DSP System Toolbox™ Release Notes

How to Contact MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

DSP System Toolbox™ Release Notes

© COPYRIGHT 2012–2013 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See `www.mathworks.com/trademarks` for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see `www.mathworks.com/patents` for more information.

Contents

R2013b

Support Package for ARM Cortex-M Processors 2
Channel and distortion measurement, cursors, and spectrogram visualization using Spectrum Analyzer in MATLAB and Simulink 3
Channel mapping for multichannel audio devices in MATLAB and Simulink 3
Variable-size support for FIR and Allpole filters in MATLAB and Simulink 3
Estimation of Power Spectrum, Cross Power Spectrum, and Transfer Function for streaming data in MATLAB 4
Data logging and archiving using Time Scope in Simulink 4
MIDI control interface support in MATLAB 4
Integer support on the output port of the MIDI Controls block 5
Kalman filter 5
Adaptive filters using Lattice, Fast Transversal, Filtered-X LMS, and Frequency Domain algorithms in MATLAB . . 5
Coupled allpass filter 6
Functionality being removed or changed 6
Configuration dialog added to Logic Analyzer 8
Complex trigger support in Time Scope 8
Default color changes for Array Plot, Time Scope, and Spectrum Analyzer 8
MATLAB System Block to include System objects in Simulink models 8
Restrictions on modifying properties in System object Impl methods 8
System objects matlab.system.System warnings 10
Removing HDL Support for NCO Block 10

R2013a

Allpass Filter System object 14

Adaptive filter System objects using RLS and Affine Projection Filter 14
Logic Analyzer System object 14
Audio System object support for tunability, variable frame size, variable number of channels, and writing MPEG-4 AAC 15
Array Plot System object for displaying vectors or arrays in 2-D and Spectrum Analyzer block with enhanced controls and features such as peak finder 16
Time Scope block with triggering and peak finder features 23
Change of the default for audio hardware API on Linux . . 25
Change of the default for audio file formats in multimedia blocks and audio file reader and writer System objects 25
Change of property default in the audio file reader System object 25
Removal of the `signalblks` package 25
Scope Snapshot display of additional scopes in Simulink Report Generator 26
Unoriented vector treated as column vector in the Biquad Filter 26
NCO HDL Optimized block 26
`HDLNCO` System object 26
HDL code generation for NCO HDL Optimized block and System object 27
Support for nonpersistent System objects 27
New method for action when System object input size changes 27
Scaled double data type support for System objects 27

R2012b

`SpectrumAnalyzer` System object 30
Cross-platform support for reading and writing WAV, FLAC, OGG, MP3 (read only), MP4 (read only), and M4a (read only) 31
Support for code generation for `CICDecimator` and `CICInterpolator` System objects 32
Support for HDL code generation for multichannel Discrete FIR Filter block 32

Time Scope enhancements, including new cursors, embedded simulation controls, and External and Rapid Accelerator modes 32
Source and sink blocks being replaced 37
Discrete IIRFilter and AllpoleFilter System objects 38
Support for MATLAB Compiler for `CICDecimator` and `CICInterpolator` System objects 38
Code generation support for `SignalSource` System object 39
Behavior change of locked System objects for loading, saving, and cloning 39
Behavior change of statistics blocks for variable-size inputs 40
Simulation state save and restore for additional blocks . . . 40
For Each subsystem support for additional blocks 41
Multi-instance model referencing support for additional blocks 41
Expanded analysis support for filter System objects 42
Removal of the signalblks package 42
Discrete filter block visible in DSP library 42
System object tunable parameter support in code generation 42
save and load methods for System objects 43
Save and restore SimState not supported for System objects 43
Map integer delay to RAM on Delay block 43
HDL support for System objects 44
HDL resource sharing for Biquad Filter block 44

R2012a

Frame-Based Processing 48
System Object Enhancements 51
Time Scope Enhancements 53
ASIO Support in To/From Audio Device Blocks and Objects 57
Video Processing Enabled for the DSP System Toolbox Multimedia File Blocks 57
System Objects Integrated into Filter Design Workflow . . 58
New Measurement Workflow 59
Discrete FIR Filter System Object 60

Inverse Dirichlet Sinc-Shaped Passband Design Added to Constrained FIR Equiripple Filter 60
Code Generation Support Added to FIR Decimator System Object 60
Filter Block Enhancements 61
Discrete FIR Filter Block Coefficient Port Changes 61
Statistics Blocks and Objects Warning for Region of Interest Processing 61
New and Updated Demos 62

R2011b

Frame-Based Processing 64
Custom System Objects 69
New Allpole Filter Block 69
New Audio Weighting Filter Functionality 70
Time Scope Enhancements 70
New Arbitrary Group Delay Design Support 71
Arbitrary Magnitude Responses Now Support Minimum Order and Minimum/Maximum Phase Equiripple Design Options 71
Support for Constrained Band Equiripple Designs in MATLAB and Simulink 71
New Sinc Frequency Factor and Sinc Power Design Options for Inverse Sinc Filters 72
New Inverse Sinc Highpass Filter Designs 72
Filterbuilder and dspfdesign Library Blocks Now Support Different Numerator and Denominator Orders for IIR Filters 73
New Stopband Shape and Stopband Decay Design Options for Equiripple Highpass Filter Designs 73
FFTW Library Support for Non-Power-of-Two Transform Length 73
MATLAB Compiler Support for dsp.DigitalDownConverter and dsp.DigitalUpConverter 73
Complex Input Support for dsp.DigitalDownConverter . . . 74
getFilters Method of dsp.DigitalDownConverter and dsp.DigitalUpConverter Now Return Actual Fixed-Point Settings 74
dsp.SineWave and dsp.BiquadFilter Properties Not Tunable 74

System Object DataType and CustomDataType Properties Changes 75
System Objects Variable-Size Input Dimensions 75
Conversion of Error and Warning Message Identifiers . . . 75
New and Updated Demos 77
Blocks Being Removed in a Future Release 77

R2011a

Product Restructuring 80
Frame-Based Processing 80
New Function for Changing the System Object Package Name from signalblks to dsp 90
New Discrete FIR Filter Block 91
New Printing Capability from the Time Scope Block 91
Improved Display Updates for the Time Scope Block and System Object 91
New Implementation Options Added to Blocks in the Filter Designs Library 91
New dsp.DigitalDownConverter and dsp.DigitalUpConverter System Objects 92
Improved Performance of FFT Implementation with FFTW library 93
Variable-Size Support for System Objects 93
System Objects FullPrecisionOverride Property Added . . . 94
'Internal rule' System Object Property Value Changed to 'Full precision' 95
MATLAB Compiler Support for System Objects 96
Viewing System Objects in the MATLAB Variable Editor 96
System Object Input and Property Warnings Changed to Errors 96
New and Updated Demos 97
Documentation Examples Renamed 97
Downsample Block No Longer Has Frame-Based Processing Latency for a Frame Size of One 98
SignalReader System Object Accepts Column Input Only 98
FrameBasedProcessing Property Removed from the dsp.DelayLine and dsp.Normalizer System Objects 99
R2010a MAT Files with System Objects Load Incorrectly 99

R2013b

Version: 8.5

New Features: Yes

Bug Fixes: No

Support Package for ARM Cortex-M Processors

The DSP System Toolbox™ Support Package for ARM® Cortex®-M Processors allows you to model your signal processing algorithm in Simulink® and generate C code. The generated code can be linked with the CMSIS library, and compiled to provide an executable to run on ARM Cortex-M processors. To use the DSP System Toolbox Support Package for ARM Cortex-M Processors, you must have the following products in addition to the DSP System Toolbox: Simulink, Simulink Coder™ , Embedded Coder®, and MATLAB® Coder. The following DSP System Toolbox blocks, which support this library, make it optimal for use in the ARM Cortex-M processors.

- Discrete FIR Filter
- FIR Decimation
- FIR Interpolation
- LMS Filter
- Biquad Filter
- FFT
- IFFT
- Correlation
- Convolution
- Mean
- RMS
- Variance
- Standard Deviation

To download and install this feature, select **Add-Ons > Get Hardware Support Packages** on the MATLAB Toolstrip. Then, use Support Package Installer to install the DSP System Toolbox Support Package for ARM Cortex-M Processors. For more information, see “Support Package for ARM Cortex-M Processors”.

Channel and distortion measurement, cursors, and spectrogram visualization using Spectrum Analyzer in MATLAB and Simulink

To enhance visualizing your data, channel measurements, distortion measurements, cursor measurements, and a spectrogram view have been added to the Spectrum Analyzer. Channel measurements show occupied bandwidth and adjacent channel power ratio (ACPR) measurements. Distortion measurements show harmonic distortion and intermodulation distortion measurements. Cursor measurements show measurements between two cursors. A spectrogram is a display of the frequencies in a signal over time.

Channel mapping for multichannel audio devices in MATLAB and Simulink

The `dsp.AudioPlayer` and `dsp.AudioRecorder` System objects, and the To Audio Device and From Audio Device blocks, now support channel mapping. The term *channel mapping* is used to refer to a 1-1 mapping that associates channels on the selected audio device to channels of the data.

When you play audio, channel mapping allows you to specify on which channel of the audio device to output a specific channel of audio data. You can specify channel mapping as a vector of output channel indices corresponding to each output channel of data being written.

When you record audio, channel mapping allows you to specify on which channel of the audio data to input a specific channel of audio device. You can specify channel mapping as a vector of audio channel indices corresponding to each channel of data being read.

Variable-size support for FIR and Allpole filters in MATLAB and Simulink

In this release, `dsp.AllpoleFilter`, `dsp.FIRFilter`, Discrete FIR Filter block, and Allpole Filter block support variable-size input. Thus the number of rows (size of the frame) can vary.

Estimation of Power Spectrum, Cross Power Spectrum, and Transfer Function for streaming data in MATLAB

This release introduces `dsp.TransferFunctionEstimator`, `dsp.SpectrumEstimator`, and `dsp.CrossSpectrumEstimator`. The transfer function estimator, estimates the complex frequency-domain transfer function from time-domain data, based on the Welch averaged periodogram method. `dsp.TransferFunctionEstimator` provides functionality similar to the Signal Processing Toolbox function `tfestimate`, albeit in a streaming-friendly manner. The PSD and Cross-PSD estimators, are also provided using the Welch averaged periodogram method, functionality similar to the Periodogram block in DSP System Toolbox. These three System objects support double- and single-precision floating point inputs. They also support C code generation.

Data logging and archiving using Time Scope in Simulink

Data logging and external mode data archiving have been added to the Time Scope block. You can now log scope data to a MAT-file.

MIDI control interface support in MATLAB

R2013b introduces these five functions that together provide the same functionality as that of the MIDI Controls block:

- `midiid` — Interactively identify a MIDI control.
- `midicontrols` — Open a group of MIDI controls for reading.
- `midiread` — Read the most recent values of the group of MIDI controls.
- `midisync` — Send values to update the group of MIDI controls.
- `midicallback` — Invoke a callback when an open control changes.

Among these functions, only `midicontrols`, `midiread`, and `midisync` support code generation.

Integer support on the output port of the MIDI Controls block

The MIDI Controls block now supports the `uint8` data type on the output port, for the range of 0 to 127, selecting the **raw MIDI** mode.

Kalman filter

This release introduces `dsp.KalmanFilter`. This filter supports C/C++ code generation, single- and double-precision floating point, MIMO, and optional control input. It also includes a subset of functionality in the corresponding block, Kalman Filter, including:

- Initial condition for estimated state
- Initial condition for estimated error covariance
- State transition matrix
- Process noise covariance
- Measurement matrix
- Measurement matrix noise covariance
- Output estimated measurement
- Output estimated state
- Multiple parallel filters
- Disable update on a subset of filters

Adaptive filters using Lattice, Fast Transversal, Filtered-X LMS, and Frequency Domain algorithms in MATLAB

The following adaptive filters are introduced in this release:

- `dsp.AdaptiveLatticeFilter`
- `dsp.FastTransversalFilter` (no code generation)
- `dsp.FilteredXLMSFilter` (no code generation)

- `dsp.FrequencyDomainAdaptiveFilter`

They have the following features:

- C/C++ code generation support
- Floating-point data type support (double and single) for inputs
- Variable frame-size for inputs
- Real and complex inputs

Coupled allpass filter

This release introduces the `dsp.CoupledAllpass` filter, which implements IIR filters as the sum of two allpass filters operating in parallel. This filter allows you to use complex coefficients. It supports `Variable-size` input, floating-point filter analysis, and the filter coefficients are tunable. You can integrate this filter with filter design workflows such as `fdesign` and `filterbuilder`.

Functionality being removed or changed

Compatibility Considerations: Yes

Using `fdesign.pulseshaping` is discouraged because it is being removed in the future. However, it still runs when you try to use this functionality, but you are encouraged to use `rcosdesign` and `gaussdesign` instead.

Migrate away from `fdesign.pulseshaping`

Using `fdesign.pulseshaping` is not recommended. Use `rcosdesign` or `gaussdesign`. This table shows examples of how to replace the old function with the new functions.

fdesign.pulseshaping	rcosdesign and gaussdesign

`sps=6;`
`span=4;`
`Beta=fdesign.pulseshaping(sps,...`
`'Square Root Raised Cosine',...`
`'Nsym,Beta',span,Beta);`
`d1=design(f1);`
`n1=d1.Numerator`
	`n1n=rcosdesign(Beta,span,sps);`
	`n1n=n1n/max(n1n)*(-1/(pi*sps)...`
	`* (pi*(Beta-1) -4*Beta))`

`g1=fdesign.pulseshaping(sps,...`
`'Square Root Raised Cosine',...`
`'N,Beta',sps*span,Beta);`
`h1=design(g1);`
`k1=h1.Numerator`
	`k1n=rcosdesign(Beta,span,sps);`
	`k1n=k1n/max(k1n)*(-1/(pi*sps)...`
	`*(pi*(Beta)-4*Beta))`

`f2=fdesign.pulseshaping(sps,...`
`'Raised Cosine, ...`
`'Nsym,Beta',span,Beta);`
`d2=design(f2);`
`n2=d2.Numerator`
	`n2n=rcosdesign(Beta,span,sps,'normal');`
	`n2n=n2n/max(abs(n2n))/sps`

`g2=fdesign.pulseshaping(sps,...`
`'Raised Cosine',...`
`'N,Beta',sps*span,Beta);`
`h2=design(g2);`
`k2=h2.Numerator`
	`k2n=rcosdesign(Beta,span,sps,'normal');`
	`k2n=k2n/max(abs(k2n))/sps`

`BT=0.3;`
`f3=fdesign.pulseshaping(sps,...`
`'Nsym,BT',span,BT);`
`d3=design(f3);`
`n3=d3.Numerator`
	`n3n=gaussdesign(BT,span,sps)`
	`'Gaussian',...`

Configuration dialog added to Logic Analyzer

A Visuals — Logic Analyzer Properties dialog box has been added to the `dsp.LogicAnalyzer`. This dialog box allows you to change the appearance of the scope. To open this dialog box, select **View > Configuration Properties**.

Complex trigger support in Time Scope

The Time Scope can plot signals as real/imaginary or magnitude/phase. In addition to using triggers from the real or imaginary data, you can now use magnitude or phase values from complex signals as triggers.

Default color changes for Array Plot, Time Scope, and Spectrum Analyzer

The following scopes use a new default color scheme to help emphasize the data being visualized:

- `dsp.ArrayPlot`
- `dsp.SpectrumAnalyzer` and Spectrum Analyzer block
- `dsp.TimeScope` and Time Scope block

MATLAB System Block to include System objects in Simulink models

The MATLAB System block is a new block in the Simulink User-Defined Functions library. Use this block to create a Simulink block that includes a System object™ in your model. This capability is useful for including your algorithm in your model.

Restrictions on modifying properties in System object Impl methods

Compatibility Considerations: Yes

When defining a new System object™, certain restrictions affect your ability to modify a property.

You cannot use any of the following methods to modify the properties of an object:

- `cloneImpl`
- `getDiscreteStateImpl`
- `getDiscreteStateSpecificationImpl`
- `getNumInputsImpl`
- `getNumOutputsImpl`
- `getOutputDataTypeImpl`
- `getOutputSizeImpl`
- `isInputDirectFeedthroughImpl`
- `isOutputComplexImpl`
- `isOutputFixedSizeImpl`
- `validateInputsImpl`
- `validatePropertiesImpl`

This restriction is required by code generation, which assumes that these methods do not change any property values. These methods are validation and querying methods that are expected to be constant and should not impact the algorithm behavior.

Also, if either of the following conditions exist:

- You plan to generate code for the object
- The object will be used in the MATLAB System block

you cannot modify tunable properties for any of the following runtime methods:

- `outputImpl`
- `processTunedPropertiesImpl`
- `resetImpl`
- `setupImpl`
- `stepImpl`
- `updateImpl`

This restriction prevents tunable parameter updates within the object from interfering with updates from outside the generated code. Tunable parameters can only be changed from outside the generated code.

Compatibility Considerations

If any of your class definition files contain code that changes a property in one of the above `Impl` methods, move that property code into an allowable `Impl` method. Refer to the System object `Impl` method reference pages for more information.

System objects matlab.system.System warnings
Compatibility Considerations: Yes

The System object base class, `matlab.system.System`, has been replaced by `matlab.System`. If you use `matlab.system.System` when defining a new System object, a warning message results.

Compatibility Considerations

Change all instances of `matlab.system.System` in your System objects code to `matlab.System`.

Removing HDL Support for NCO Block
Compatibility Considerations: Yes

HDL support for the NCO block will be removed in a future release. Use the NCO HDL Optimized block instead.

Compatibility Considerations

In the current release, if you generate HDL code for the NCO block, a warning message appears. In a future release, any attempt to generate HDL code for the NCO block will cause an error.

R2013a

Version: 8.4

New Features: Yes

Bug Fixes: No

Allpass Filter System object

This release introduces an Allpass Filter System object™, `dsp.AllpassFilter` , which unifies functionality already existing in a number of dfilt objects under various names. `dsp.AllpassFilter` supports three different allpass structures and it can handle both single-section and cascaded configurations. It also supports double and single floating point, multichannel and variable-length input, tunability of filter coefficients and filter analysis.

Adaptive filter System objects using RLS and Affine Projection Filter

This release introduces two adaptive filter System objects, `dsp.RLSFilter` and `dsp.AffineProjectionFilter`. These System objects both support double and single floating point, and code generation.

Logic Analyzer System object

As of R2013a, DSP System Toolbox software provides a new Logic Analyzer System object that enables you to view the transitions of signals. To create a Logic Analyzer System object variable called *h*, at the MATLAB command prompt, type `h=dsp.LogicAnalyzer`. The Logic Analyzer has several graphical features:

- Multiple signals in a single window — The *y*-axis of the display can contain a number of channels, vertically tiled on top of each other.
- Ability to vary the display style — You can modify the name, height, color, and font size of each wave.
- Analog and Digital display formats — Both discrete and continuous signals can appear as waves or be tiled vertically in the display.
- Data numerical display options — You can display numerical values in various numeric systems, including unsigned decimal, signed decimal, hexadecimal, octal, and binary form. Such flexibility is especially useful for visualizing fixed-point signals.

- Cursors to mark points of interest and view values — By placing a cursor at a discrete time stamp, you can position a solid vertical line on the display and observe the values of each channel at that time stamp.
- Dividers to delineate groups of waves in a channel — By placing a divider on the display, you can separate waves by horizontal dashed lines.

For more information, see the `dsp.LogicAnalyzer` System object reference topic.

Audio System object support for tunability, variable frame size, variable number of channels, and writing MPEG-4 AAC

Effective 13a, variable frame size and variable number of channels are supported by `dsp.AudioPlayer`. If you use variable-size signals with this System object, you may experience sound dropouts when the size of the input changes. With the added support, you can avoid this behavior. Before you start the simulation, call `setup` for a signal with maximum dimensions.

This release also enhances `dsp.AudioPlayer` and `dsp.AudioRecorder` System objects to make sample rate, buffer size, and queue duration tunable. Tuning these properties stops and restarts the sound card, which creates a pause. The length of the pause depends on your buffer size and queue duration.

In addition, the `dsp.AudioFileWriter` System object and To Multimedia File block are enhanced to support MPEG-4 AAC audio files on Windows® 7, and Mac OS X. You can use both M4A and MP4 extensions. The following platform specific restrictions apply when writing these files:

Windows 7	Mac OS X
• Only sample rates of 44100 and 48000 Hz are supported.	• Only mono or stereo outputs are allowed.
• Only mono or stereo outputs are allowed.	

Windows 7	Mac OS X
• Output data is padded on both front and back of the signal, with extra samples of silence. Windows AAC encoder places sharp fade-in and fade-out on an audio signal, causing the signal to be slightly longer in samples when written to disk.	• Not all sampling rates are supported, although the Mac Audio Toolbox API do not explicitly specify a restriction.
• A minimum of 1025 samples must be written to the MPEG-4 AAC file.	

Array Plot System object for displaying vectors or arrays in 2-D and Spectrum Analyzer block with enhanced controls and features such as peak finder

Compatibility Considerations: Yes

As of R2013a, DSP System Toolbox software provides a new Array Plot System object that enables you to visualize streaming data in two dimensions. Using Array Plot, you can visualize any set of data on the *y*-axis, opposite another set of data on the *x*-axis, labeling the *x*- and *y*-axes anything you choose. To create an Array Plot System object variable called *h*, at the MATLAB command prompt, type `h=dsp.ArrayPlot`. Array Plot contains the following panels.

- **Cursor Measurements** — shows cursors on all the displays. In the **Settings** pane, you can choose either waveform cursors, which are always attached to the signal data, or screen cursors, which may be placed anywhere on the axes. In the **Measurements** pane, you can see the *x*-axis value, *y*-axis value, and other calculated values at the locations of the cursors.
- **Signal Statistics** — displays the maximum, minimum, peak-to-peak difference, mean, median, and RMS values of a selected signal. It also shows the corresponding *x*-axis values at which the maximum and minimum values occur.

- **Peak Finder** — displays *y*-axis maxima and the corresponding *x*-axis values at which they occur. These displays allow you to modify the settings for peak threshold, maximum number of peaks, and peak excursion.

For more information, see the `dsp.ArrayPlot` System object reference topic.

As of R2013a, DSP System Toolbox software provides a new Spectrum Analyzer block to replace the Spectrum Scope block in the Sinks library. Using Spectrum Analyzer, you can view the power spectrum or power spectral density of signals. The graphical interface of the Spectrum Analyzer block resembles that of the `dsp.SpectrumAnalyzer` System object. Spectrum Analyzer contains the following panels.

- **Spectrum Settings** — enables you to modify settings to control how the spectrum is calculated. You can modify such parameters as frequency span, resolution bandwidth, number of spectral averages, and number of FFT points. You can also choose between a one-sided or two-sided spectrum and toggle normal, maximum hold, and minimum hold trace views.
- **Peak Finder** — displays spectral maxima and the corresponding frequencies at which they occur. These displays allow you to modify the settings for peak threshold, maximum number of peaks, and peak excursion.

You can programmatically modify parameters of the Spectrum Analyzer block using MATLAB code. To do so, you can first use the Simulink `get_param` function to get an instance of the `spbscopes.SpectrumAnalyzerConfiguration` class. Then, you can use dot notation or the `get` and `set` commands to modify properties of the Spectrum Analyzer block.

For more information, see the Spectrum Analyzer block reference topic.

Compatibility Considerations

All Simulink models containing Spectrum Scope blocks load with Spectrum Analyzer blocks in R2013a or later. The Spectrum Scope block had several dialog box parameters that do not appear in any Spectrum Analyzer block settings.

- Several options that were available on the Parameters dialog box of the Spectrum Scope block are no longer available or have changed. The parameters of Spectrum Scope map to Spectrum Analyzer parameters in the following manner.

R2012b Spectrum Scope Block Parameters dialog box Tab name	R2012b Spectrum Scope Parameter	R2013a Spectrum Analyzer Change	R2013a Spectrum Analyzer Equivalent Parameter
Scope Properties	**Buffer input** check box	R2013a Spectrum Analyzer does not require that input signals are buffered. Spectrum Analyzer determines the number of samples needed using the value of the **RBW** parameter. Regardless of whether the input is a frame-based or sample-based signal, Spectrum Analyzer calculates the spectrum once it has acquired the requisite number of samples.	For Spectrum Scope blocks in R2012b or earlier models, the equivalent R2013a Spectrum Analyzer **RBW** value is given by the equation: $$RBW = \frac{K \times F_s}{L}$$ In the preceding equation, K is the window constant calculated for a segment length of 1000, F_s is the sample rate of the block, and L is the buffer length. If the input signal to the R2012b Spectrum Scope block was frame-based and the **Buffer input** check box was cleared, then the R2013a Spectrum Analyzer computes the

R2012b Spectrum Scope Block Parameters dialog box Tab name	R2012b Spectrum Scope Parameters	R2013a Spectrum Analyzer Change	R2013a Spectrum Analyzer Equivalent Parameter
			RBW value with L set to the frame size of the input signal.
Scope Properties	**Buffer size** parameter	R2013a Spectrum Analyzer uses the **RBW** parameter to determine the requisite number of samples to calculate the spectrum, instead of using the buffer size or frame length.	For Spectrum Scope blocks in R2012b or earlier models, if the input signal was frame-based and the **Buffer input** check box was selected, then the R2013a Spectrum Analyzer computes the **RBW** value with L set to the value of the **Buffer size** parameter.
Scope Properties	**Buffer Overlap** parameter	R2013a Spectrum Analyzer has an **Overlap %** parameter that is directly related to buffer overlap.	R2013a Spectrum Analyzer will compute its **Overlap %** using the equation: $O_p = {O_l}/{L} \times 100$ In the preceding equation, O_p is **Overlap %** parameter value, O_l is the R2012b Spectrum Scope **Buffer overlap** parameter value, and L is the buffer length.

R2012b Spectrum Scope Block Parameters dialog box Tab name	R2012b Spectrum Scope Parameters	R2013a Spectrum Analyzer Change	R2013a Spectrum Analyzer Equivalent Parameter
Scope Properties	**Window** parameter	R2013a Spectrum Analyzer does not have the `Bartlett`, `Blackman`, `Triang`, or `Hanning` settings.	Spectrum Scope blocks in R2012b or earlier models with a window parameter set to any of these values will have their **Window** parameter set to `Hann` in the R2013a Spectrum Analyzer.
Scope Properties	**Window Sampling** parameter	R2013a Spectrum Analyzer does not have a `Periodic` option. All window sampling is now	n/a
Display Properties	**Persistence** check box — this setting would execute the equivalent of the MATLAB `hold on` command, adding another line for each	This option is not available in the R2013a Spectrum Analyzer, which has replaced this feature with the trace options, **Normal Trace**, **Max Hold Trace**, and **Min Hold Trace**.	Spectrum Scope blocks in R2012b or earlier models with persistence enabled will have their **Max Hold Trace** check box selected in the R2013a Spectrum Analyzer.

R2012b Spectrum Scope Block Parameters dialog box Tab name	R2012b Spectrum Scope Parameters	R2013a Spectrum Analyzer Change	R2013a Spectrum Analyzer Equivalent Parameter
	spectrum computation on the display.		
Display Properties	**Compact Display** check box	There is no equivalent capability in the R2013a Spectrum Analyzer.	n/a
Axis Properties	**Inherit Sample time from input** check box	R2013a Spectrum Analyzer always uses the sample time of the input signal.	n/a
Axis Properties	**Frequency display limits** parameter	R2013a Spectrum Analyzer determines the range of frequencies calculated based on the **Full Span**, **FStart (Hz)**, and **FStop (Hz)** parameters.	If this parameter was set to: • `Auto` — R2013a Spectrum Analyzer selects the **Full Span** check box on the Spectrum Settings panel, Main options pane. ▪ `User-defined` — R2013a Spectrum Analyzer clears the **Full Span** check box on the Spectrum

R2012b Spectrum Scope Block Parameters dialog box Tab name	R2012b Spectrum Scope Parameters	R2013a Spectrum Analyzer Change	R2013a Spectrum Analyzer Equivalent Parameter
			Settings panel Main options pane.
Axis Properties	**Minimum frequency (Hz)** parameter	R2013a Spectrum Analyzer determines the range of frequencies calculated based on the **Full Span**, **FStart (Hz)**, and **FStop (Hz)** parameters.	If the `User-defined` parameter was chosen, then this parameter maps to the R2013a Spectrum Analyzer **FStart (Hz)** parameter.
Axis Properties	**Maximum frequency (Hz)** parameter	R2013a Spectrum Analyzer determines the range of frequencies calculated based on the **Full Span**, **FStart (Hz)**, and **FStop (Hz)** parameters.	If the `User-defined` parameter was chosen, then this parameter maps to the R2013a Spectrum Analyzer **FStop (Hz)** parameter.
Line Properties	**Line visibilities**, **Line styles**, **Line markers**, and **Line colors** parameters	There are no equivalent capabilities in the R2013a Spectrum Analyzer.	Once the simulation has started, you can modify the line styles, markers, and colors using the Style dialog box.

- The R2012b Spectrum Scope allowed you to retain the axes limits over multiple simulations by selecting **Axes > Save Axes Settings**. There is no equivalent capability in the R2013a Spectrum Analyzer. However, you can automatically scale the axes to a specified range using the Tools—Plot Navigation Properties dialog box.

Time Scope block with triggering and peak finder features

As of R2013a, DSP System Toolbox provides the following enhancements to the Time Scope block and the `dsp.TimeScope` System object:

- "Triggers Panel" on page 23
- "Peak Finder Features" on page 24
- "Panning Capability" on page 24
- "Programmatic Access" on page 24
- "Scale Axes Limits After 10 Updates" on page 24

For more information on these enhancements, see the Time Scope block reference topic or the `dsp.TimeScope` System object reference topic.

Triggers Panel

The Time Scope contains a new **Triggers** panel that allows you to pause the display only when certain events occur. You can use the Triggers panel when you want to align or search for events.

- In the **Mode** pane, you choose how often the display should update.
- In the **Source / Type** pane, you choose the type of events on which to stop. Events include edges (rising, falling, or both), pulse width, transition, runt, window, or timeout.
- In the **Levels / Timing** pane, you can set the trigger level and hysteresis value.
- In the **Delay / Holdoff** pane, you can offset the trigger position by a fixed delay or set the minimum possible time between trigger events.

To access the **Triggers** panel, in the Time Scope toolbar, click the Triggers button ( ). Alternatively, in the Time Scope menu, select **Tools > Triggers**.

Peak Finder Features

Effective in R2013a, the peak finder now has the option of displaying more than 10 maxima at once. You can also use the Peaks panel to toggle the labels for each local maximum. This capability allows you to choose whether all the labels show time, value, or both time and value. R2013a also provides the capability to sort in either ascending or descending order by value or by time.

Panning Capability

Effective in R2013a, you can pan in all directions on the Time Scope display.

In the Time Scope toolbar, click the Pan button ( ). Alternatively, in the Time Scope menu, select **Tools > Pan**. Then, click and drag the mouse to the left or right to view a different range of data on the *time*-axis. Click and drag the mouse up or down to see a different Amplitude range on the *y*-axis.

Programmatic Access

Effective in R2013a, you can change the properties of the Time Scope block using MATLAB commands. To do so, you can first use the Simulink `get_param` function to get an instance of the `Simulink.scopes.TimeScopeConfiguration` class. Then, you can use dot notation or the `get` and `set` commands to modify properties of the Time Scope block.

For more information on these enhancements, see the Simulink.scopes.TimeScopeConfiguration class reference topic.

Scale Axes Limits After 10 Updates

Effective in R2013a, you can scale the axes limits of the Time Scope displays soon after the simulation starts. To do so, in the Time Scope menu, select **Tools > Scale Axes Limits After 10 Updates**.

Change of the default for audio hardware API on Linux

In this release, the default for audio hardware API on Linux® has changed from OSS to ALSA.

Change of the default for audio file formats in multimedia blocks and audio file reader and writer System objects

In the To Multimedia File block and the `dsp.AudioFileWriter` System object, the file format default is now WAV. In the From Multimedia File block and the `dsp.AudioFileReader` System object, the file format default is now MP3.

Change of property default in the audio file reader System object

In the `dsp.AudioFileReader` System object, the property `OutputDataType` returns doubles as default.

Removal of the `signalblks` package

Compatibility Considerations: Yes

The `signalblks` package has been removed. Instead, for System object classes, and properties, use the `dsp` package.

Compatibility Considerations

To automatically update the existing code, where the `signalblks` package is used, run `sysobjupdate`. This function updates System object code to work in the current release. The application recursively searches the specified folder and subfolders for MATLAB files that contain renamed System object packages, classes, and properties.

Scope Snapshot display of additional scopes in Simulink Report Generator

Using Simulink Report Generator™ software, you can include snapshots of the display produced by a Scope block in a generated report. The Scope Snapshot component, which inserts images of the Simulink Scope block and XY Graph block, now supports the Time Scope block and Spectrum Analyzer block in DSP System Toolbox software.

Note This feature requires that you have a license for the Simulink Report Generator product.

For more information, see the Simulink Report Generator product documentation.

Unoriented vector treated as column vector in the Biquad Filter

Starting this release, the unoriented vector is treated as a column vector in numerator and denominator coefficient ports.

NCO HDL Optimized block

The NCO HDL Optimized block provides hardware friendly control signals, optional reset port, and an optional external dither input port. It also provides a reset function that resets the phase to its initial value during the sinusoid output generation. In addition, it includes an option to output the internal phase and hardware-friendly control signals including valid in and valid out.

`HDLNCO` System object

The `dsp.HDLNCO` System object, like the `dsp.NCO` System object, generates real or complex sinusoidal signals. In addition, the NCO HDL-optimized System object provides hardware friendly control signals, optional reset signal, and an optional external dither input signal.

HDL code generation for NCO HDL Optimized block and System object

Release R2013a provides HDL code generation support for the new NCO HDL Optimized block and dsp.HDLNCO System object. To generate HDL code, you must have an HDL Coder™ license.

Support for nonpersistent System objects

You can now generate code for local variables that contain references to System objects. In previous releases, you could not generate code for these objects unless they were assigned to persistent variables.

New method for action when System object input size changes

The new `processInputSizeChangeImpl` method allows you to specify actions to take when an input to a System object you defined changes size. If an input changes size after the first call to `step`, the actions defined in `processInputSizeChangeImpl` occur when `step` is next called on that object.

Scaled double data type support for System objects

System objects now support scaled double data types.

R2012b

Version: 8.3

New Features: Yes

Bug Fixes: No

`SpectrumAnalyzer` System object

As of R2012b, DSP System Toolbox software provides a new Spectrum Analyzer System object that enables you to view the power spectrum or power spectral density of signals. To create a Spectrum Analyzer System object variable called *h*, at the MATLAB command prompt, type `h=dsp.SpectrumAnalyzer`. The Spectrum Analyzer has several panels and dialog boxes that allow you to perform the following operations:

- Spectrum Settings panel — The **Spectrum Settings** panel enables you to modify settings to control the manner in which the spectrum is calculated. You can modify such parameters as frequency span, resolution bandwidth, number of spectral averages, and number of FFT points. You can also choose between a one-sided or two-sided spectrum and toggle normal, maximum hold, and minimum hold trace views. To hide or display the **Spectrum Settings** panel, in the Spectrum Analyzer toolbar, select the Spectrum Settings button ( ). Alternatively, in the Spectrum Analyzer menu, select **View > Spectrum Settings**.
- Peak Finder panel — The **Peak Finder** panel displays maxima and the frequencies at which they occur. These displays allow you to modify the settings for peak threshold, maximum number of peaks, and peak excursion. In the Spectrum Analyzer toolbar, click the Peak Finder button ( ). Alternatively, in the Spectrum Analyzer menu, select **Tools > Measurements > Peak Finder**.
- Properties dialog box — The Spectrum Analyzer provides a dialog box that allows you to control the most common properties of a display, including the grid lines, titles, *y*-axis labels and *y*-axis limits. To change options for a display, in the Spectrum Analyzer toolbar, click the Properties button ( ). Alternately, in the Spectrum Analyzer menu, select **View > Properties**. You can also right-click the display, and select **Properties**.
- Style dialog box — The Spectrum Analyzer allows you to customize the style of displays using a Style dialog box. You can change the color of the figure containing the displays, the background and foreground colors of display axes, and properties of lines in a display. To view or modify the line style of the active signal, in the Spectrum Analyzer menu, select **View > Style**. You can also right-click the display and select **Style**.

- Axes Scaling Options — The Spectrum Analyzer enables you with the ability to automatically scale the axes to a specified range. In the Spectrum Analyzer menu, select **Tools > Axes Scaling Options** to modify these settings. To manually scale the axes to the limits you specified, click the Scale Axes Limits button ( ).

For more information, see the dsp.SpectrumAnalyzer System object reference topic.

Cross-platform support for reading and writing WAV, FLAC, OGG, MP3 (read only), MP4 (read only), and M4a (read only)

The `dsp.AudioFileReader` System object, and the From Multimedia File block, now support the following audio file formats on all platforms:

- MP3
- MP4
- M4a
- WAV
- FLAC
- OGG

The `dsp.AudioFileWriter` System object, and the To Multimedia File block, now support the following audio file formats on all platforms:

- WAV
- FLAC
- OGG

Support for code generation for `CICDecimator` and `CICInterpolator` System objects

The following System objects now support code generation in MATLAB via the `codegen` command:

- `dsp.CICDecimator`
- `dsp.CICInterpolator`

To use the `codegen` function, you must have a MATLAB Coder license. See Use System Objects in MATLAB Code Generation for more information.

Support for HDL code generation for multichannel Discrete FIR Filter block

Discrete FIR Filter block accepts vector input and supports multichannel implementation for better resource utilization.

- With vector input and channel sharing option `on`, the block supports multichannel fully parallel FIR, including direct form FIR, sym/antisym FIR, and FIRT. Support for all implementation parameters, for example: multiplier pipeline, add pipeline registers.
- With vector input and channel sharing option `off`, the block instantiates one filter implementation for each channel. If the input vector size is N, N identical filters are instantiated.

For fully parallel architecture option for FIR filters only.

Time Scope enhancements, including new cursors, embedded simulation controls, and External and Rapid Accelerator modes

As of R2012b, DSP System Toolbox provides the following enhancements to the Time Scope block and the `dsp.TimeScope` System object:

- “Cursor measurements panel” on page 33

- "Additional embedded simulation controls" on page 33
- "Support for external mode and rapid accelerator mode" on page 34
- "Properties dialog box" on page 35
- "Axes Maximization" on page 35
- "Automatic calculation of Time Span" on page 36
- "ReduceUpdates property" on page 36
- "Support for conditional subsystems" on page 37

Cursor measurements panel

The Time Scope contains a new **Cursor Measurements** panel that shows cursors on all the Time Scope displays. In the **Settings** pane, you may choose either waveform cursors, which are always attached to the signal data, or screen cursors, which may be placed anywhere on the axes. The **Measurements** pane shows the time, amplitude, and other calculated values at the locations of the cursors. In the Time Scope toolbar, click the Cursor Measurements button ( ). Alternatively, in the Time Scope menu, select **Tools > Measurements > Cursor Measurements**.

For more information, see the Time Scope block reference topic or the `dsp.TimeScope` System object reference topic.

Additional embedded simulation controls

Effective in R2012b, additional embedded simulation controls are available through the Simulation Toolbar and the Simulation Stepping Options dialog box. In previous releases, the Time Scope block featured a Simulation Toolbar. With this toolbar, you could control the progression of increasing simulation time from the Time Scope GUI by clicking the Run, Pause, Stop, and Next Step buttons. In R2012b, the Simulation Stepping Options dialog box provides you with the ability to further control the simulation behavior. This dialog box allows you to enable the button on the Simulation Toolbar to take a Previous Step. Additionally, you can pause the simulation at a specified time, specify previous stepping options, and modify the number of steps for forward and backward movement. To access these controls, from the Time Scope menu, select **Simulation > Stepping Options**. Alternatively, if

previous stepping is disabled, in the Time Scope toolbar, click the Previous Step button. The Simulation Stepping Options dialog box appears.

You can enable Time Scope to show a Previous Step button ( ), which allows you to move the simulation time backward by one time step. In the Simulation Stepping Options dialog box, in the **Previous stepping options** group, select the **Enable Previous Stepping** check box. To test this feature, first run the simulation, and then pause the simulation. When the simulation is paused, you can now click the Previous Step button to regress the simulation back by one time step.

Note This feature is available for the Time Scope block but not for the `dsp.TimeScope` System object.

For more information, see the Time Scope block reference topic.

Support for external mode and rapid accelerator mode

As of R2012b, the Time Scope block supports two additional simulation modes in Simulink, External mode and Rapid Accelerator mode. You can use External mode to tune block parameters in real time and view block outputs in many types of blocks and subsystems. External mode establishes communication between a host system, where the Simulink environment resides, and a target system, where the executable runs after it is generated by the code generation and build process. For more information about External mode, see Host/Target Communication in the Simulink Coder product documentation.

You can use Rapid Accelerator mode as a method to increase the execution speed of your Simulink model. Rapid Accelerator mode creates an executable that includes the solver and model methods. This executable resides outside of MATLAB and Simulink. Rapid Accelerator mode uses External mode to communicate with Simulink. For more information about Rapid Accelerator mode, see Acceleration in the Simulink product documentation.

Note This feature is available for the Time Scope block but not for the `dsp.TimeScope` System object.

For more information about Time Scope, see the Time Scope block reference topic.

Properties dialog box

As of R2012b, the Time Scope block provides a centralized location where you can modify the most important properties of a display. The Properties dialog box contains the most frequently modified Time Scope settings, including all the parameters from the Tools:Plot Navigation Options dialog box and the Visuals:Time Domain Options dialog box. It also includes **Open at Start of Simulation** and **Number of Input Ports** from the **File** menu. To open this dialog box, in the Time Scope toolbar, click the Properties button ( ). Alternately, in the Time Scope menu, select **View > Properties**. You can also right-click on the display and select **Properties**.

Note This feature is available for the Time Scope block but not for the `dsp.TimeScope` System object. In the `dsp.TimeScope` System object GUI, when you select **View > Properties**, the Visuals:Time Domain Options dialog box appears, as in R2012a. This same dialog box also appears when you right-click on the display and select **Properties**.

For more information about Time Scope, see the Time Scope block reference topic.

Axes Maximization

In R2012b, you can specify whether to display the Time Scope block or System object in maximized axes mode. In this mode, the axes are expanded to fill the entire display. In each display, there is no space to show titles or axis labels. The values at the axis tick marks appear on top of the axes. You can select one of the following options:

- `Auto` — In this mode, the axes appear maximized in all displays only if the **Title** and **Y-Axis label** parameters are empty for every display. If you enter any value in any display for either of these parameters, the axes are not maximized.
- `On` — In this mode, the axes appear maximized in all displays. Any values entered into the **Title** and **Y-Axis label** parameters are hidden.
- `Off` — In this mode, none of the axes appear maximized.

The default setting is `Auto`. In the Time Scope GUI, you can set this property in the **Main** pane of the Properties dialog box. To change options using the `dsp.TimeScope` System object, set the `MaximizeAxes` property to the intended option. For more information, see the Time Scope block reference topic or the `dsp.TimeScope` System object reference topic.

Automatic calculation of Time Span

As of R2012b, the Time Scope block can automatically calculate the **Time Span** parameter using the simulation **Start Time** and **Stop Time** parameters. By default, the Time Scope block has the **Time Span** parameter set to **Auto calculate**. To modify the **Time Span** parameter to use a different value, open the Properties dialog box and click the **Time** tab.

Note This feature is available for the Time Scope block but not for the `dsp.TimeScope` System object.

For more information about Time Scope, see the Time Scope block reference topic.

ReduceUpdates property

As of R2012b, the Time Scope System object has an additional property called `ReduceUpdates`. By default, this property is set to `true`. When this property is `true`, the Time Scope updates the displays at a rate not exceeding 20 hertz. When you set this property to `false`, the Time Scope updates every time the `step` method is called. The simulation speed is faster when this property is set to `true`. Using this property is equivalent to selecting the **Reduce**

Updates to Improve Performance check box in the **Simulation** menu of the Time Scope GUI.

For more information about this property, see the `dsp.TimeScope` System object reference topic.

Support for conditional subsystems

In previous releases, the Time Scope block could be used within an enabled subsystem. As of R2012b, the Time Scope block can also be placed in a triggered subsystem, an enabled and triggered subsystem, and a function-call subsystem. For more information about these types of subsystems, see Conditional Subsystems in the Simulink documentation.

Note This feature is available for the Time Scope block but not for the `dsp.TimeScope` System object.

For more information about Time Scope, see the Time Scope block reference topic.

Source and sink blocks being replaced

Compatibility Considerations: Yes

The following Windows platform blocks now map to other existing blocks that work on all platforms:

Deprecated blocks	Blocks mapped to
From Wave Device	From Audio Device
To Wave Device	To Audio Device
From Wave File	From Multimedia File
To Wave File	To Multimedia File

This mapping is transparent; no `slupdate` is needed. When you open an existing model that contains the original blocks, the replacement blocks are automatically substituted. If you save the model, the replacement blocks are saved instead of the original blocks.

Compatibility Considerations

Because mapped blocks do not have identical functionality, incompatibilities can be introduced in certain cases. The From Wave File can have an optional `start-of-file` indicator port, whereas the From Multimedia File cannot. Therefore, if you load an old model where From Wave File has the `start-of-file` port, a broken link results. In this case, a warning message appears, providing a link to start_of_file_example, which shows you how to correct the problem.

Discrete IIRFilter and AllpoleFilter System objects

This release introduces a discrete IIR Filter System object `dsp.IIRFilter` and a discrete Allpole Filter System object `dsp.AllpoleFilter`.

The IIR Filter System object implements the algorithm, inputs, and outputs described on the Discrete Filter block reference page. The object properties correspond to the block parameters. Both this object and its corresponding block let you specify whether to process inputs as individual samples or as frames of data. This System object supports code generation.

The Allpole Filter System object implements the algorithm, inputs, and outputs described on the Allpole Filter block reference page. The object properties correspond to the block parameters. Both this object and its corresponding block let you specify whether to process inputs as individual samples or as frames of data. This System object supports code generation.

Support for MATLAB Compiler for `CICDecimator` and `CICInterpolator` System objects

The following System objects are now supported by MATLAB Compiler™:

- `dsp.CICDecimator`
- `dsp.CICInterpolator`

For more information, see Using System Objects with MATLAB Compiler.

Code generation support for `SignalSource` System object

`dsp.SignalSource` now support code generation in MATLAB via the `codegen` command:To use the `codegen` function, you must have a MATLAB Coder license. See Use System Objects in MATLAB Code Generation for more information.

Behavior change of locked System objects for loading, saving, and cloning

In the previous release, saving, loading, and cloning a locked System object would result in an unlocked System object. This System object had the same property values as the one from which it was cloned, but not the same internal state.

In this release, it does not matter whether you save a locked System object into a MAT file and load it later or clone a locked System object using the `clone` method. In either case, the result is a locked System object with the same property values and the same internal states.

There are, however, a few exceptions. For the following System objects, if you call the `clone` method, the resulting System object is not locked, but if you save or load the System object into and from a MAT file, the result is a locked System object.

- `dsp.MatFileWriter`
- `dsp.AudioRecorder`
- `dsp.AudioPlayer`

Thus, for the above System objects, if you call the `clone` method, you get an unlocked System object with the same property values.

Another exception involves the following System objects:

- `dsp.AudioFileWriter`
- `dsp.AudioFileReader`

For these System objects, if you save a locked System object to a MAT file and load it later, you get an unlocked System object with the same property values. However you do not get the same internal states. This behavior is the same as in the previous release.

Behavior change of statistics blocks for variable-size inputs

When the inputs are of variable size, the running mode behavior of the following blocks has changed:

- Mean
- RMS
- Variance
- Standard Deviation
- Minimum
- Maximum

If the **Input processing** parameter is set to `Elements as channels (sample based)`, the block state is reset when any input dimension changes.

If the **Input processing** parameter is set to `Columns as channels (frame based)`, then the behavior depends on two options:

- When the number of input channels (i.e., number of columns) changes, the block state is reset to its initial condition.
- When the number of input channels remains the same, there is no reset, even if the channel length (i.e., number of rows) changes.

Simulation state save and restore for additional blocks

In this release, there are additional blocks that support simulation state save and restore. These are:

- Cumulative Sum
- Cumulative Product
- Mean
- Variance
- RMS
- Standard Deviation
- Queue
- Stack

Note The Queue and Stack blocks do not support SimState save and restore in dynamic memory allocation mode.

For more information on simulation state save and restore, see Save and Restore Simulation State as SimState.

For Each subsystem support for additional blocks

In this release, most DSP blocks have been updated to support the For Each subsystem. For details about the For Each subsystem, see For Each. For a list of supported blocks, see the *For Each Subsystem Support* column in the table titled *Simulink Block Data Type Support for DSP System Toolbox*. This table can be accessed by typing `showsignalblockdatatypetable` at the command line.

Multi-instance model referencing support for additional blocks

In this release, most DSP blocks have been updated to support multi-instance normal mode model referencing. For details about model referencing, see Model Reference. The blocks that support model referencing are the same blocks that support the For Each subsystem. Therefore for a list of supported blocks, see the *For Each Subsystem Support* column in the table titled

Simulink Block Data Type Support for DSP System Toolbox. This table can be accessed by typing `showsignalblockdatatypetable` at the command line.

Expanded analysis support for filter System objects

In the previous release, the filter analysis methods for `dfilt` and `mfilt` objects were extended to filter System objects. This release expands the number of supporting analysis methods to include the following:

- `noisepsd`
- `noisepsdopts`
- `freqrespest`
- `freqrespopts`

For a comprehensive list of supported analysis methods, see Analysis Methods for Filter System Objects.

Removal of the signalblks package

In this release, the `signalblks` package is being removed and any instantiation of a `signalblks` object causes an error message. In future releases, the functionality will be removed entirely, so you should now use the corresponding object in the DSP System Toolbox.

Discrete filter block visible in DSP library

In addition to accessing the Discrete Filter block from the Simulink library, you can now also access it from the DSP System Toolbox library.

System object tunable parameter support in code generation

You can change tunable properties in user-defined System objects at any time, regardless of whether the object is locked. For System objects predefined in the software, the object must be locked. In previous releases, you could tune

System object properties only for a limited number of predefined System objects in generated code.

save and load methods for System objects

You can use the `save` method to save System objects to a MAT file. If the object is locked, its state information is saved, also. You can recall and use those saved objects with the `load` method.

You can also create your own `save` and `load` methods for a System object you create. To do so, use the `saveObjectImpl` and `loadObjectImpl`, respectively, in your class definition file.

Save and restore SimState not supported for System objects

Compatibility Considerations: Yes

The Save and Restore Simulation State as SimState option is no longer supported for any System object in a MATLAB Function block. This option was removed because it prevented parameter tunability for System objects, which is important in code generation.

Compatibility Considerations

If you need to save and restore simulation states, you may be able to use a corresponding Simulink block, instead of a System object.

Map integer delay to RAM on Delay block

`UseRAM` is a block-level parameter on the IntegerDelay block that you access with the HDL Block properties GUI. You assign it an `On` or `Off` value:

- `On`: Map the integer delay to a RAM. `On` is not a guarantee that a RAM is inferred: if all conditions are met (including the threshold criteria), only then is the RAM inferred.
- `Off`: The integer delay is always mapped to registers.

HDL support for System objects

HDL support for the following System objects has been added with release R2012b:

- `dsp.BiquadFilter`

HDL resource sharing for Biquad Filter block

HDL support for second-order section, direct-form I and second-order section, direct-form II filter structures has been added with release R2012b.

Supported architectures:

- Fully Parallel ('default' interface)

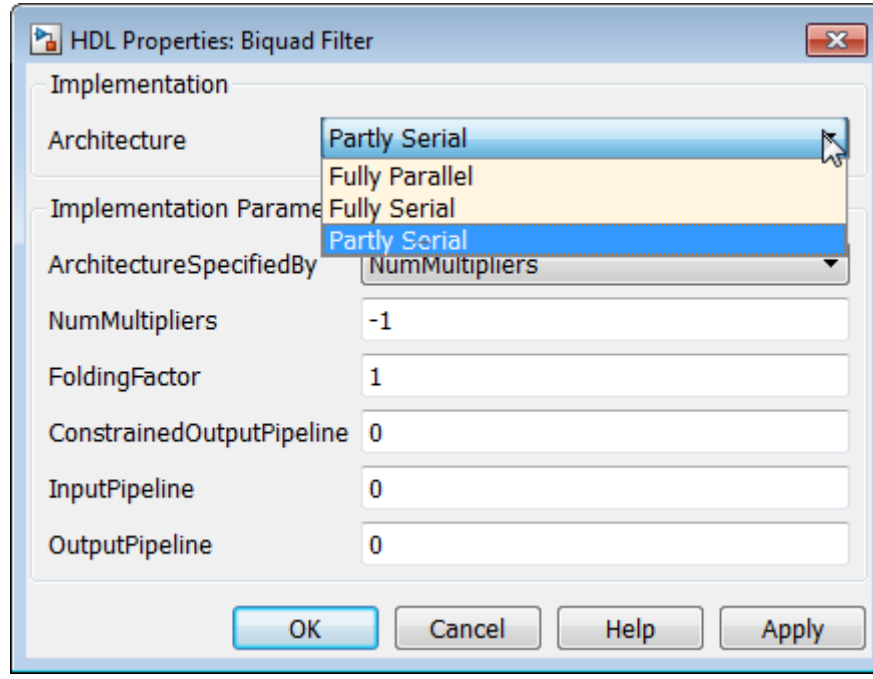
  AddPipelineRegisters, ConstrainedOutputPipeline, CoeffMultipliers, InputPipeline, OutputPipeline

- Fully Serial

  ConstrainedOutputPipeline, InputPipeline, OutputPipeline

- Partly Serial

  ConstrainedOutputPipeline, InputPipeline, OutputPipeline, ArchitectureSpecifiedBy, FoldingFactor, NumMultipliers


HDL Properties: Biquad Filter
Implementation
Architecture
Partly Serial
Fully Parallel
Fully Serial
Partly Serial
Implementation Parame
ArchitectureSpecifiedBy
NumMultipliers
NumMultipliers
-1
FoldingFactor
1
ConstrainedOutputPipeline
0
InputPipeline
0
OutputPipeline
0
OK
Cancel
Help
Apply

R2012a

Version: 8.2

New Features: Yes

Bug Fixes: Yes

Frame-Based Processing
Compatibility Considerations: Yes

Beginning in R2010b, MathWorks has been significantly changing the handling of frame-based processing. For more information, see "Frame-Based Processing" on page 64 in the R2011b Release Notes.

The following sections provide more detailed information about the specific R2012a DSP System Toolbox software changes that are helping to enable the transition to the new paradigm for frame-based processing:

- "Inherited Option of the **Input Processing** Parameter Now Warns" on page 48
- "Logging Frame-Based Signals in Simulink" on page 49
- "Model Reference and Using slupdate" on page 50
- "Removing Mixed Frameness Support for Bus Signals on Unit Delay and Delay" on page 51
- "Audio Output Sampling Mode Added to the From Multimedia File Block" on page 51

Inherited Option of the Input Processing Parameter Now Warns

Some DSP System Toolbox blocks are able to process both sample- and frame-based signals. After the transition to the new way of handling frame-based processing, signals will no longer carry information about their frame status. Blocks that can perform both sample- and frame-based processing will have a new parameter that allows you to specify the appropriate processing behavior.

To prepare for this change, many blocks received a new **Input processing** parameter in previous releases. You can set this parameter to `Columns as channels (frame based)` or `Elements as channels (sample based)`, depending upon the type of processing you want. The third choice, `Inherited (this choice will be removed - see release notes)`, is a temporary selection that is available to help you migrate your existing models from the old paradigm of frame-based processing to the new paradigm.

In this release your model will warn when the following conditions are all met for any block in your model:

- The **Input processing** parameter is set to `Inherited (this choice will be removed - see release notes).`
- The input signal is sample based.
- The input signal is a vector, matrix, or N-dimensional array.

Compatibility Considerations

To eliminate this warning, you must upgrade your existing models using the `slupdate` function. The function detects all blocks that have `Inherited (this choice will be removed - see release notes)` selected for the **Input processing** parameter. It then asks you whether you would like to upgrade each block. If you select yes, the function detects the status of the frame bit on the input port of the block. If the frame bit is `1` (frames), the function sets the **Input processing** parameter to `Columns as channels (frame based)`. If the bit is `0` (samples), the function sets the parameter to `Elements as channels (sample based)`.

In a future release, the frame bit and the `Inherited (this choice will be removed - see release notes)` option will be removed. At that time, the **Input processing** parameter in models that have not been upgraded will automatically be set to either `Columns as channels (frame based)` or `Elements as channels (sample based)`. The option set will depend on the library default setting for each block. If the library default setting does not match the parameter setting in your model, your model will produce unexpected results. Additionally, after the frame bit is removed, you will no longer be able to upgrade your models using the `slupdate` function. Therefore, you should upgrade your existing modes using `slupdate` as soon as possible.

Logging Frame-Based Signals in Simulink

In this release, a new warning message appears when a Simulink model is logging frame-based signals and the **Signal logging format** is set to `ModelDataLogs`. In `ModelDataLogs` mode, signals are logged differently depending on the status of the frame bit, as shown in the following table:

Status of Frame Bit	Today	When Frame Bit Is Removed
Sample-based	3-D array with samples in time in the third dimension	3-D array with samples in time in the third dimension
Frame-based	2-D array with frames in time concatenated in the first dimension	3-D array with samples in time in the third dimension

This warning advises you to switch your **Signal logging format** to `Dataset`. The `Dataset` logging mode logs all 2-D signals as 3-D arrays, so its behavior is not dependent on the status of the frame bit.

When you get the warning message, to continue logging signals as a 2-D array:

1. Select **Simulation > Model Configuration Parameters > Data Import/Export**, and change **Signal logging format** to `Dataset`. To do so for multiple models, click on the link provided in the warning message.
2. Simulate the model.
3. Use the dsp.util.getLogsArray function to extract the logged signal as a 2-D array.

Model Reference and Using slupdate

In this release, the Model block has been updated so that its operation does not depend on the frame status of its input signals.

Compatibility Considerations

In a future release, signals will not have a frameness attribute, therefore models that use the Model block must be updated to retain their behavior. If you are using a model with a Model block in it, follow the steps below to update your model:

1. For both the child and the parent models:

- In the **Model Configuration Parameters** dialog box, select the **Diagnostics > Compatibility** pane.
- Change the **Block behavior depends on input frame status** parameter to `warning`.

2 For both the child and the parent models, run `slupdate`.

3 For the child model only:

- In the **Model Configuration Parameters** dialog box, select the **Diagnostics > Compatibility** pane.
- Change the **Block behavior depends on input frame status** parameter to `error`.

Removing Mixed Frameness Support for Bus Signals on Unit Delay and Delay

This release phases out support for buses with mixed sample- and frame-based elements on Simulink's Unit Delay and Delay blocks. Support is also removed from the DSP System Toolbox's Delay block. When the frame bit is removed in a future release, any Delay block that has a bus input of mixed frameness will produce different results.

Compatibility Considerations

This incompatibility is phased over multiple releases. In 2012a, the blocks will produce a warning message. In a future release, when the frame bit is removed, the blocks will produce an error message.

Audio Output Sampling Mode Added to the From Multimedia File Block

The From Multimedia File block now provides a new parameter. This parameter allows you to select frame- or sample-based audio output processing.

System Object Enhancements

Compatibility Considerations: Yes

- "Code Generation for System Objects" on page 52
- "New MAT-File Reader and Writer System Objects" on page 52
- "New System Object Option on File Menu" on page 52
- "Variable-Size Input Support for System Objects" on page 52
- "Data Type Support for System Objects" on page 53
- "New Property Attribute to Define States" on page 53
- "New Methods to Validate Properties and Get States from System Objects" on page 53
- "matlab.system.System changed to matlab.System" on page 53

Code Generation for System Objects

System objects defined by users now support C code generation. To generate code, you must have the MATLAB Coder product.

New MAT-File Reader and Writer System Objects

R2012a adds two new System objects, `dsp.MatFileReader` and `dsp.MatFileWriter`. These System objects stream data into and out of MAT-files.

New System Object Option on File Menu

The File menu on the MATLAB desktop now includes a **New > System object** menu item. This option opens a System object class template, which you can use to define a System object class.

Variable-Size Input Support for System Objects

System objects that you define now support inputs that change size at run time.

Data Type Support for System Objects

System objects that you define now support all MATLAB data types as inputs and outputs.

New Property Attribute to Define States

R2012a adds the new `DiscreteState` attribute for properties in your System object class definition file. Discrete states are values calculated during one step of an object's algorithm that are needed during future steps.

New Methods to Validate Properties and Get States from System Objects

The following methods have been added:

- `validateProperties` – Checks that the System object is in a valid configuration. This applies only to objects that have a defined `validatePropertiesImpl` method.
- `getDiscreteState` – Returns a `struct` containing System object properties that have the `DiscreteState` attribute.

matlab.system.System changed to matlab.System

The base System object class name has changed from `matlab.system.System` to `matlab.System`.

Compatibility Considerations

The previous `matlab.system.System` class will remain valid for existing System objects. When you define new System objects, your class file should inherit from the `matlab.System` class.

Time Scope Enhancements

- "Time Domain Measurements in Time Scope" on page 54
- "Multiple Display Support in Time Scope" on page 54
- "Style Dialog Box in Time Scope" on page 55

- "Sampled Data as Stairs in Time Scope" on page 55
- "Complex Data Support in Time Scope" on page 56
- "Additional Time Scope Enhancements" on page 56

Time Domain Measurements in Time Scope

As of R2012a, the Time Scope block and System object support the time domain signal measurements **Signal statistics**, **Bilevel measurements**, and **Peak finder**. The **Signal Statistics** panel displays the maximum, minimum, peak-to-peak difference, mean, median, and RMS values of a selected signal. It also displays the times at which the maximum and minimum values occur. The **Bilevel Measurements** panel displays information about a selected signal's transitions, overshoots or undershoots, and cycles. The **Peak Finder** panel displays maxima and the times at which they occur. These displays allow you to modify the settings for peak threshold, maximum number of peaks, and peak excursion.

To use the new time domain measurements features in the Time Scope block, click one of the three corresponding buttons in the Time Scope toolbar. You can also access these panels by selecting **Measurements** from the **Tools** menu.

See the Time Scope reference topic for more information.

Multiple Display Support in Time Scope

R2012a allows you to choose to have multiple displays in the Time Scope, using both the block and the System object. This feature allows you to tile your screen into a number of separate displays, up to a grid of 4 rows and 4 columns. You may find multiple displays useful when the Time Scope takes multiple input signals.

To set the number of displays on the Time Scope, click the layout button in the Time Scope toolbar. You can also select **Layout** from the **View** menu. To set the number of displays using the `dsp.TimeScope` System object, set the `LayoutDimensions` property.

To change options for a display, select **Properties** from the **View** menu. Select the **Display** tab, and use the menu to select the display you want

to update. You can also right-click on the axes, and select **Properties**. To change options using the `dsp.TimeScope` System object, set the `ActiveDisplay` property to the intended display number.

See the Time Scope reference topic for more information.

Style Dialog Box in Time Scope

R2012a enhances the Time Scope block and System object by allowing you to customize the style of displays using a Style dialog box. You are able to change the color of the figure containing the displays, the background and foreground colors of display axes, and properties of lines in a display.

The Style dialog box replaces the **Line Properties** menu item that was used in previous releases for customizing line properties. To open the Style dialog box, select **Style** from the **View** menu.

Note This release changes the Time Scope default axes and line colors. The Time Scope initially displays the axes as black instead of white, as shown in previous releases. For a real, single-channel signal, Time Scope now displays a yellow line instead of a blue line. Models containing Time Scope blocks that were created using older versions of DSP System Toolbox will not be affected by this change.

See the Time Scope reference topic for more information.

Sampled Data as Stairs in Time Scope

In previous releases, the Time Scope plotted a sampled signal as lines connecting each of the sampled values. This approach is similar to the functionality of the MATLAB `line` or `plot` function. In R2012a, the Time Scope block and System object can also plot a sampled signal as horizontal lines. These lines represent a sample value for a discrete sample period connected by vertical lines to represent a change in values occurring at each new sample. This type of plot is commonly called a *Stairstep* graph and has functionality similar to that of the MATLAB `stairs` function. Stairstep graphs are useful for drawing time history graphs of digitally sampled data.

To display a sampled signal as a Stairstep graph, first select **Style** from the **View** menu. The Style dialog box opens, allowing you to set the **Plot type** drop down box to `Stairs`. The three options available are `Line`, `Stairs`, and `Auto`. If using the `dsp.TimeScope` System object, set the `PlotType` property to the string `'stairs'`.

See the Time Scope reference topic for more information.

Complex Data Support in Time Scope

Beginning in this release, the Time Scope block and System object will support complex data input. The complex data is displayed by default in real and imaginary form as differently colored lines on the same axes. Alternately, you can display the magnitude and phase of the signal on separate axes in the same display.

To change the complex data options in the Time Scope display, select **Properties** from the **View** menu. Then, select or deselect the **Plot signals as magnitude and phase** check box. You can also right-click on the axes and select **Properties**. To change these properties using the `dsp.TimeScope` System object, set the `MagnitudePhase` property to either `true` or `false`.

See the Time Scope reference topic for more information.

Additional Time Scope Enhancements

In additional to the modifications mentioned above, R2012a also includes the following enhancements to the Time Scope:

Ability to Change the Time Units of the Display

In previous releases Time Scope always displayed time in metric units. In R2012a, the Time Scope block and System object allow you to label the X-axis in two additional ways. First, you can ensure that the X-axis is always labeled as `Time (seconds)` and that the appropriate power of 10 appears in the bottom-right corner of the Time Scope display. Second, you can remove the units in the X-axis label entirely.

To change the manner in which the time units are displayed, select **Properties** from the **View** menu. Then, set the **Time Units** parameter

to either `Seconds` or `None`. The default option is `Metric (based on Time Span)`. To change these properties using the `dsp.TimeScope` System object, set the `TimeUnits` property to either `Seconds` or `None`. The default property value is `Metric`.

See the Time Scope reference topic for more information.

Simulink Enumerations Supported in Time Scope Block

In previous releases, the Time Scope block supported input signals of floating-point and fixed-point data types. R2012a adds support for Simulink enumerated data types.

Note This feature is available only for the Time Scope block but not for the Time Scope System object. Also, support is provided only for Simulink enumerations, but not for generic MATLAB enumeration classes.

See the Time Scope reference topic for more information.

ASIO Support in To/From Audio Device Blocks and Objects

The To and From Audio Device blocks and the dsp.AudioPlayer and dsp.AudioRecorder system objects all now support ASIO™ as an API. ASIO is used to communicate with the audio hardware. To set ASIO as the Audio Hardware API, select **Preferences** from the MATLAB Toolstrip. Then select DSP System Toolbox from the tree menu. If the ASIO selection is disabled, it is due to the ASIO device not being connected.

Video Processing Enabled for the DSP System Toolbox Multimedia File Blocks

The To Multimedia File and From Multimedia File blocks no longer require a Computer Vision System Toolbox™ license for video processing. You can use the DSP From Multimedia File block to read video and the To Multimedia File block to write video files.

System Objects Integrated into Filter Design Workflow

- "Integration of System Objects into Filter Design via fdesign, FDATool, and Filterbuilder" on page 58
- "Convert dfilt and mfilt Filter Objects to System Objects" on page 58
- "Filter Analysis and Conversion Methods for System Object Filters" on page 58

Integration of System Objects into Filter Design via fdesign, FDATool, and Filterbuilder

In R2012a, you can use the `fdesign` workflow and interactive tools, `fdatool` and `filterbuilder`, to create IIR, FIR, and multirate System object filters.

Using the interactive tools, you can generate MATLAB code to construct a System object filter. You can also generate MATLAB code to filter data with your System object that is compatible with C/C++ code generation. C/C++ code generation requires the MATLAB Coder software.

Convert dfilt and mfilt Filter Objects to System Objects

In R2012a, you can convert `dfilt` and `mfilt` objects to System objects. Use the `sysobj` method to convert an existing `dfilt` or `mfilt` object to a System object. Refer to the command-line help for `dfilt.sysobj` and `mfilt.sysobj` for details on supported single-rate and multirate filter structures.

Filter Analysis and Conversion Methods for System Object Filters

The R2012a release extends filter analysis and conversion methods for `dfilt` and `mfilt` objects to System object filters. For System object filters, you can examine the filter magnitude, impulse, step, zero phase, group delay, and phase responses. You can also view a pole-zero plot of your filter's *z*-transform.

Additionally, you can obtain detailed measurements and implementation costs for your System object filter. For System object filters, you can determine if the phase response is linear. For FIR linear-phase filters, you can determine the type of linear phase. You can also assess the stability of

your filter design and whether your design represents a minimum-phase or maximum-phase system.

To obtain a comprehensive list of supported methods and links to the command-line help, enter

```
dsp.SystemObjectFilter.helpFilterAnalysis
```

at the command line, where *`SystemObjectFilter`* is a specific System object filter class name. For example:

```
dsp.BiquadFilter.helpFilterAnalysis
```

New Measurement Workflow

- "Measurements for Bilevel Pulse Waveforms" on page 59
- "System Objects for Peak-to-RMS and Peak-to-Peak Measurements" on page 60

Measurements for Bilevel Pulse Waveforms

The R2012a release introduces System objects that perform a number of basic measurements on bilevel pulse waveforms. These measurements include:

- State-level estimation for bilevel pulse waveforms using the histogram method. See the help for `dsp.StateLevels` for details.
- Transition metrics for bilevel pulse waveforms. The System object, `dsp.TransitionMetrics`, determines low-, middle-, and high-reference level crossings and also duration and slew rate. You can also use `dsp.TransitionMetrics` to measure the behavior of bilevel waveforms in pretransition and posttransition regions such as overshoot, undershoot, and settling time.

  Using `dsp.PulseMetrics`, you can measure transition rise and fall times. `dsp.PulseMetrics` contains a superset of the capabilities found in `dsp.TransitionMetrics`.
- Cycle metrics for bilevel pulse waveforms. You can use `dsp.PulseMetrics` to measure pulse width, pulse separation, pulse period, and duty cycle.

System Objects for Peak-to-RMS and Peak-to-Peak Measurements

The R2012a release introduces System objects to measure the root-mean-square (RMS) value of a waveform. These System objects also measure the peak-to-RMS and peak-to-peak values. For details, see the reference pages for `dsp.RMS`, `dsp.PeakToRMS`, and `dsp.PeakToPeak`.

Discrete FIR Filter System Object

This release introduces a discrete FIR Filter System object, which filters each channel of the input using static or time-varying FIR filter implementations. This System object implements the algorithm, inputs, and outputs described on the Discrete FIR Filter block reference page. The object properties correspond to the block parameters. Both this object and its corresponding block let you specify whether to process inputs as individual samples or as frames of data. The object uses the `FrameBasedProcessing` property. The block uses the **Input processing** parameter. This System object supports code generation.

Inverse Dirichlet Sinc-Shaped Passband Design Added to Constrained FIR Equiripple Filter

The R2012a release adds the ability to design a constrained FIR equiripple filter with an inverse-Dirichlet-sinc-shaped passband using `firceqrip`. An inverse-Dirichlet-sinc-shaped response is often used to compensate for the Dirichlet-sinc-shaped response of a cascade integrator comb (CIC) filter. An inverse-sinc-shaped response is a valid approximation to the response of a CIC filter only when the sampling rate change is sufficiently high. The Dirichlet sinc provides a more exact match to the response of a CIC filter.

Code Generation Support Added to FIR Decimator System Object

In this release, FIR Decimator has been added to the list of objects that can now support code generation in MATLAB via the codegen function.

Filter Block Enhancements

- "IC/Coefficient Parameter Ports in the Simulink Discrete Filter and Discrete Transfer Function" on page 61
- "Reset Port for Resetting Filter State in Filter Blocks" on page 61

IC/Coefficient Parameter Ports in the Simulink Discrete Filter and Discrete Transfer Function

The Simulink Discrete Filter and Discrete Transfer Function blocks now have the capability of specifying the numerator and denominator coefficients via either input parameter ports or block dialog boxes. Also, the initial filter states can be specified via an input parameter port or a block dialog box.

Reset Port for Resetting Filter State in Filter Blocks

The Simulink Discrete Filter and Discrete Transfer Function blocks now allow the filter states to be reset via a reset parameter port called External reset.

Discrete FIR Filter Block Coefficient Port Changes

Compatibility Considerations: Yes

In this release, if you feed a column vector input into the coefficient port of the Discrete FIR Filter block, the block issues a command-line warning. This warning will state that column vector inputs to the coefficient port are not supported and that you will see an error message in future releases.

Compatibility Considerations

You are advised to run `slupdate` to insert a reshape block and transpose the input from a column vector to a row vector.

Statistics Blocks and Objects Warning for Region of Interest Processing

Compatibility Considerations: Yes

ROI processing will be removed in a future release. Currently, ROI processing is available only if you have a Computer Vision System Toolbox license. If you do not have a license for that product, you can still use ROI processing, but you are limited to the use of ROI type rectangles.

Compatibility Considerations

When you use region of interest (ROI) processing, MATLAB will issue a warning. ROI processing will be removed in a future release.

New and Updated Demos

R2012a adds the following new demo:

Using System Objects with MATLAB Coder — Shows you how to use the MATLAB Coder product to generate code for a MATLAB file that uses System objects.

Additionally, this release updates the Arbitrary Magnitude Filter Design demo to include examples that showcase the new capabilities of the arbitrary magnitude filter design.

R2011b

Version: 8.1

New Features: Yes

Bug Fixes: Yes

Frame-Based Processing

Compatibility Considerations: Yes

In signal processing applications, you often need to process sequential samples of data at once as a group, rather than one sample at a time. DSP System Toolbox documentation refers to the former as frame-based processing and the latter as sample-based processing. A frame is a collection of samples of data, sequential in time.

Historically, Simulink-family products that can perform frame-based processing propagate frame-based signals throughout a model. The frame status is an attribute of the signals in a model, just as data type and dimensions are attributes of a signal. The Simulink engine propagates the frame attribute of a signal by means of a frame bit, which can either be on or off. When the frame bit is on, Simulink interprets the signal as frame based and displays it as a double line, rather than the single line sample-based signal.

General Product-Wide Changes

Beginning in R2010b, MathWorks® started to significantly change the handling of frame-based processing. In the future, frame status will no longer be a signal attribute. Instead, individual blocks will control whether they treat inputs as frames of data or as samples of data. To learn how a particular block handles its input, you can refer to the block reference page.

To make the transition to the new paradigm of frame-based processing, many blocks have received new parameters. You can view an example of how to use these parameters to control sample- and frame-based processing in R2011b and future releases. To open the model, type ex_inputprocessing at the MATLAB command line. This model demonstrates how a block can process a signal as sample based or frame based, depending on the setting of that block's **Input processing** parameter.

Notice that when the Discrete FIR Filter and Time Scope blocks are configured to perform frame-based processing, they interpret columns as channels and treat the 2-by-2 input signal as two independent channels. Conversely, when the blocks are configured to perform sample-based processing, they interpret elements as channels and treat the 2-by-2 input signal as four independent

channels. For further information about sample- and frame-based processing, see Sample- and Frame-Based Concepts.

The following sections provide more detailed information about the specific R2011b DSP System Toolbox software changes that are helping to enable the transition to the new way of frame-based processing:

- "Logging Signals in Simulink" on page 66
- "Triggered to Workspace" on page 66
- "Digital Filter Design Block" on page 67
- "Filterbuilder, FDATool and the Filter Realization Wizard Block" on page 68
- "Changes to Row Vector Processing for dsp.Convolver, dsp.CrossCorrelator, and dsp.Interpolator System Objects" on page 69

Compatibility Considerations

During this transition to the new way of handling frame-based processing, both the old way (frame status as an attribute of a signal) and the new way (each block controls whether to treat inputs as samples or as frames) will coexist for a few releases. For now, the frame bit will still flow throughout a model, and you will still see double signal lines in your existing models that perform frame-based processing.

- **Backward Compatibility** — By default, when you load an existing model in R2011b any new parameters related to the frame-based processing change will be set to their backward-compatible option. For example, if any blocks in your existing models received a new **Input processing** parameter this release, that parameter will be set to `Inherited (this choice will be removed - see release notes)` when you load your model in R2011b. This setting enables your existing models to continue working as expected until you upgrade them. Because the inherited option will be removed in a future release, you should upgrade your existing models as soon as possible.
- **slupdate Function** — To upgrade your existing models to the new way of handling frame-based processing, you can use the `slupdate` function. Your model must be compilable in order to run the `slupdate` function. The function detects all blocks in your model that are in need of updating, and

asks you whether you would like to upgrade each block. If you select yes, the `slupdate` function updates your blocks accordingly.

- **Timely Update to Avoid Unexpected Results** — It is important to update your existing models as soon as possible because the frame bit will be removed in a future release. At that time, any blocks that have not yet been upgraded to work with the new paradigm of frame-based processing will automatically transition to perform their library default behavior. The library default behavior of the block might not produce the results you expected, thus causing undesired results in your models. Once the frame bit is removed, you will no longer be able to upgrade your models using the `slupdate` function. Therefore, you should upgrade your existing modes using `slupdate` as soon as possible.

Logging Signals in Simulink

R2011b adds new capabilities to the DSP System Toolbox product for logging signals in Simulink. When you log signals using the `Dataset` logging mode, you can now use DSP System Toolbox utility functions to help you access that logged data in either a 2-D or 3-D format. For more information about selecting a signal logging format, see Specifying the Signal Logging Data Format in the Simulink documentation.

After you log a signal using the `Dataset` logging mode, you can choose to extract that logged signal in either a 2-D or 3-D format. To fully support this new workflow, the following utility functions and class have been added to the DSP System Toolbox product:

- `dsp.util.getLogsArray` — Formats and returns a 2-D or 3-D MATLAB array from a logged signal in a `Dataset` object.
- `dsp.util.getSignalPath` — Returns all paths to signals with a specified name in the `Dataset` object.
- `dsp.util.SignalPath` — Contains path information for signals in `Simulink.SimulationData.Dataset` objects.

Triggered to Workspace

R2011b adds a new **Save 2-D signals as** parameter to the Triggered to Workspace block. This parameter allows you to specify whether the block saves 2-D signals as 2-D arrays or as 3-D arrays. To provide for backward

compatibility, the **Save 2-D signals as** parameter also has an option `Inherit from input (this choice will be removed  see release notes)`. When you select this option, the block saves sample-based data as a 3-D array and frame-based data as a 2-D array.

Compatibility Considerations

In a future release, the following option will be removed: `Inherit from input (this choice will be removed  see release notes)`. From this time forward, you must specify whether the block saves 2-D signals as 2-D or 3-D arrays. The block will no longer make that choice based on the status of the frame bit.

You can use the `slupdate` function to upgrade your existing models that contain a Triggered To Workspace block. The function detects whether your models contain any Triggered To Workspace blocks with the **Save 2-D signals as** parameter set to `Inherit from input (this choice will be removed  see release notes)`. If you do, the function detects the status of the frame bit and sets the **Save 2-D signals as** parameter accordingly.

- If the input signal is frame based, the function sets the **Save 2-D signals as** parameter to `2-D array (concatenate along first dimension)`.
- If the input signal is sample based, the function sets the **Save 2-D signals as** parameter to `3-D array (concatenate along third dimension)`.

Digital Filter Design Block

R2011b adds a new **Input processing** parameter to the Digital Filter Design block. This parameter allows you to choose whether you want the block to perform sample- or frame-based processing on the input. You can set this parameter to either `Elements as channels (sample based)` or `Columns as channels (frame based)`. The third choice, `Inherited (this choice will be removed - see release notes)`, is a temporary selection. This additional option will help you as you move control of frame-based processing from the signals to the blocks themselves.

Compatibility Considerations

When you load an existing model R2011b, all Digital Filter Design blocks in your model will have the new **Input processing** parameter. By default, it will be set to `Inherited (this choice will be removed - see release notes)`. This setting enables your existing models to continue to work as expected until you upgrade them. Although your old models will still work when you open and run them in R2011b, you should upgrade them as soon as possible.

You can upgrade your existing models using the `slupdate` function. The function detects all blocks that have `Inherited (this choice will be removed - see release notes)` selected for the **Input processing** parameter. It then asks you whether you would like to upgrade each block. If you select yes, the function detects the status of the frame bit on the input port of the block. If the frame bit is `1` (frames), the function sets the **Input processing** parameter to `Columns as channels (frame based)`. If the bit is `0` (samples), the function sets the parameter to `Elements as channels (sample based)`.

In a future release, the frame bit and the `Inherited (this choice will be removed - see release notes)` option will be removed. At that time, the **Input processing** parameter on blocks in models that have not been upgraded will automatically be set to the block's library default setting. In the case of the Digital Filter Design block, the library default setting is `Columns as channels (frame based)`. If the library default setting does not match the parameter setting in your model, your model will produce unexpected results.

Additionally, after the frame bit is removed, you will no longer be able to upgrade your models using the `slupdate` function. Therefore, you should upgrade your existing modes using `slupdate` as soon as possible.

Filterbuilder, FDATool and the Filter Realization Wizard Block

R2011b adds new **Input processing** and **Rate options** parameters to `filterbuilder`, `FDATool` and the Filter Realization Wizard block. For `filterbuilder`, these new parameters are available when you click the **Generate Model** button on the Code Generation pane of the dialog box. For `FDATool` and the Filter Realization Wizard block, the new parameters are

available on the Realize Model pane of the dialog box. When you use the **Realize Model** button to create a block, you can use the **Input processing** parameter to specify whether the block will perform sample- or frame-based processing on its input.

If you are creating a multirate filter block, the **Rate options** parameter will also be available. This parameter allows you to specify whether the filter block you create will `Enforce single-rate processing` or `Allow multirate processing`.

Changes to Row Vector Processing for dsp.Convolver, dsp.CrossCorrelator, and dsp.Interpolator System Objects

In previous releases, the `dsp.Convolver`, `dsp.CrossCorrelator`, and `dsp.Interpolator` System objects processed row vector inputs as a column vector. As of R2011b, these objects now process row vector inputs as a row vector (multiple channels).

Compatibility Considerations

Starting in R2011b, you must update your code to transpose the row vector data to a column vector before providing it as an input to the `dsp.Convolver`, `dsp.CrossCorrelator`, or `dsp.Interpolator` System objects.

Custom System Objects

You can now create custom System objects in MATLAB. This capability allows you to define your own System objects for time-based and data-driven algorithms, I/O, and visualizations. The System object API provides a set of implementation and service methods that you incorporate into your code to implement your algorithm. See Define New System Objects in the DSP System Toolbox documentation for more information.

New Allpole Filter Block

R2011b adds a new Allpole Filter block to the Filtering/Filter Implementations library. This block provides direct form, direct form transposed, and Lattice AR allpole filter structures.

New Audio Weighting Filter Functionality

R2011b adds new audio weighting filter functionality to MATLAB and Simulink. In MATLAB, you can now design audio weighting filters in the `filterbuilder` GUI or by using the preexisting `fdesign.audioweighting` object. In Simulink, you can use the new Audio Weighting Filter block from the Filtering/Filter Designs library.

Time Scope Enhancements

R2011b includes the following enhancements to the Time Scope:

- **Improvements to default signal names in the scope legend** — In previous releases, the default names for signals displayed by the Time Scope block were `Channel 1`, `Channel 2`, `Channel 3`, etc. In R2011b, the default naming convention has been improved to also identify the source of the signal. For example, if the input to the Time Scope block is two separate two channel signals named `SignalA` and `SignalB`, the default legend names would appear as:

  `SignalA:1`, `SignalA:2`, `SignalB:1`, `SignalB:2`

  See the Time Scope reference topic for more information.

- **New scrolling display mode simplifies debugging process** — R2011b adds a new option to the Time Scope block and System object. This option allows you to specify how the scope displays new data beyond the visible time span. In previous releases, the scope always displayed new data up until it reached the maximum X-axis limit. When the data reached the maximum X-axis limit of the scope window, the scope cleared the display and updated the time offset value. It then displayed subsequent data points starting from the minimum X-axis limit. In the new scrolling display mode, the scope scrolls old data to the left to make room for new data on the right side of the scope display. This mode is graphically intensive and can affect run-time performance, but it is beneficial for debugging and for monitoring time-varying signals.

  To use the new scrolling display mode in the Time Scope block, set the **Time span overrun mode** parameter to `Scroll` on the **Visuals:TimeDomainOptions** dialog box. To use the new scrolling display mode in the `dsp.TimeScope` System object, set the

`TimeSpanOverrunMode` property to `Scroll`. By default, both the block and the System object display data using the previously supported `Wrap` mode.

New Arbitrary Group Delay Design Support

This release adds a new `fdesign.arbgrpdelay` filter specification object. *Arbitrary group delay filters* are allpass filters useful for correcting phase distortion introduced by other filters. Systems with nonlinear phase responses result in nonconstant group delay, which causes dispersion of the frequency components of the signal. This type of phase distortion can be undesirable even if the magnitude distortion introduced by the filter produces the desired effect. In these cases, you can compensate for the phase distortion by cascading the frequency-selective filter with an allpass filter that compensates for the group delay.

Arbitrary Magnitude Responses Now Support Minimum Order and Minimum/Maximum Phase Equiripple Design Options

R2011b adds new minimum order, minimum phase equiripple, and maximum phase equiripple design options. These design options are now available on the Arbitrary Response design panel in `filterbuilder` and through the `fdesign.arbmag` filter specification object.

Support for Constrained Band Equiripple Designs in MATLAB and Simulink

R2011b adds support for constrained band equiripple designs to the following filter response types:

fdesign Filter Specification Object	filterbuilder Response String	dspfdesign Library Block
`fdesign.arbmag`	`arbmag`	Arbitrary Response Filter
`fdesign.bandpass`	`bandpass`	Bandpass Filter

fdesign Filter Specification Object	filterbuilder Response String	dspfdesign Library Block
`fdesign.bandstop`	`bandstop`	Bandstop Filter
`fdesign.differentiator`	`diff`	Differentiator Filter

New Sinc Frequency Factor and Sinc Power Design Options for Inverse Sinc Filters

R2011b adds two new design options for designing inverse sinc filters in MATLAB and Simulink. The new design options allow you to control the sinc frequency factor and sinc power for inverse sinc filters designed with `fdesign.isinclp`, the new `fdesign.isinchp`, the `isinc` `filterbuilder` response type, or the Inverse Sinc Filter block in the `dspfdesign` library.

These new design options allow you to design inverse sinc lowpass filters with a passband magnitude response equal to H(ω) = sinc(Cω)^(–P). *`C`* is the sinc frequency factor, and *`P`* is the sinc power. Similarly, you can design inverse sinc highpass filters with a passband magnitude response equal to H(ω) = sinc(C(1-ω))^(–P). For both the highpass and lowpass filters, the default values of *`C`* and *`P`* are set to 0.5 and 1, respectively. For more information about the sinc frequency factor and sinc power design options, see the corresponding reference topics.

New Inverse Sinc Highpass Filter Designs

This release adds support for designing highpass inverse sinc filters in MATLAB and Simulink. This capability is available through a new `fdesign.isinchp` filter specification object as well as a new `isinchp` `filterbuilder` response type.

Filterbuilder and dspfdesign Library Blocks Now Support Different Numerator and Denominator Orders for IIR Filters

As of R2011b, you can now specify different numerator and denominator orders for IIR filters designed using certain filter responses. This capability is available for the bandpass, bandstop, highpass, and lowpass filter responses in the `filterbuilder` GUI and the corresponding `dspfdesign` library blocks.

New Stopband Shape and Stopband Decay Design Options for Equiripple Highpass Filter Designs

This release adds new stopband shape and stopband decay design options in MATLAB and Simulink. These options are available through the `fdesign.highpass` filter specification object, the `highpass` `filterbuilder` response type, and the Highpass Filter block in the `dspfdesign` library.

FFTW Library Support for Non-Power-of-Two Transform Length

The FFT, IFFT blocks, and the `dsp.IFFT`, `dsp.FFT` System objects include the use of the FFTW library. The blocks and objects now support non-power-of-two transform lengths.

MATLAB Compiler Support for dsp.DigitalDownConverter and dsp.DigitalUpConverter

R2011b adds MATLAB Compiler support for the `dsp.DigitalDownConverter` and `dsp.DigitalUpConverter` System objects. With this capability, you can use the MATLAB Compiler to take MATLAB files, which can include System objects, as input and generate standalone applications.

Complex Input Support for dsp.DigitalDownConverter

The `dsp.DigitalDownConverter` System object now supports complex inputs.

getFilters Method of dsp.DigitalDownConverter and dsp.DigitalUpConverter Now Return Actual Fixed-Point Settings

You can now access the actual fixed-point settings of the filter being used by the `dsp.DigitalDownConverter` and `dsp.DigitalUpConverter` System objects. To do so, you must first provide a fixed-point input to the object using the step method. Then, after the object is locked, call the `getFilters` method to access the actual fixed-point properties of the filter being implemented by the System object. Calling `getFilters` on an unlocked `dsp.DigitalDownConverter` or `dsp.DigitalUpConverter` System object returns the same results as previous releases.

dsp.SineWave and dsp.BiquadFilter Properties Not Tunable

Compatibility Considerations: Yes

The following `dsp.SineWave` properties are now nontunable:

- `Frequency`
- `PhaseOffset`

The following `dsp.BiquadFilter` properties are now nontunable:

- `SOSMatrix`
- `ScaleValues`

When objects are locked (i.e., after calling the `step` method), you cannot change any nontunable property values.

Compatibility Considerations

Review any code that changes any `dsp.SineWave` or `dsp.BiquadFilter` property value after calling the `step` method. You should update the code to use property values that do not change.

System Object DataType and CustomDataType Properties Changes

Compatibility Considerations: Yes

When you set a System object, fixed-point `<xxx>DataType` property to `` `Custom' ``, it activates a dependent `Custom<xxx>DataType` property. If you set that dependent `Custom<xxx>DataType` property before setting its `<xxx>DataType` property, a warning message displays. `<xxx>` differs for each object.

Compatibility Considerations

Previously, setting the dependent `Custom<xxx>DataType` property would automatically change its `<xxx>DataType` property to `` `Custom' ``. If you have code that sets the dependent property first, avoid warnings by updating your code. Set the `<xxx>DataType` property to `` `Custom' `` before setting its `Custom<xxx>DataType` property.

Note If you have a `Custom<xxx>DataType` in your code, but do not explicitly update your code to change `<xxx>DataType` to `` `Custom' ``, you may see different numerical output.

System Objects Variable-Size Input Dimensions

System objects that process variable-size input now also accept inputs where the number of input dimensions change.

Conversion of Error and Warning Message Identifiers

Compatibility Considerations: Yes

R2011b changes some error and warning message identifiers in DSP System Toolbox software. For System objects, both error and warning message identifiers have changed. On the Simulink side, the Time Scope block has one warning message with a new identifier.

Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages. You can also use them in code that uses a `try/catch` statement and performs an action based on a specific error identifier.

For example, for System objects, the `MATLAB:system:System:inputSpecsChangedWarning` identifier has changed to `MATLAB:system:inputSpecsChangedWarning`. If your code checks for `MATLAB:system:System:inputSpecsChangedWarning`, you must update it to check for `MATLAB:system:inputSpecsChangedWarning` instead.

For the Time Scope block, the `Simulink:Engine:Simulink:Engine:UnableToUpdateDisplayInRapidAccelMode` identifier has changed to `Simulink:Engine:UINotUpdatedDuringRapidAccelSim`. If your code checks for `Simulink:Engine:Simulink:Engine:UnableToUpdateDisplayInRapidAccelMode`, you must update it to check for `Simulink:Engine:UINotUpdatedDuringRapidAccelSim` instead.

To determine the identifier for a warning, run the following command just after you see the warning:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable *MSGID*.

To determine the identifier for an error that appears at the MATLAB prompt, run the following command just after you see the error:

```
exception = MException.last;
MSGID = exception.identifier;
```

Warning messages indicate a potential issue with your code. While you can turn off a warning, a suggested alternative is to change your code so it runs without warnings.

New and Updated Demos

R2011b adds the following new demos:

- 3-Band Parametric Audio Equalizer Using UDP Packets and Code Generation — Provides a three-band parametric equalizer algorithm based in MATLAB. The demo allows you to dynamically adjust the coefficients of the filters and shows you how to use the MATLAB Coder product to build a standalone executable file that you can run outside of MATLAB.
- Creating New Kinds of System Objects for File Input and Output — Provides an example of creating custom System objects in MATLAB for file input and output.

Additionally, this release updates the IIR Filter Design Given a Prescribed Group Delay demo to use the new `fdesign.arbgrpdelay` object.

Blocks Being Removed in a Future Release

Compatibility Considerations: Yes

The following blocks will be removed from the DSP System Toolbox product in a future release.

Block Being Removed (library)	Replacement Block
Digital FIR Filter Design (`dspddes3`)	Discrete FIR Filter
Remez FIR Filter Design (`dspddes3`)	Discrete FIR Filter
Least Squares FIR Filter Design (`dspddes3`)	Discrete FIR Filter
Digital FIR Raised Cosine Filter Design (`dspddes3`)	Discrete FIR Filter

Block Being Removed (library)	Replacement Block
Digital IIR Filter Design (dspddes3)	Discrete Filter
Yule-Walker IIR Filter Design (dspddes3)	Discrete Filter
Integer Delay (dspobslib)	Delay
Dyadic Analysis Filter Bank (dspobslib)	Dyadic Analysis Filter Bank (dspmlti4)
Dyadic Synthesis Filter Bank (dspobslib)	Dyadic Synthesis Filter Bank (dspmlti4)
Wavelet Analysis (dspobslib)	DWT
Wavelet Synthesis (dspobslib)	IDWT
Direct-Form II Transpose Filter (dsparch3)	Digital Filter
Time-Varying Direct-Form II Transpose Filter (dsparch3)	Digital Filter, Discrete FIR Filter, or Allpole Filter
Time-Varying Lattice Filter (dsparch3)	Digital Filter, Discrete FIR Filter, or Allpole Filter

Compatibility Considerations

Beginning in R2011b, Simulink will generate a warning when you load a model that contains one or more of the blocks listed in the preceding table. To ensure that your models continue to work as expected when these blocks are removed from the product in a future release, it is strongly recommended that you replace these unsupported blocks as soon as possible. You can automatically update the blocks in your model by using the `slupdate` function.

R2011a

Version: 8.0

New Features: Yes

Bug Fixes: Yes

Product Restructuring

The DSP System Toolbox product replaces the Signal Processing Blockset™ and Filter Design Toolbox™ products in R2011a.

You can access archived documentation for the Signal Processing Blockset and Filter Design Toolbox products on the MathWorks Web site.

Frame-Based Processing

Compatibility Considerations: Yes

In signal processing applications, you often need to process sequential samples of data at once as a group, rather than one sample at a time. DSP System Toolbox documentation refers to the former as frame-based processing and the latter as sample-based processing. A frame is a collection of samples of data, sequential in time.

Historically, Simulink-family products that can perform frame-based processing propagate frame-based signals throughout a model. The frame status is an attribute of the signals in a model, just as data type and dimensions are attributes of a signal. The Simulink engine propagates the frame attribute of a signal by means of a frame bit, which can either be on or off. When the frame bit is on, Simulink interprets the signal as frame based and displays it as a double line, rather than the single line sample-based signal.

General Product-Wide Changes

Beginning in R2010b, MathWorks started to significantly change the handling of frame-based processing. In the future, frame status will no longer be a signal attribute. Instead, individual blocks will control whether they treat inputs as frames of data or as samples of data. To learn how a particular block handles its input, you can refer to the block reference page.

To make the transition to the new paradigm of frame-based processing, many blocks have received new parameters. You can view an example of how to use these parameters to control sample- and frame-based processing in R2011a and future releases. To open the model, type ex_inputprocessing at the MATLAB command line. This model demonstrates how a block can

process a signal as sample based or frame based, depending on the setting of that block's **Input processing** parameter.

Notice that when the Discrete FIR Filter and Time Scope blocks are configured to perform frame-based processing, they interpret columns as channels and treat the 2-by-2 input signal as two independent channels. Conversely, when the blocks are configured to perform sample-based processing, they interpret elements as channels and treat the 2-by-2 input signal as four independent channels. For further information about sample- and frame-based processing, see Sample- and Frame-Based Concepts.

The following sections provide more detailed information about the specific R2011a DSP System Toolbox software changes that are helping to enable the transition to the new way of frame-based processing:

- "Blocks with a New Input Processing Parameter" on page 82
- "Changes to the Overlap-Add FFT Filter, Overlap-Save FFT Filter, and Short-Time FFT Blocks" on page 84
- "Difference Block Changes" on page 86
- "Signal To Workspace Block Changes" on page 86
- "Spectrum Scope Block Changes" on page 87
- "Sample-Based Row Vector Processing Changes" on page 87

Compatibility Considerations

During this transition to the new way of handling frame-based processing, both the old way (frame status as an attribute of a signal) and the new way (each block controls whether to treat inputs as samples or as frames) will coexist for a few releases. For now, the frame bit will still flow throughout a model, and you will still see double signal lines in your existing models that perform frame-based processing.

- **Backward Compatibility** — By default, when you load an existing model in R2011a any new parameters related to the frame-based processing change will be set to their backward-compatible option. For example, if any blocks in your existing models received a new **Input processing** parameter this release, that parameter will be set to `Inherited (this choice will`

`be removed - see release notes`) when you load your model in R2011a. This setting enables your existing models to continue working as expected until you upgrade them. Because the inherited option will be removed in a future release, you should upgrade your existing models as soon as possible.

- **slupdate Function** — To upgrade your existing models to the new way of handling frame-based processing, you can use the `slupdate` function. Your model must be compilable in order to run the `slupdate` function. The function detects all blocks in your model that are in need of updating, and asks you whether you would like to upgrade each block. If you select yes, the `slupdate` function updates your blocks accordingly.
- **Timely Update to Avoid Unexpected Results** — It is important to update your existing models as soon as possible because the frame bit will be removed in a future release. At that time, any blocks that have not yet been upgraded to work with the new paradigm of frame-based processing will automatically transition to perform their library default behavior. The library default behavior of the block might not produce the results you expected, thus causing undesired results in your models. Once the frame bit is removed, you will no longer be able to upgrade your models using the `slupdate` function. Therefore, you should upgrade your existing modes using `slupdate` as soon as possible.

For more detailed information about the specific compatibility considerations related to the R2011a frame-based processing changes, see the following Compatibility Considerations sections.

Blocks with a New Input Processing Parameter

Some DSP System Toolbox blocks are able to process both sample- and frame-based signals. After the transition to the new way of handling frame-based processing, signals will no longer carry information about their frame status. Blocks that can perform both sample- and frame-based processing will require a new parameter that allows you to specify the appropriate processing behavior. To prepare for this change, many blocks are receiving a new **Input processing** parameter. You can set this parameter to `Columns as channels (frame based)` or `Elements as channels (sample based)`, depending upon the type of processing you want. The third choice, `Inherited (this choice will be removed - see release notes)`, is a temporary selection. This additional option will help you to migrate your

existing models from the old paradigm of frame-based processing to the new paradigm.

For a list of blocks that received a new **Input processing** parameter in R2011a, expand the following list.

Blocks with the New Input Processing Parameter

- Arbitrary Response Filter
- Bandpass Filter
- Bandstop Filter
- CIC Compensator
- CIC Filter
- Comb Filter
- Differentiator Filter
- Halfband Filter
- Highpass Filter
- Hilbert Filter
- Inverse Sinc Filter
- Lowpass Filter
- Nyquist Filter
- Octave Filter
- Parametric Equalizer
- Peak-Notch Filter
- Pulse Shaping Filter
- Unwrap

For a list of blocks that received an **Input processing** parameter in R2010b, see the R2010b Signal Processing Blockset Release Notes.

Compatibility Considerations

When you load an existing model R2011a, any block with the new **Input processing** parameter will show a setting of `Inherited (this choice will be removed - see release notes)`. This setting enables your existing models to continue to work as expected until you upgrade them. Although your old models will still work when you open and run them in R2011a, you should upgrade them as soon as possible.

You can upgrade your existing models, using the `slupdate` function. The function detects all blocks that have `Inherited (this choice will be removed - see release notes)` selected for the **Input processing** parameter, and asks you whether you would like to upgrade each block. If you select yes, the function detects the status of the frame bit on the input port of the block. If the frame bit is `1` (frames), the function sets the **Input processing** parameter to `Columns as channels (frame based)`. If the bit is `0` (samples), the function sets the parameter to `Elements as channels (sample based)`.

In a future release, the frame bit and the `Inherited (this choice will be removed - see release notes)` option will be removed. At that time, the **Input processing** parameter in models that have not been upgraded will automatically be set to either `Columns as channels (frame based)` or `Elements as channels (sample based)`, depending on the library default setting for each block. If the library default setting does not match the parameter setting in your model, your model will produce unexpected results. Additionally, after the frame bit is removed, you will no longer be able to upgrade your models using the `slupdate` function. Therefore, you should upgrade your existing modes using `slupdate` as soon as possible.

Changes to the Overlap-Add FFT Filter, Overlap-Save FFT Filter, and Short-Time FFT Blocks

R2011a updates the Overlap-Add FFT Filter, Overlap-Save FFT Filter, and Short-Time FFT blocks to the use new way of frame-based processing. In previous releases, the frame status of the input signal determined how these blocks processed the input. In R2011a, the default behavior of these blocks is to always perform frame-based processing.

Unless you specify otherwise, these blocks now treat each column of the input signal as an individual channel, regardless of its frame status. You can now enable the behavior change in these blocks while still allowing for backward compatibility. This release adds a **Treat Mx1 and unoriented sample-based signals as** parameter for this purpose. This parameter will be removed in a future release, at which point the blocks will always perform frame-based processing.

Compatibility Considerations

The **Treat Mx1 and unoriented sample-based signals as** parameter will be removed in a future release. From that point, the Overlap-Add FFT Filter, Overlap-Save FFT Filter, and Short-Time FFT blocks will always perform frame-based processing.

You can use the `slupdate` function to upgrade your existing models that contain one of these blocks. The function detects all Overlap-Add FFT Filter, Overlap-Save FFT Filter, and Short-Time FFT blocks in your model. Then, if you allow it to, `slupdate` performs the following actions:

- If the input to the block is an *M*-by-1 or unoriented sample-based signal, the `slupdate` function:
  - Places a Transpose block in front of the affected block in your model. This block transposes the *M*-by-1 or unoriented sample-based input into a 1-by-*M* row vector. By converting the input to a row vector, the block continues to produce the same results as in previous releases (an M_o-by-M_i output).
  - Sets the **Treat Mx1 and unoriented sample-based signals as** parameter to `One channel`. This setting ensures that your model will continue to produce the same results when the **Treat Mx1 and unoriented sample-based signals as** parameter is removed in a future release.
- If the input to the block is *not* an *M*-by-1 or unoriented sample-based signal, the `slupdate` function sets the **Treat Mx1 and unoriented sample-based signals as** parameter to `One channel`. This setting does not affect the behavior of your current model. However, the change does ensure that your model will continue to produce the same results when the

Treat Mx1 and unoriented sample-based signals as parameter is removed in a future release.

Difference Block Changes

R2011a adds a new **Running difference** parameter to the Difference block.

Compatibility Considerations

In a future release, the following option for the **Running difference** parameter will be removed: `Inherit from input (this choice will be removed see release notes)`. From this time forward, you must specify whether or not the block computes a running difference; the block will no longer make that choice based on the status of the frame bit.

You can use the `slupdate` function to upgrade your existing models that contain a Difference block. The function detects whether your models contain any Difference blocks with the **Running difference** parameter set to `Inherit from input (this choice will be removed see release notes)`. If you do, the function detects the status of the frame bit, and sets the **Running difference** parameter accordingly.

Signal To Workspace Block Changes

R2011a updates the Signal To Workspace block. The block now allows you to choose an output format using the **Save format** parameter. You can choose to save your data as an `Array`, `Structure`, or `Structure with time`.

Additionally, the old **Frames** parameter has been replaced by a new **Save 2-D signals as** parameter. This parameter allows you to specify whether the block saves 2-D signals as a 2-D array, or as a 3-D array. To provide for backward compatibility, the **Save 2-D signals as** parameter also has an option `Inherit from input (this choice will be removed see release notes)`. When you select this option, the block saves sample-based data as a 3-D array and frame-based data as a 2-D array.

Compatibility Considerations

In a future release, the following option will be removed: `Inherit from input (this choice will be removed see release notes)`. From this time forward, you must specify whether the block saves signals as a 2-D or 3-D array. The block will no longer make that choice based on the status of the frame bit.

You can use the `slupdate` function to upgrade your existing models that contain a Signal To Workspace block. The function detects whether your models contain any Signal To Workspace blocks with the **Save 2-D signals as** parameter set to `Inherit from input (this choice will be removed see release notes)`. If you do, the function detects the status of the frame bit and sets the **Save 2-D signals as** parameter accordingly.

- If the input signal is frame based, the function sets the **Save 2-D signals as** parameter to `2-D array (concatenate along first dimension)`.
- If the input signal is sample based, the function sets the **Save 2-D signals as** parameter to `3-D array (concatenate along third dimension)`.

Spectrum Scope Block Changes

R2011a updates the Spectrum Scope block to use the new way of frame-based processing. To enable this change, the block received a new **Treat Mx1 and unoriented sample-based signals as** parameter. This new parameter is available only when you select the **Buffer input** check box. By default, the new parameter is set to `One channel`. In this mode, the block treats *M*-by-1 and unoriented sample-based input as a single column vector and buffers the input along that column.

Sample-Based Row Vector Processing Changes

In previous releases, some DSP System Toolbox blocks handled sample-based row vector inputs in a special way. Of the blocks that can treat sample-based row vector inputs differently, there are two categories:

- Some blocks have a **Treat sample-based row input as a column** check box which allows you to explicitly specify how the block should treat sample-based row vector inputs. Expand the following section for a full list of these blocks.

Blocks with a Check Box

- Maximum
- Mean
- Median
- Minimum
- Normalization
- RMS
- Standard Deviation
- Variance

- Other blocks automatically treat a sample-based row vector input as a single channel (column vector). Expand the following section for a full list of these blocks.

Blocks That Implicitly Treat Sample-Based Row Vectors as a Single Channel

- Autocorrelation
- Autocorrelation LPC
- Burg AR Estimator
- Burg Method
- Complex Cepstrum
- Convolution
- Correlation
- Covariance AR Estimator
- Covariance Method
- DCT
- FFT
- IDCT
- IFFT

- Interpolation
- Levinson-Durbin
- LPC to LSF/LSP Conversion
- LPC to/from Cepstral Coefficients
- LPC to/from RC
- LPC/RC to Autocorrelation
- LSF/LSP to LPC Conversion
- Modified Covariance AR Estimator
- Modified Covariance Method
- Peak Finder
- Polynomial Stability Test
- Real Cepstrum
- Sort
- Window Function
- Yule-Walker AR Estimator
- Yule-Walker Method

The special treatment of sample-based row vector inputs will be removed in a future release. See the compatibility considerations for more information about how this change will affect your models.

Compatibility Considerations

The blocks listed will continue to work as expected in R2011a. However, in a future release these blocks will produce a warning when you provide them with a sample-based row vector input, and eventually, their behavior will change.

You can prepare your models for the upcoming change by running the `slupdate` function. If the function detects any blocks that have a **Treat sample-based row input as a column** check box, it performs the following actions:

- If the input to the block is a sample-based row vector, and the **Treat sample-based row input as a column** check box is selected, the `slupdate` function places a Transpose block in front of the affected block. The Transpose block transposes the sample-based row vector into a column vector, which is then input into the affected block. Transposing the input signal ensures that your model will produce the same results in future releases.
- If the **Treat sample-based row input as a column** check box is not selected, or if the input to the block is not a sample-based row vector, the `slupdate` function takes no action. Your model will continue to work as expected in future releases.

If the `slupdate` function detects any blocks that automatically treat sample-based row vectors as a column, it performs the following actions:

- If the input to the block is a sample-based row vector, the `slupdate` function places a Transpose block in front of the affected block. The Transpose block transposes the sample-based row vector into a column vector, which is then input into the affected block. Transposing the input signal ensures that your model will produce the same results in future releases.
- If the input to the block is not a sample-based row vector, the `slupdate` function takes no action. Your model will continue to work as expected in future releases.

New Function for Changing the System Object Package Name from signalblks to dsp

Compatibility Considerations: Yes

In R2010b, the package name of Signal Processing Blockset™ System objects changed from `signalblks` to `dsp`. In R2011a, a new function is available to help you update your code. You can use the `sysobjupdate` function to recursively search a folder and its subfolders for MATLAB files that contain System object packages, classes, and properties that have been renamed.

Compatibility Considerations

If you have any existing System object code that uses a package name of `signalblks`, you should use the `sysobjupdate` function to update your code. For more information, type help sysobjupdate at the MATLAB command line.

New Discrete FIR Filter Block

R2011a adds a new Discrete FIR Filter block to the DSP System Toolbox Filtering/Filter Implementations library. The block is an implementation of the Simulink Discrete FIR Filter block.

New Printing Capability from the Time Scope Block

You can now print the data you see in the Time Scope block. To send the data to your printer, select **File > Print ...** from the scope menu. To print the data to a MATLAB figure, select **File > Print to Figure**.

Improved Display Updates for the Time Scope Block and System Object

R2011a introduces the capability to improve the performance of the Time Scope block and `dsp.TimeScope` System object by reducing the frequency with which the display updates. You can now choose between this new enhanced performance mode and the old behavior by selecting **Reduce Updates to Improve Performance** from the **Simulation** menu of the block, or the **Playback** menu of the System object. By default, both the block and System object operate in the new enhanced performance mode.

New Implementation Options Added to Blocks in the Filter Designs Library

Compatibility Considerations: Yes

This release provides filter customization options for blocks in the Filtering/Filter Designs library. You can access these options in the **Filter implementation** section of the block dialog box:

- Implement designed filters as Simulink basic elements or as a digital filter.
- Customize filters built using Simulink basic elements using the **Optimizations** parameters.

Blocks in the Filtering/Filter Designs library also support **Input processing** and **Rate options** parameters in R2011a. For more information, see "Blocks with a New Input Processing Parameter" on page 82.

Compatibility Considerations

- Frame-based processing and filters with algebraic loops — For filters that contain sample-by-sample feedback, using a lumped-element implementation instead of Simulink basic elements can eliminate algebraic loops. For supported blocks, use the `slupdate` function on older models with designed filters to convert the designed filters into lumped filters. You can enable this feature manually by clearing the **Use basic elements for filter customization** check box.

  For filters with algebraic loops that do not have this option, specify sample-based processing by setting the **Input processing** parameter to `Elements as channels (sample based)`.
- **Rate Options** parameter — Filters that allow multirate processing, such as FIR decimators and interpolaters, perform single-rate processing by default. For more information, see the block reference pages.

New dsp.DigitalDownConverter and dsp.DigitalUpConverter System Objects

This release adds new `dsp.DigitalDownConverter` and `dsp.DigitalUpConverter` System objects. The digital up converter (DUC) and digital down converter (DDC) System objects provide tools to design interpolation/decimation filters and simplify the steps required to implement the up/down conversion process.

Improved Performance of FFT Implementation with FFTW library

The FFT, IFFT blocks include the use of the FFTW library.

Variable-Size Support for System Objects

Compatibility Considerations: Yes

The following System objects support inputs that change their size at runtime.

- `dsp.ArrayVectorAdder`
- `dsp.ArrayVectorDivider`
- `dsp.ArrayVectorMultiplier`
- `dsp.ArrayVectorSubtractor`
- `dsp.FFT`
- `dsp.IFFT`
- `dsp.Maximum`
- `dsp.Mean`
- `dsp.Minimum`
- `dsp.Normalizer`
- `dsp.RMS`
- `dsp.StandardDeviation`
- `dsp.UDPReceiver`
- `dsp.UDPSender`
- `dsp.Variance`

Compatibility Considerations

For the `dsp.UDPSender` and `dsp.UDPReceiver` System objects only, you should update your code to stop sending or receiving any data length settings.

Support for variable-size data makes the data length settings redundant. For example,

```
% Change these lines to remove explicit lengths:
     step(hudps, dataSent, dataLength);
     [dataReceived len] = step(hudpr);
     bytesReceived = bytesReceived + ...
        length(dataReceived) len;

% Code lines with lengths removed:
  step(hudps,datasent);
   [dataReceived] = step(hudpr);
   bytesReceived = bytesReceived + ...
      length(dataReceived);
```

System Objects FullPrecisionOverride Property Added

Compatibility Considerations: Yes

A `FullPrecisionOverride` property has been added to the System objects listed below. This property is a convenient way to control whether the object uses full precision to process fixed-point input.

When you set this property to `true`, which is the default, it eliminates the need to set many fixed-point properties individually. It also hides the display of these properties (such as `RoundingMode`, `OverflowAction`, etc.) because they are no longer applicable individually.

To set individual fixed-point properties, you must first set `FullPrecisionOverride` to `false`.

Note The `CoefficientDataType` property is not controlled by `FullPrecisionOverride`

The following System objects are affected:

- `dsp.ArrayVectorAdder`

- `dsp.ArrayVectorSubtractor`
- `dsp.Autocorrelator`
- `dsp.Convolver`
- `dsp.Crosscorrelator`
- `dsp.FIRDecimator`
- `dsp.FIRInterpolator`
- `dsp.FIRRateConverter`
- `dsp.SubbandAnalysisFilter`
- `dsp.SubbandSynthesisFilter`
- `dsp.Window`

Compatibility Considerations

All of these System objects have their new `FullPrecisionOverride` property set to the default, `true`. If you had set any fixed-point properties to non-default values for these objects, those values are ignored. As a result, you may see different numerical answers from those answers in a previous release. To use your nondefault fixed-point settings, you must first change `FullPrecisionOverride` to `false`.

'Internal rule' System Object Property Value Changed to 'Full precision'

Compatibility Considerations: Yes

To clarify the value of many `DataType` properties, the `'Internal rule'` option has been changed to `'Full precision'`.

Compatibility Considerations

The objects allow you to enter either `'Internal rule'` or `'Full precision'`. If you enter `'Internal rule'`, that option is stored as `'Full precision'`.

MATLAB Compiler Support for System Objects

The DSP System Toolbox supports the MATLAB Compiler for most System objects. With this capability, you can use the MATLAB Compiler to take MATLAB files, which can include System objects, as input and generate standalone applications.

The following System objects are not supported by the MATLAB Compiler software:

- `dsp.CICDecimator`
- `dsp.CICInterpolator`
- `dsp.DigitalDownConverter`
- `dsp.DigitalUpConverter`
- `dsp.TimeScope`

Viewing System Objects in the MATLAB Variable Editor

The MATLAB Variable Editor now displays System objects properties in the same order as they display at the command line. Note that the Variable Editor provides a read-only view for System objects.

System Object Input and Property Warnings Changed to Errors

Compatibility Considerations: Yes

When a System object is locked (e.g., after the `step` method has been called), the following situations now produce an error. This change prevents the loss of state information.

- Changing the input data type
- Changing the number of input dimensions
- Changing the input complexity from real to complex
- Changing the data type, dimension, or complexity of tunable property

- Changing the value of a nontunable property

Compatibility Considerations

Previously, the object issued a warning for these situations. The object then unlocked, reset its state information, relocked, and continued processing. To update existing code so that it does not produce an error, use the `release` method before changing any of the items listed above.

New and Updated Demos

R2011a adds the following new demos:

- Digital Up and Down Conversion for Family Radio Service — Shows you how to use the new `dsp.DigitalDownConverter` and `dsp.DigitalUpConverter` System objects to design a Family Radio Service (FRS) transmitter and receiver.
- Design and Analysis of a Digital Down Converter — Shows you how to use the `dsp.DigitalDownConverter` System object to simplify the steps required to emulate the TI Graychip 4016 digital down converter.
- Using System Objects with MATLAB Compiler — Shows you how to use MATLAB Compiler to create a standalone application from MATLAB System objects.

Additionally, the Simulink-based demo, GSM Digital Down Converter, has been enhanced to use the Fixed-Point Toolbox™ `cordicrotate` function. The demo now allows you to compare an NCO-based mixer to a CORDIC-based mixer.

Documentation Examples Renamed

Compatibility Considerations: Yes

In previous releases, the example models used throughout the Signal Processing Blockset™ documentation were named with a prefix of `doc_`. In R2011a, this prefix has changed to `ex_`. For example, in R2010b, you could launch an example model using the Time Scope block by typing

`doc_timescope_tut` at the MATLAB command line. To launch the same model in R2011a, you must type ex_timescope_tut at the command line.

Compatibility Considerations

You can no longer launch DSP System Toolbox documentation example models using the `doc_` name. To open these models in R2011a, you must replace the `doc_` prefix in the model name with `ex_`.

Downsample Block No Longer Has Frame-Based Processing Latency for a Frame Size of One

Compatibility Considerations: Yes

As of R2011a, the Downsample block no longer exhibits frame-based processing latency when the input frame size is one.

Compatibility Considerations

Existing models that use the Downsample block in frame-based processing mode may produce different results in R2011a. Specifically, the Downsample block no longer has one-frame of latency when the input frame size is one. If your model uses a Downsample block in frame-based processing mode and the input frame size is one, you will see different results when you run your model in R2011a. If you need to restore the one-frame latency, you can use a Delay block to delay the output of the Downsample block by one frame.

SignalReader System Object Accepts Column Input Only

Compatibility Considerations: Yes

The `SignalReader` System object now accepts column inputs only.

Compatibility Considerations

Update any code with row input to the `SignalReader` object to convert the input to column form before passing it to the object. (Note that this change occurred in R2010b.)

FrameBasedProcessing Property Removed from the dsp.DelayLine and dsp.Normalizer System Objects

Compatibility Considerations: Yes

In R2010b, the `FrameBasedProcessing` property was removed from the `dsp.DelayLine` and `dsp.Normalizer` System objects. Both objects now treat each column of the input as a separate channel (frame-based processing).

Compatibility Considerations

As of R2010b, MATLAB issues a warning when you set the `FrameBasedProcessing` property of the `dsp.DelayLine` or `dsp.Normalizer` System objects.

R2010a MAT Files with System Objects Load Incorrectly

Compatibility Considerations: Yes

If you saved a System object to a MAT file in R2010a and load that file in R2011a, MATLAB may display a warning that the constructor must preserve the class of the returned object. This occurs because an aspect of the class definition changed for that object in R2011a. The object's saved property settings may not restore correctly.

Compatibility Considerations

MAT files containing a System object saved in R2010a may not load correctly in R2011a. You should recreate the object with the desired property values and save the MAT file.