black and white. To print in color go to the Preferences menu in the System Design window and select Customize. Click the System Colors tab and enable the Print and Copy in Color option.

- **Plot Intensity** — This option is used to adjust the intensity of the data waveform when printed. It does not affect other features of the plot.

Ext/APG



When an External or APG plot is opened:

● Always prompt for plot modification parameters

○ Only prompt when plot size is at least

50000 samples

This option allows the user to have full control of how many samples will be opened in the analysis window from an external or an APG generated plot. Enabling this option allows the user to minimize the amount of memory needed to view results.

Chapter 20. Technical Support

Eagleware-Elanix pays special attention to knowing the SystemView product and its features, and takes pride in the quality of our SystemView technical support. All of the support engineers are experienced in the application areas for which SystemView is used; DSP, signal processing, controls, communication systems, and applied mathematics. Several of the SystemView application/support engineers have advanced degrees in their area of expertise. The support needed is often more than a requirement to know where to find a token or how to access a particular feature. ELANIX engineers can often help to diagnose a design problem, or even contribute to the design itself.

20.1 Accessing Technical Support

- e-mail: support@eagleware.com
- Web: http://www.eagleware.com
- Phone: +1 (678)291-0719, 8:30am to 5:30pm, Eastern Standard Time

20.2 Consulting Services

Eagleware-Elanix offers professional consulting services in DSP, signal processing, and communication systems on a time and materials basis. We also perform SystemView instructional seminars at sites that wish to conduct short courses in the use, techniques, and applications of SystemView. Contact Eagleware-Elanix for information about these services.

Appendix A. Selected *SystemView* Examples

A.1 Hermite Polynomial

File: Hermpoly.svu

This SystemView example solves the differential equation,

$$\frac{d^2 x}{dt^2} + \left(\lambda - t^2\right)x = 0$$

In quantum mechanics, this is Schroedinger's equation for a harmonic oscillator. The parameter λ can have only discrete values of the form 2*n*+1, with *n* being an integer. In this example, *n*=10.

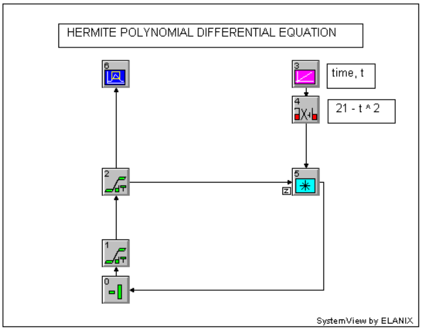


Figure A.1-1. Hermite polynomial in SystemView

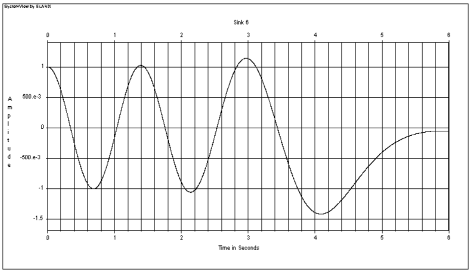


Figure A.1-2. Hermite polynomial from quantum mechanics

Order	Token	Name	Group	Parameters
0		Negate	Gain/Scale Operators	None
1		Integral	Integral / Differential	(1) Zero Order (2) Initial Condition (default) = 0
2		Integral	Integral / Differential	(1) Zero Order (2) Initial condition = 1

3		Time	Aperiodic / External Sinks	Gain =1
				Offset =0

4	X+	Polynomial	Algebraic Functions	X^5 Coeff=0 X^2 Coeff=-1 X^4 Coeff=0 X^1 Coeff=0 X^3 Coeff=0 X^0 Coeff=21
5	X	Multiplier		
6		Analysis	Analysis / Export	

Table A.1-1. Tokens used in Hermpoly.svu

A.2 Chip Rate Detector

File: chiprate.svu

This SystemView example implements a delay-and-multiply type chip rate detector used to detect PN-type spread spectrum signals. Mathematically, it is performing the operation,

$$z(\tau, f) = \int_0^T s(t)s^*(t-\tau)e^{2\pi jft}\,dt$$

The signal is on a 21.4 MHz carrier. After the I/Q down-conversion, the sampling rate can be reduced to the information bandwidth (tokens 14, 15). The FFT is performed using the Sink Calculator in the Analysis window.

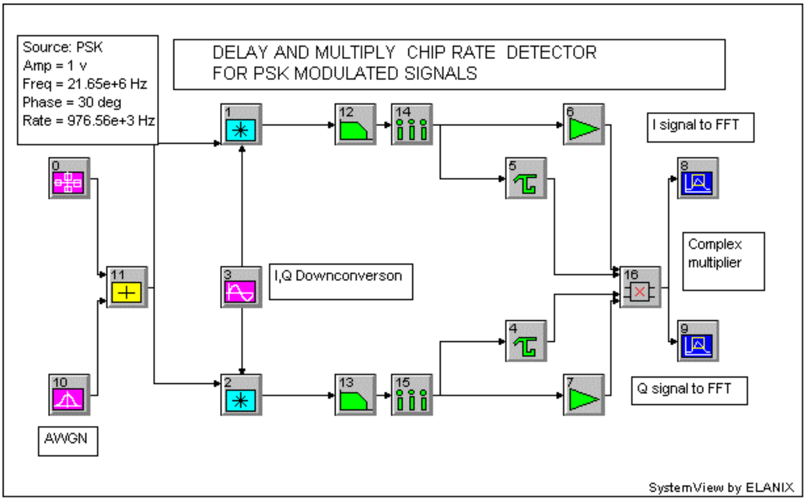


Figure A.2-1. Chip rate detector

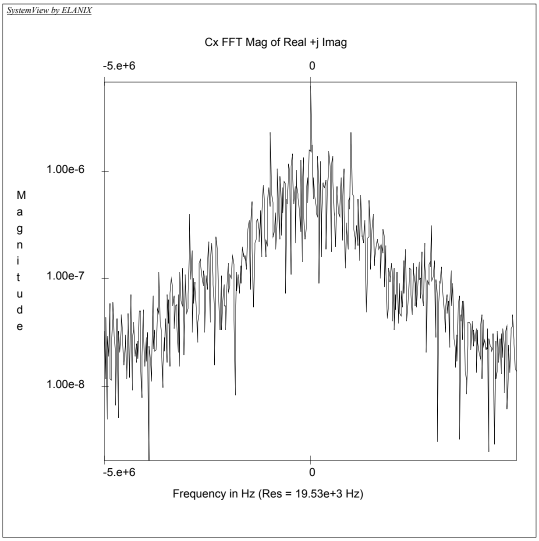


Figure A.2-2. Output of FFT showing rate lines at odd multiples of 976.56 KHz

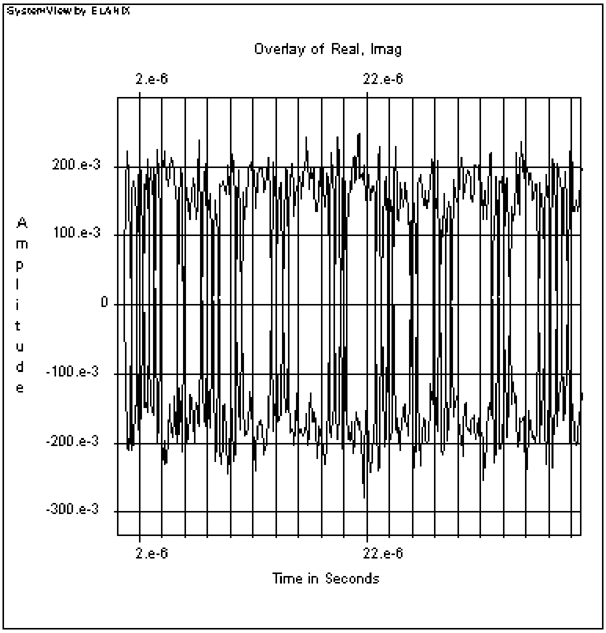


Figure A.2-3. Chip rate results

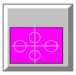
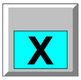
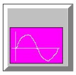
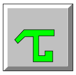
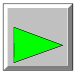
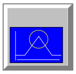
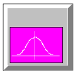
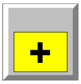
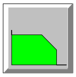
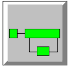
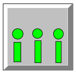
Order	Token	Name	Group	Parameters
0		PSK Carrier	Sinusoid / Periodic	(1) Amplitude = 1 (2) Frequency (Hz) = 21.65. (3) Carrier Phase (deg) = 30 (4) Symbol Rate (Hz) = 976.56e+3 (5) Number of Symbols = 2
1,2		Multiplier		
3		Sinusoid	Sinusoid / Periodic	(1) Amplitude (v) = 1 (2) Frequency (Hz) = 21.4e+6. (3) Phase (deg) = 0
4,5		Delay	Delay	(1) Time Delay (sec) = 499.999999 e-9 (2) Type: Non-Interpolating
6,7		Gain	Gain / Scale	(1) Gain Value = 1 (2) Gain Units: Linear
8,9		Analysis	Analysis / Export	
10		Gauss Noise	Noise / PN	(1) Standard Deviation = 300 e-3 (2) Mean = 0 (3) Constant Para: Std Deviation
11		Adder		
12,13		Linear Sys	Filters / Linear Systems	Several parameters (see the linear system window  and Chapter 6)
14,15		Sampler	Sample / Hold	(1) Sample Rate (Hz) = 10e+6 (2) Sample Aperture Duration (sec) = 0 (3) Aperture Jitter (sec) = 0 (4) Selection: Interpolating

Table A.2-1. Summary of chip rate token types

A.3 Single Frequency LMS Canceller Loop

File: lms.svu

This SystemView example implements a single frequency canceller adaptive loop employing the LMS algorithm. The loop adjusts the weight of the sine and cosine of the reference frequency to bring the composite signal (the sum of the outputs of MetaSystem tokens 6 and 17) 180 degrees out of phase with the input signal (token 0). Since the adaptive architecture for each weight is identical, they are implemented as MetaSystems (tokens 6 and 17).

The first plot shows the loop adapting to an input sine wave offset in phase. The weights converge to the proper values and the error goes to zero.

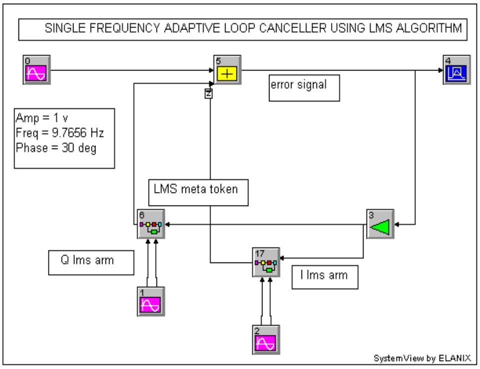


Figure A.3-1. LMS Loop

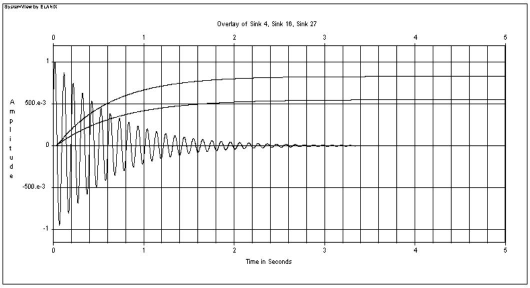


Figure A.3-2. Operation of LMS Loop
showing error signals and tap weight convergence

A.4 Blind Equalizer

File: equal.svu

This SystemView example implements a seven-tap blind equalizer that derives the error signal by a method developed by Lucky. Since each arm of the equalizer is identical in form, it is implemented via a MetaSystem. This significantly reduces the effort required to create the simulation, and it also reduces the Token count in the System window as shown. The output graphic shows the weights adapting toward these theoretical values. The channel has a transfer function:

$$H(z) = 1 + 0.5z^{-1}$$

The equalizer must then have a transfer function:

$$T(z) = \frac{1}{H(z)} = 1 - 0.5z^{-1} + 0.25z^{-2} + \ldots$$

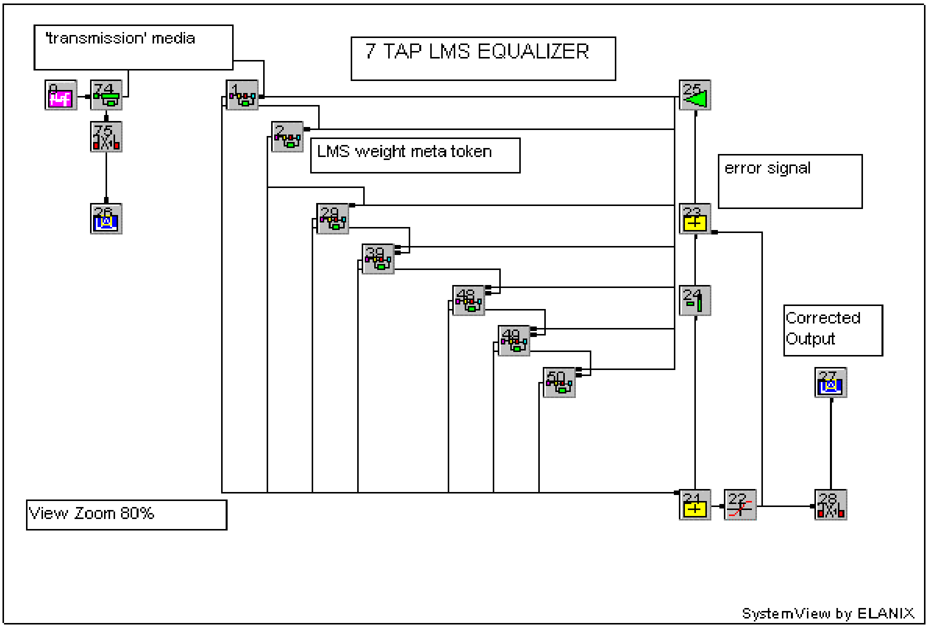


Figure A.4-1. Equalizer

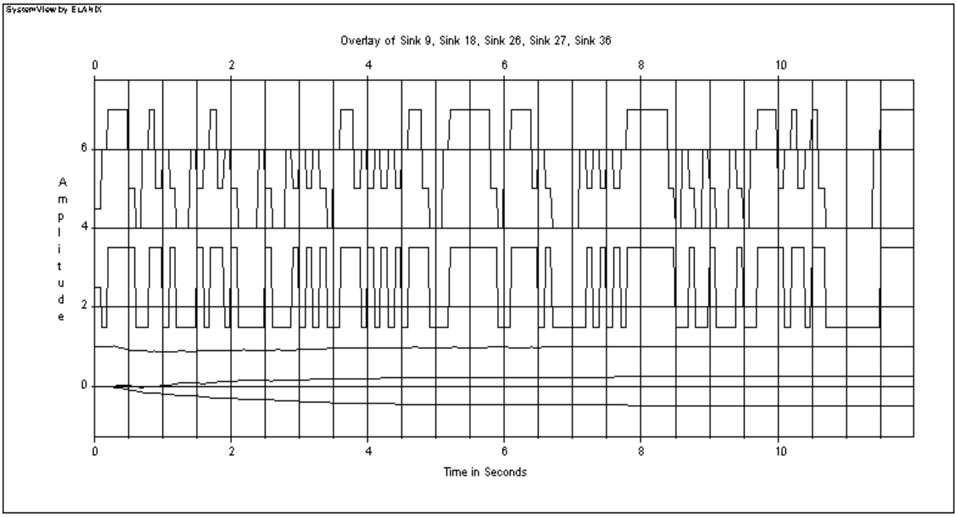


Figure A.4-2. Overlaid results

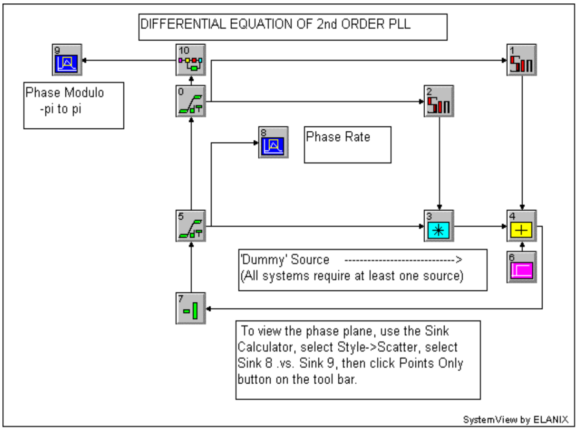
A.5 Second Order PLL Phase Plane Trajectory

File: pll.svu
This SystemView example implements the solution of the differential equation,

$$\frac{d^2\phi}{dt^2} + \cos\phi \frac{d\phi}{dt} + a\sin\phi = 0$$

Which represents the operation of a second-order PLL.

An interesting presentation of the behavior of this system is to use phase-plane trajectories, which plot the frequency error vs. the phase error. The output Sinks 8 and 9 are the individual time histories. Use the Sink Calculator to create a two-dimensional scatter plot of the two output variables. SystemView will allow you to plot any combination of such output pairs in this format. Also note the “dummy” source used in this example.



DIFFERENTIAL EQUATION OF 2nd ORDER PLL

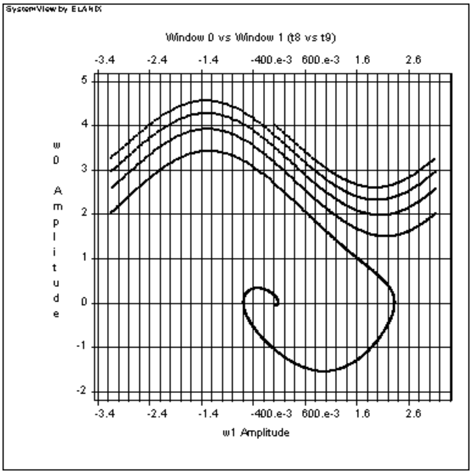
Phase Modulo
-pi to pi

Phase Rate

'Dummy' Source ---------------------------->
(All systems require at least one source)

To view the phase plane, use the Sink Calculator, select Style->Scatter, select Sink 8 .vs. Sink 9, then click Points Only button on the tool bar.

Figure A.5-1. Second-order PLL



A.5-2. Second-order PLL phase-plane diagram

Figure A.5-2 shows the phase-plane diagram. The vertical axis is frequency and the horizontal axis is phase modulo $[-\pi, \pi]$. The loop skips several cycles before locking. To see the dynamics, use the animate feature in the analysis window.

A.6 Modeling An Unknown System

File: model.svu
This example implements an LMS adaptation of an unknown plant. The "unknown" plant and the LMS transversal filter are both driven by white Gaussian noise. In this example, the filter has the exact order as the unknown plant. The graphic shows the filter weights adapting to their correct values, and the corresponding reduction in the error signal.

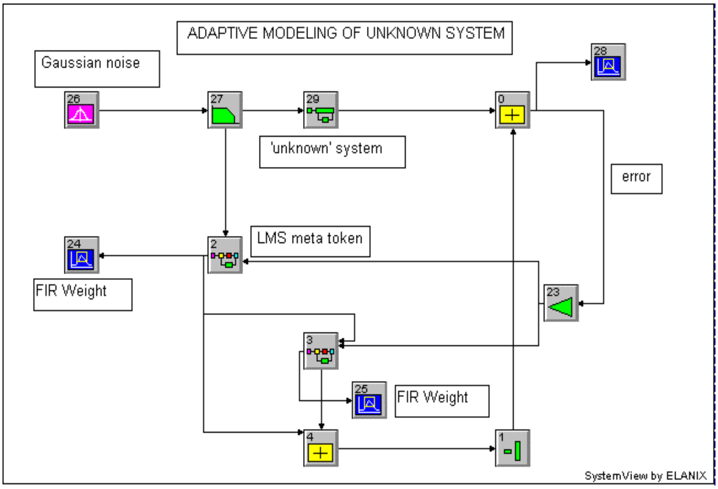


Figure A.6-1. Modeling an unknown system

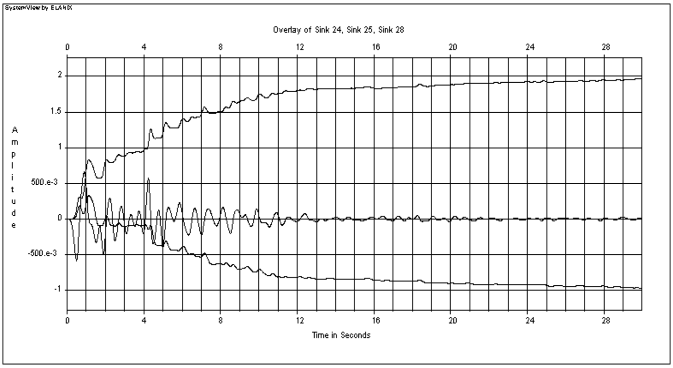


Figure A.6-2. LMS adaptation to unknown plant

A.7 Generation of 16 QAM

File: qam.svu
this simple SystemView example generates a 16-QAM (quadrature amplitude modulation) system by separately modulating a sine and cosine with independent, four-level amplitudes, and forming the sum at token 7. Tokens 1 and 2 down-convert the composite signal back into baseband I and Q waveforms. Linear System tokens 3 and 4 filters out the double frequency term.

The resulting demodulated baseband waveforms are shown in the first graphic. Using the SystemView scatter plot capability, the I and Q can be plotted against each other producing the familiar 16-QAM constellation. A small amount of Gaussian noise (token 15) has been added to make the constellation more visible.

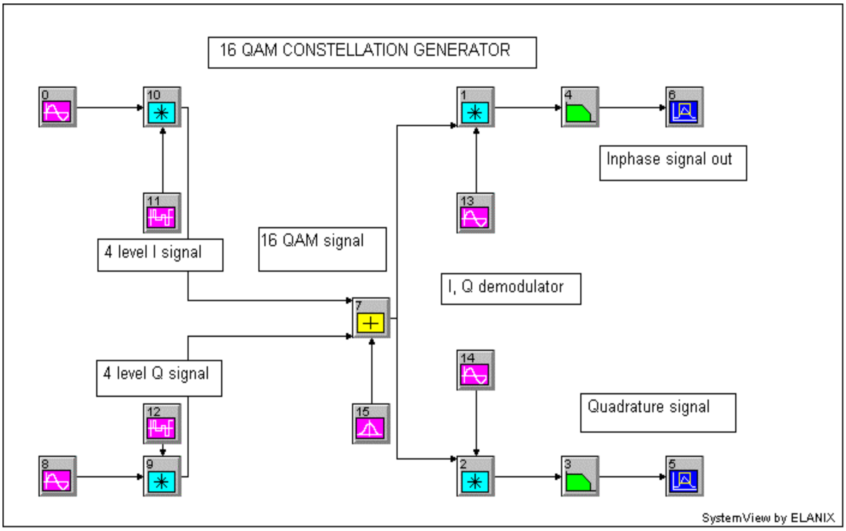


Figure A.7-1. QAM

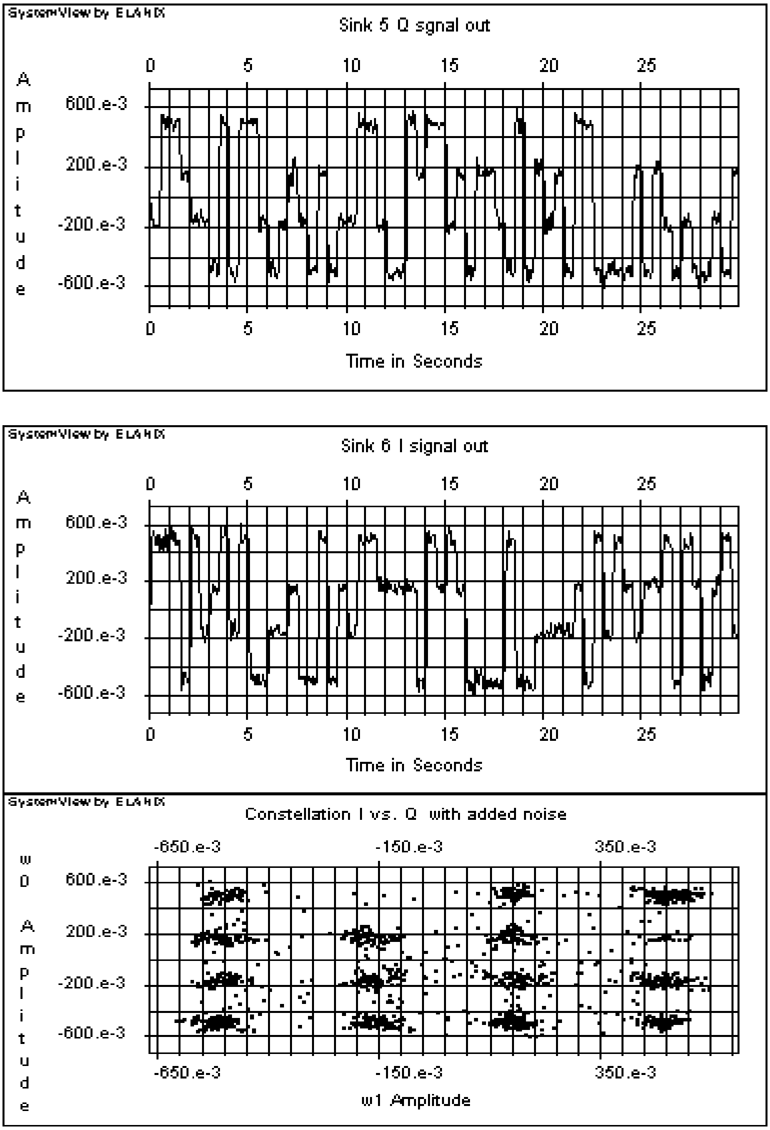
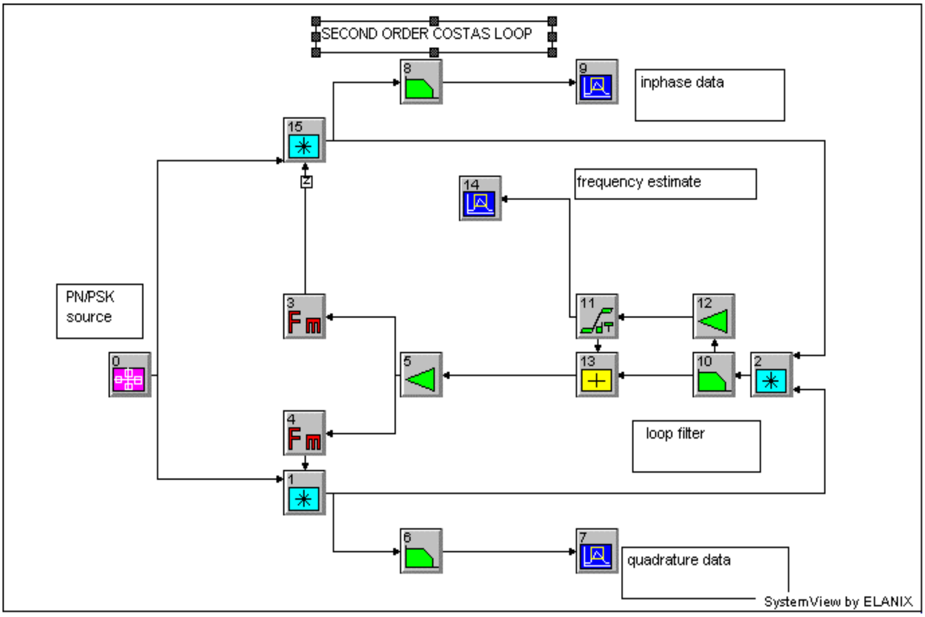


Figure A.7-2. I, Q and Constellation

A.8 Second-Order Costas Loop

File: costas.svu
This SystemView example implements a second-order Costas demodulator for a PN/PSK signal. The input is a binary PSK signal having a phase and frequency different than the reference loop. The generated error signal (token 2) is the product of the I and Q signals. For random binary PSK, this product is statistically independent of the data.

The graphic shows the I and Q signal as the loop locks. First, the system skips cycles while the frequency is pulled in, and then the phase locks. The result is a baseband I signal having all of the signal power, and a Q signal locked out.



A.8-1. Costas Loop

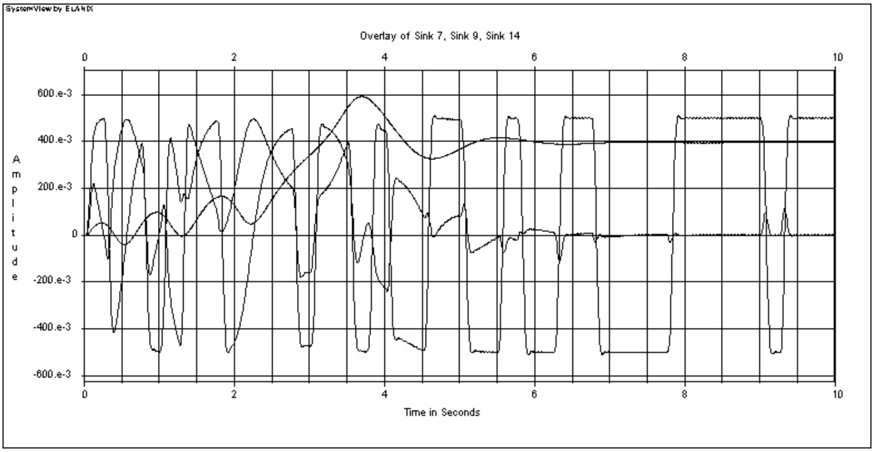
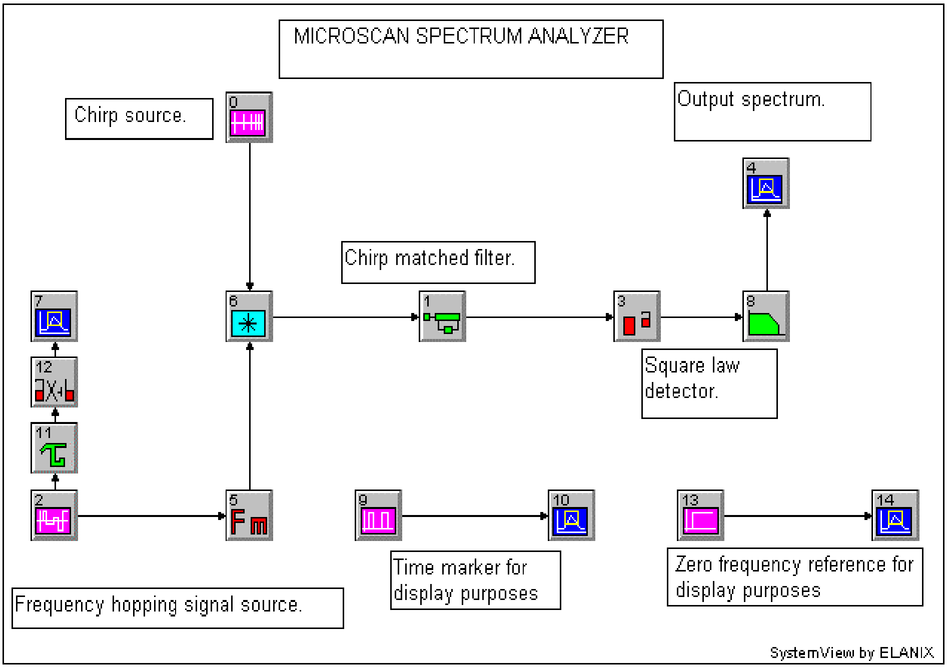


Figure A.8-2. Costas Loop tracking in phase and frequency

A.9 Microscan Receiver

File: microscn.svu
the microscan receiver, or wideband spectrum analyzer, is well known in signal processing and military electronic systems. It is based on the unique properties of a linear FM chirp waveform. The key attribute is that a shift in frequency at the input translates to a shift in time in the processed output. The basic components are (1) the chirp source, and (2) a linear matched filter for the chirp.

In this example, the input signal is a frequency-agile signal (or frequency-hopped signal, FH). The hop time is synchronized with the chirp sweep time. During each hop the instantaneous frequency is mixed with the chirp source. This has the effect of shifting the apparent frequency excursion of the chirp. The resultant signal is passed through the chirp-matched filter. The time corresponding to the correlation peak of the matched filter is linearly related to the frequency shift. The correlation peak then moves back and forth in time in proportion to the frequency shift. This is, of course, a spectrum analyzer. The amplitude variation in the output is due to the fact that the shifted chirp does not completely "fill" the matched filter at the correlation time. The greater the frequency shift, the greater the amplitude loss.


MICROSCAN SPECTRUM ANALYZER
Chirp source.
0
Output spectrum.
4
Chirp matched filter.
7
6
1
3
8
12
Square law detector.
11
2
5
9
10
13
14
Time marker for display purposes
Zero frequency reference for display purposes
Frequency hopping signal source.
SystemView by ELANIX

A.9-1. Microscan receiver in SystemView

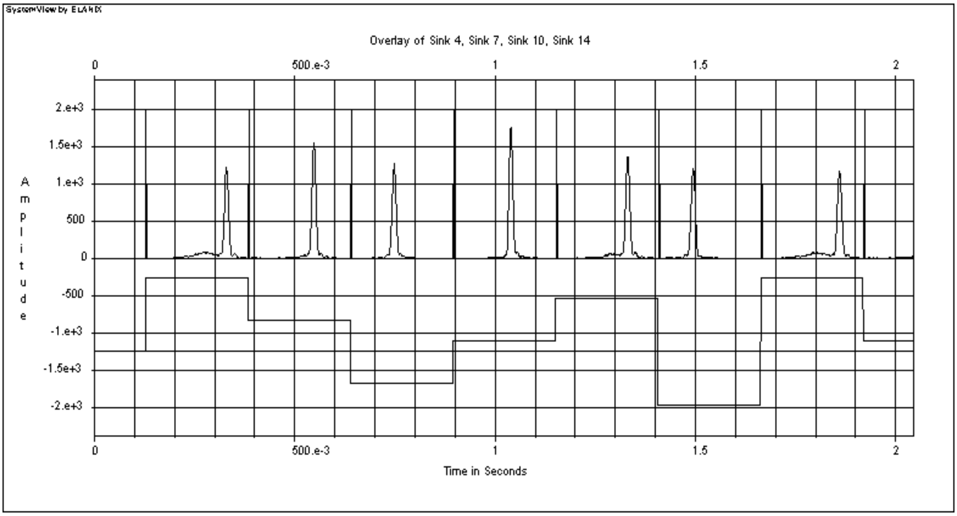


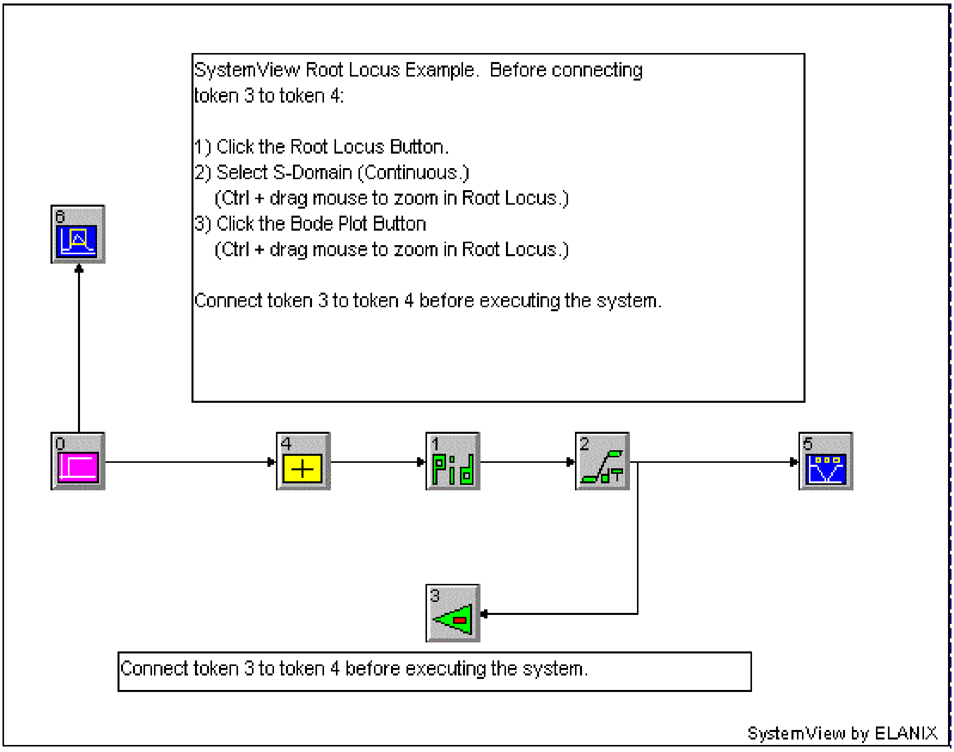
Figure A.9-2. Microscan analyzer showing FH characteristic

A.10 Root Locus

File: rlocus.svu
this example system shows the use of the root locus and Bode plot features in SystemView. Note that the feedback connection between tokens 3 and 4 has been broken. This allows SystemView to compute the open loop transfer function describing the Root Locus and Bode plots for the connected system.

Figure A.10-2 is the resulting root locus, where the zoom feature has been used to enable examination of behavior near the origin.

Figure A.10-3 is the Bode plot for the same system.



SystemView Root Locus Example. Before connecting token 3 to token 4:

1) Click the Root Locus Button.
2) Select S-Domain (Continuous.)
   (Ctrl + drag mouse to zoom in Root Locus.)
3) Click the Bode Plot Button
   (Ctrl + drag mouse to zoom in Root Locus.)

Connect token 3 to token 4 before executing the system.

Connect token 3 to token 4 before executing the system.

Figure A.10-1. Root locus example

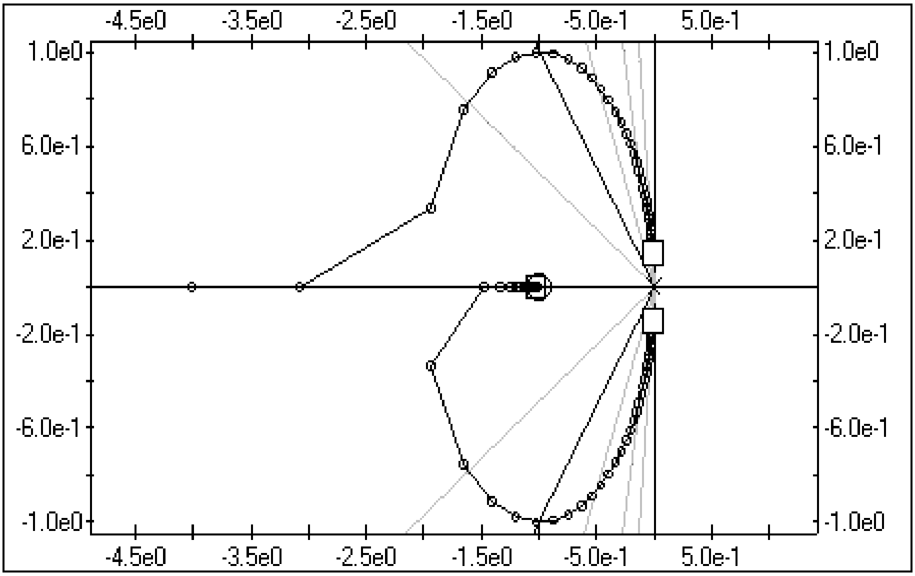


Figure A.10-2. Open loop root locus (zoomed)

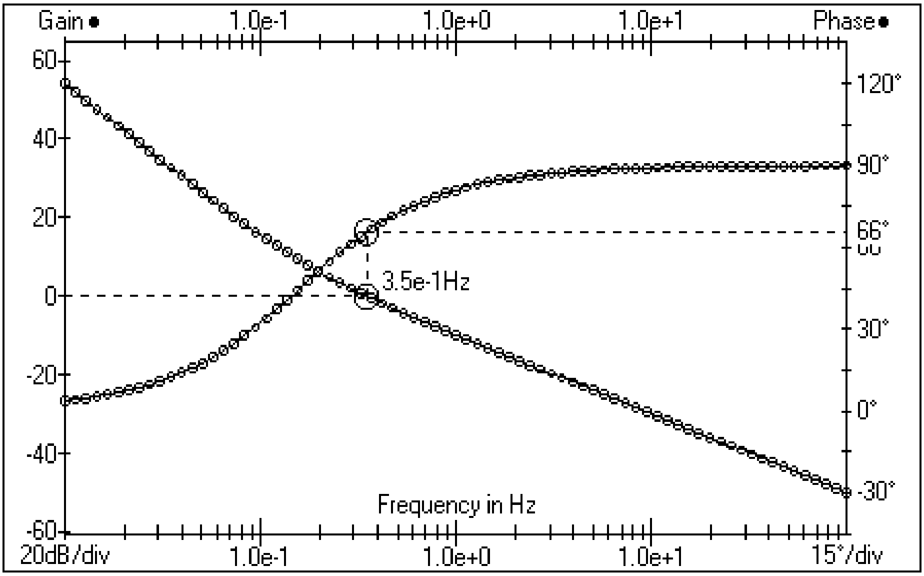
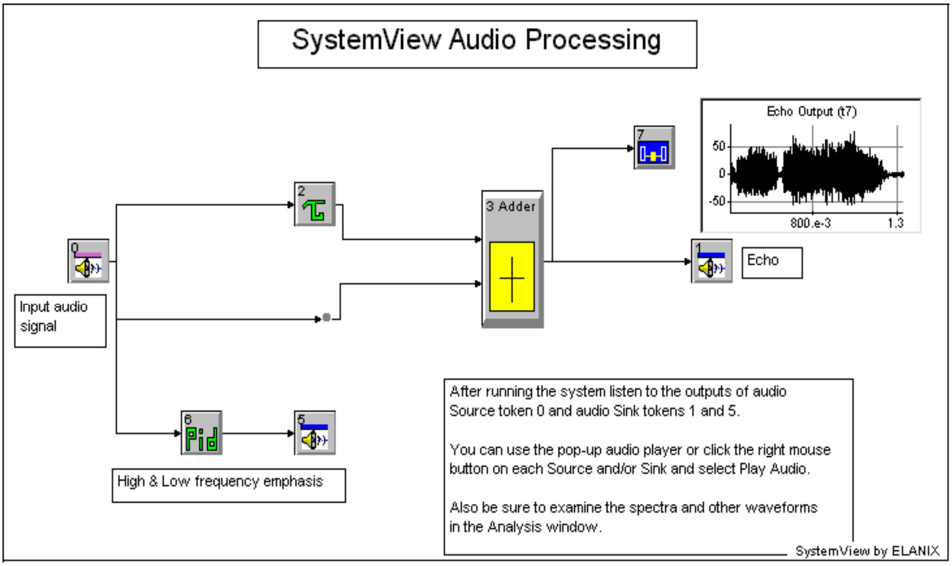


Figure A.10-3. Open loop Bode plot

A.11 Audio System

File: Audio.svu
This example system shows how to use the special audio Sources and Sinks. SystemView supports both one and two channel (monaural and stereo), and 8-bits or 16-bits per sample. For Sources the external file must be a WAV file. The Audio Sinks output their data in the standard WAV format.

After running this example, you can listen to the input and output audio signals using the pop-up audio players, or by clicking the right mouse button and selecting “Play Audio”.



SystemView Audio Processing

After running the system listen to the outputs of audio Source token 0 and audio Sink tokens 1 and 5.

You can use the pop-up audio player or click the right mouse button on each Source and/or Sink and select Play Audio.

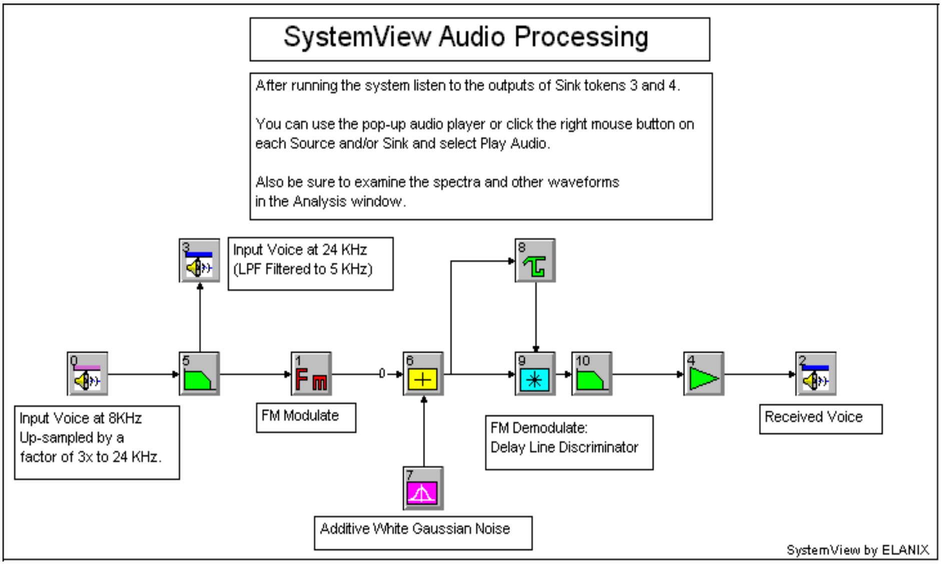
Also be sure to examine the spectra and other waveforms in the Analysis window.

A.12 Audio FM System

File: AudioFM.svu
this example system shows the special audio Sources and Sinks used in a FM transmit and receive system. The FM demodulation is performed using a delay line discriminator.

SystemView supports both one and two channel (monaural and stereo), and 8-bits or 16-bits per sample. For Sources, the external file must be a WAV file. The Audio Sinks output their data in the standard WAV format.

After running this example, you can listen to the input and output audio signals using the pop-up audio players, or by clicking the right mouse button and selecting “Play Audio”.



SystemView Audio Processing

After running the system listen to the outputs of Sink tokens 3 and 4.

You can use the pop-up audio player or click the right mouse button on each Source and/or Sink and select Play Audio.

Also be sure to examine the spectra and other waveforms in the Analysis window.

Appendix B. Help Topics and Shortcuts

The following provides various Help Topics, and Keyboard/Mouse shortcuts. **Note**: Unspecified references to "click," refer to the left mouse button, right button clicks are noted.

- **Animation** — In the Analysis window select Options under the Preferences menu. Select Animation tab and set the desired parameters. Select a plot window and click on the Animate button. Animation is informative with scatter plots.
- **Auto Analysis Sink** — Press the Alt + Ctrl keys and click the desired location in the system for placement of a new automatically-defined Analysis Sink.
- **Automatic windowing before spectra and FFTs** — In the Analysis window, select Auto Window under the Preferences menu. You can then specify the desired window to be automatically applied.
- **Averaging spectral data** — In the Sink Calculator select the Operator tab and click Time Slice. Perform the Spectrum operation on the Time Sliced plot window. Select the Algebra tab in the Sink Calculator and click Average.
- **Changing plot values** — In the Analysis window move the mouse to a plot sample (the x-y display will turn red). Double click the mouse to edit the sample.
- **Complementary group delete** — After clicking the Delete button, click right mouse button and drag to delete all tokens *outside* the box.
- **Copying a system into the current system** — Launch a new MetaSystem token and double click the new token. Select any SystemView file. Select Explode MetaSystem under the Tokens menu and click the new MetaSystem.
- **Creating MetaSystem** — Click the Create MetaSystem button. Then click/drag the mouse to select a group of tokens.
- **Custom plot captions** — In the Analysis window, double click a plot caption to edit the caption.

- **Deleting a token** — As an alternative to clicking the Delete button on the toolbar, you can drag a token back to the Token Reservoir and drop it on the appropriate generic token.
- **Duplicating groups of tokens** — After clicking the Duplicate button, click the mouse button and draw a window around the group of tokens and their connections, to be duplicated. Then drag duplicated tokens to appropriate area.
- **Dynamic Parameter Changes** — With the System Probe enabled, and while the system is running, click any token in your system with the right mouse button, select Edit Parameters to dynamically change the token parameters.
- **Editing plot data** — In the Sink Calculator, select the Data tab. You can pad zeros, extract, append, insert, and delete plot data.
- **Undo Last Operation**— In the System Design Window, this feature allows you to undo any previous edit to your system.
- **Undo Last Undo**— Allows an undo to the previous undo to the system
- **Expanded screen** — In the Analysis window, enlarge a plot window by dragging its border. Move the expanded window partially off screen. Use the Analysis scroll bars to view the expanded virtual screen.
- **Exporting Plot Graphics to a File** — In the Analysis window, select Export as Metafile under the File menu. This command allows you to export the active plot window as a fully scaleable Windows metafile (.wmf).
- **Exporting Plot Graphics** — In the Analysis window, select Copy as Metafile under the Edit menu. This allows you to copy the active plot window to the clipboard as a fully scaleable Windows metafile. In the target application, select Paste under the Edit menu.
- **Fast disconnect** — For a token with several input/output connections, click the Disconnect button. Now, click the mouse button and draw an outline around the token. All I/O for that token will be disconnected

- **Fast token parameters** — Click any defined token with the right mouse button, and then select Edit Parameters. This skips the Library window, and goes directly to the Parameter window.
- **Fast zoom in the Analysis window** — Click and draw a window around area to be zoomed.

- **FIR filter gain** — While in the Linear System window or the Custom FIR window, selecting Force Gain, allows you to force a FIR filter to have a specific gain at a specific frequency.
- **Group delete** — Click the delete button, and draw a window around tokens to be deleted
- **Linear System Window** — Click the right mouse button within the plot to customize the background and foreground colors, copy, or print the current plot. Click the left mouse button and drag to zoom.
- **Moving all tokens** — Click and drag the token.
- **Moving token groups --** Click the mouse button and draw a window around any group of tokens. Then move the window accordingly.
- **Multiple plot zoom** — In the Analysis window, click the left mouse button and drag to zoom open plot windows to the same scale.
- **Note Pads** — Click right mouse button twice, to change Note Pad fonts, font color, and background color. The options are also available under the NotePads pull-down menu
- **Optional Libraries** — The optional library tokens simplify system development and allow SystemView to run faster than designs using equivalent MetaSystems.
- **Plot scrolling** — After zooming a plot in the Analysis window, click the left mouse button and drag within a plot to expose different views of your data.
- **Professional Maintenance** — With the SystemView professional maintenance program you receive technical support and FREE upgrades when they are released.

- **Quick Printing** — Press Print Screen button on the keyboard to copy the screen to the Windows clipboard. Press Alt and Print Screen keys to copy the active plot window in the Analysis window. Paste the image in the word processor or other program and print.
- **Random parameter variations** — Select Global Links under Tools menu, select un (uniform) or gn (Gaussian) under System Variables list, and then type algebraic relationship.
- **Real Time and SystemView Sinks** — Click the right mouse button to customize the background and foreground colors, copy, or print the current plot. Size the Sink using the size controls.
- **Retaining only the last value of a simulation run** — Use the Final Value Sink to retain the final (or last) value from a run.
- **Saving MetaSystems** — Select Save MetaSystem under the File menu, then click the MetaSystem token.
- **Shortcut for Connecting Tokens** — Press Ctrl key and click the From and To tokens.
- **Shortcut for Disconnecting Tokens** — Press the Shift key and click the From and To tokens.
- **System Probe** — Click the Pause button (| |) on the Probe control panel, to pause the system run and view the current Sink data in the Analysis window. Click the Go button (>) to resume the run.
- **System Probe** — Use the probe Slider to make rough changes to the time/freq span. For finer changes, if the Slider has the focus, use the Page Up, Page Down, and Arrow keys.
- **System Probe** — While the system is running, press and hold the Ctrl key. Then, click/drag the mouse to zoom the Probe time display.
- **System Probe** — You can pre-set the probe to any token before the system is run. Click and drag the Probe button at the bottom of the screen and drop the probe on the desired token. Now run the system.
- **System Probe** —You can size the probe by moving the mouse to the probe border. When the mouse arrow pointer appears, click and drag to the desired size. The probe plot will rescale only while the system is running.
- **System Probe** — You can view the output of tokens located in MetaSystem with the System Probe. Click the desired MetaSystem and then click any token within that MetaSystem.

- **Viewing System Time** — Place the mouse on the Define System Time icon, or press the left mouse button anywhere in the system design area. System time is displayed in the information panel at the upper right hand corner of the screen.
- **Viewing token inputs and outputs** — Place mouse pointer on any token. The input and output token numbers for that token will be displayed in the information panel at the lower left corner of the screen.
- **Viewing token parameters** Place mouse pointer over a system token to display a pop up parameter window.