DSP Blockset Release Notes

The "DSP Blockset 5.0 Release Notes" on page 1-1 describe the changes introduced in the latest version of the DSP Blockset. The DSP Blockset Release Notes also provide information about the earlier versions of the product, in case you are upgrading from a version that was released prior to Release 12.1.

The following topics are discussed in these Release Notes:

- "New Features" on page 1-2
- "Major Bug Fixes" on page 1-17
- "Upgrading from an Earlier Release" on page 1-18
- "Platform Limitations" on page 1-22
- "Known Software Problems" on page 1-23

If you are upgrading from a release earlier than Release 12.1, you should also see these sections:

- "DSP Blockset 4.1 Release Notes" on page 2-1
- "DSP Blockset 4.0 Release Notes" on page 3-1

If you are upgrading from a release prior to Release 11, see "DSP Blockset Notes from Earlier Releases" on page 4-1.

Printing the Release Notes

If you would like to print the Release Notes, you can link to a PDF version.

Contents

1 DSP Blockset 5.0 Release Notes

2 DSP Blockset 4.1 Release Notes

3 DSP Blockset 4.0 Release Notes

4 DSP Blockset Notes from Earlier Releases

DSP Blockset 5.0 Release Notes

New Features

This section introduces the new features and enhancements added to the DSP Blockset 5.0 (Release 13) since DSP Blockset 4.1 (Release 12.1).

If you are upgrading from a release earlier than Release 12.1, then you should see “New Features” on page 2-2 of the Release Notes for DSP Blockset 4.1.

Two major new features of DSP Blockset 5.0 include the following:

- Full single-precision support in all blocks
- Full support of the embedded real-time (ERT) generated ANSI C code (requires the Real-Time Workshop Embedded Coder)

For details about these and other new features and enhancements, see the following topics:

- “Full Single-Precision Support” on page 1-3
- “Full Support of Embedded Real-Time (ERT) C Code Generation” on page 1-3
- “Smaller, Faster Generated C Code That Requires Less RAM” on page 1-3
- “Full Boolean Data Type Support” on page 1-4
- “Expanding Fixed-Point Data Type Support” on page 1-4
- “New Blocks” on page 1-6
- “Enhanced Blocks” on page 1-9
- “New and Enhanced Demos” on page 1-14
- “Audio Blocks Relocated” on page 1-21
- “New Options for Event Detection” on page 1-15

The increased support for single-precision computation and ERT code generation, together with other major enhancements, provide an efficient solution for the design and implementation of complete, high-performance floating-point DSP systems.

Full Single-Precision Support

All DSP Blockset blocks now support single-precision floating-point computation in both simulation and Real-Time Workshop C code generation.

To learn more about data type support in the DSP Blockset, see the topic on data type support in the DSP Blockset documentation.

Full Support of Embedded Real-Time (ERT) C Code Generation

All DSP Blockset blocks now support embedded real-time (ERT) ANSI C code generation (requires the Real-Time Workshop Embedded Coder).

To learn more about C code generation in the DSP Blockset, see the topic on code generation support in the DSP Blockset documentation.

Smaller, Faster Generated C Code That Requires Less RAM

Real-Time Workshop (RTW) ANSI C code generated from DSP Blockset blocks is now smaller, faster, and requires less RAM, largely due to the following enhancements:

- **Function reuse (run-time libraries)** — The code generated from most blocks now *reuses* common algorithmic functions from ANSI C run-time libraries, leading to smaller generated code. In addition, the functions in the run-time libraries are highly optimized, resulting in faster generated code that requires less RAM.
- **Parameter reuse (RTW run-time parameters)** — For most blocks, if there are multiple instances of a block that all have the same value for a specific parameter, each block instance points to the same variable in the generated code, thereby reducing memory requirements.

For more information, see the topic on code generation support in the DSP Blockset documentation.

Full Boolean Data Type Support

All block input ports that accept logical signals now support the Boolean data type. In addition, for all outputs that are logical signals, the default data type is now Boolean.

For a list of DSP Blockset blocks that support Boolean data types, see the topic on Boolean data type support in the DSP Blockset documentation.

In some cases, you may want to override the Simulink default and *disable* Boolean support, as described in "New Default Setting Enables Boolean Data Type Support" on page 1-19. For more information about disabling Boolean support, see the following topics in the DSP Blockset documentation:

- Steps to disable Boolean support
- The effects of enabling and disabling Boolean support

Expanding Fixed-Point Data Type Support

The DSP Blockset 5.0 includes more blocks that support fixed-point data types, including some source blocks. Support of fixed-point data types will continue to expand in future releases.

The following blocks are all the blocks in the DSP Blockset that support fixed-point data types. You can use all of these blocks with Fixed-Point Blockset blocks, and with Simulink blocks that support fixed-point data types. These blocks are colored orange in the DSP Blockset library. (Some of the blocks are Simulink blocks that are available in the DSP Blockset libraries as well as the Simulink libraries.)

To take full advantage of DSP Blockset fixed-point capabilities, you must install the Fixed-Point Blockset. For more information, see the topic on licensing information in the Fixed-Point Blockset documentation.

For more information on the DSP Blockset blocks that support fixed-point, see the topic on fixed-point support in the DSP Blockset documentation.

- Buffer
- Check Signal Attributes
- Constant Diagonal Matrix
- Convert 1-D to 2-D
- Convert 2-D to 1-D
- Create Diagonal Matrix
- Data Type Conversion (Simulink block)
- Delay Line
- Discrete Impulse
- Display (Simulink block)
- Downsample
- DSP Constant
- Edge Detector
- Extract Diagonal
- Extract Triangular Matrix
- Filter Realization Wizard
- Flip
- Frame Status Conversion
- Identity Matrix
- Inherit Complexity
- Integer Delay
- Matrix Concatenation (Simulink block)
- Matrix Viewer
- Maximum
- Minimum
- Multiphase Clock
- Multiport Selector
- N-Sample Enable
- N-Sample Switch
- Overwrite values
- Pad
- Permute Matrix
- Queue
- Repeat
- Sample and Hold
- Selector (Simulink block)
- Signal To Workspace
- Sine Wave
- Spectrum Scope
- Stack
- Submatrix
- Time Scope (Simulink block)
- Toeplitz
- Transpose
- Triggered Delay Line
- Triggered To Workspace
- Unbuffer
- Upsample
- Variable Integer Delay
- Variable Selector
- Vector Scope
- Zero Pad

New Blocks

The key new block for Release 13 is the Digital Filter block. For descriptions of this and other new blocks and links to their online reference pages, see the following sections:

- "Cumulative Product" on page 1-6
- "Digital Filter" on page 1-6
- "DWT and IDWT" on page 1-7
- "Interpolation" on page 1-7
- "LPC to LSF/LSP Conversion and LSF/LSP to LPC Conversion" on page 1-7
- "Overwrite Values" on page 1-8
- "Two-Channel Analysis Subband Filter and Two-Channel Synthesis Subband Filter" on page 1-8

Cumulative Product

The new Cumulative Product block is in the Math Operations library. This block computes the cumulative product of the row or column elements of the input matrix.

Digital Filter

The new Digital Filter block in the Filter Designs library is ideal for implementing digital filters that you have already designed. The block supports a variety of filter structures, and can implement *static filters* with fixed coefficients, as well as *time-varying filters* with coefficients that change over time. Using the filter you specify, the block individually filters each channel of the input signal and outputs the result. The block's output numerically matches the output of the `filter` function in the Filter Design Toolbox and the `filter` function in the Signal Processing Toolbox.

You can use the Digital Filter block to efficiently simulate the numerical behavior of your filter on a single- or double-precision floating-point system such as a personal computer or DSP chip. You can also use the block to generate highly optimized Real-Time Workshop ANSI C code for use in single- and double-precision floating-point embedded systems. (Note that

To implement a filter with the Digital Filter block, you must provide the following basic information about your filter:

- Whether the filter transfer function is FIR with all zeros, IIR with all poles, or IIR with poles and zeros
- The desired filter structure
- The filter coefficients (entered as a parameter, or from an additional port, which is useful for time-varying filters)

The block supports the following filter structures:

- Direct form
- Direct form I
- Direct form II
- Transposed direct form
- Direct form I transposed
- Direct form II transposed
- Biquadratic direct form II transposed (second order sections)
- Lattice AR
- Lattice MA

DWT and IDWT

The new DWT block in the Transforms library computes the discrete wavelet transform (DWT) of its input. The IDWT block computes the *inverse* DWT of its input. These blocks are identical to the Dyadic Analysis Filter Bank and Dyadic Synthesis Filter Bank blocks in the Multirate Filters library.

The DWT block uses a filter bank with specified highpass and lowpass FIR filters to decompose the input signal into subbands that have smaller bandwidths and slower sample rates. Similarly, the IDWT block uses a filter bank to reconstruct the original signal from the subbands output by the DWT block.

The filter bank filters in both blocks can be user-defined or wavelet-based. Use of wavelet-based filters requires the Wavelet Toolbox.

Interpolation

The new Interpolation block in the Signal Operations library interpolates real-valued input signals at points between sample times by using linear or FIR interpolation. You specify the times at which the block should interpolate points, and the block outputs the interpolated values.

LPC to LSF/LSP Conversion and LSF/LSP to LPC Conversion

The new LPC to LSF/LSP Conversion block and LSF/LSP to LPC Conversion block in the Linear Prediction library are ideal for speech applications such as speech coding and speech recognition. The first block allows you to convert linear prediction coefficients (LPCs) to line spectral pairs (LSPs) or line spectral frequencies (LSFs). The second block converts LSFs or LSPs back to LPCs. When converting LPCs, you can adjust the accuracy of the LSF or LSP output and set the block to flag invalid outputs due to unstable polynomial inputs.

Overwrite Values

The new Overwrite Values block in the Matrix Operations library overwrites either a specified submatrix of the input matrix or a specified portion of the input's main diagonal.

Two-Channel Analysis Subband Filter and Two-Channel Synthesis Subband Filter

The new Two-Channel Analysis Subband Filter block in the Multirate Filters library decomposes a signal into a high-frequency subband and a low-frequency subband using the specified highpass and lowpass FIR filters. Each subband has half the bandwidth and half the sample rate of the original signal.

Similarly, the new Two-Channel Synthesis Subband Filter block uses a filter bank to reconstruct the original signal from the high-frequency and a low-frequency subbands output by the Two-Channel Analysis Subband Filter block.

Enhanced Blocks

For summaries of individual block enhancements for Release 13 and links to their online reference pages, see the following sections:

- "Audio Blocks — New 24- and 32-Bit Support" on page 1-9
- "Autocorrelation, Correlation, Convolution — New Optimization Options" on page 1-10
- "Cumulative Sum — Intraframe Running Sums of Frame-Based Columns" on page 1-10
- "DCT and IDCT — New Optimization Options" on page 1-10
- "Digital Filter Design — Supports More Filter Structures" on page 1-10
- "Dyadic Analysis Filter Bank and Dyadic Synthesis Filter Bank Enhancements" on page 1-11
- "Filter Realization Wizard — New Filter Design Options, Better Fixed-Point Support, More Structures" on page 1-11
- "Random Source — Better Random Signals and New Option" on page 1-13
- "Sine Wave — Accepts Zero and Negative Frequencies" on page 1-13

Audio Blocks — New 24- and 32-Bit Support

The To Wave Device and From Wave Device blocks now support 24-bit audio devices in addition to 8- and 16-bit audio devices.

The To Wave File and From Wave File blocks now support 24 bits per sample and 32 bits per sample in addition to the previous 8- and 16-bit support.

All four audio blocks are now located in a new library. For more information, see "Audio Blocks Relocated" on page 1-21.

Autocorrelation, Correlation, Convolution — New Optimization Options

The Autocorrelation, Correlation, and Convolution blocks used to compute only in the time domain. Now you can set their computation domains to one of the following:

- **Time domain** — Minimizes memory use
- **Frequency domain** — Depending on the input length, may require fewer computations than computing in the time domain

The Correlation and Convolution blocks also have a third computation domain option, **Fastest domain**, which computes in the domain that minimizes the number of computations (time domain or frequency domain). The Autocorrelation block does not yet support this option.

Cumulative Sum — Intraframe Running Sums of Frame-Based Columns

The Cumulative Sum block can now compute a running sum of each column of a frame-based input, independent of the running sum of columns of previous inputs.

DCT and IDCT — New Optimization Options

The DCT and IDCT blocks use sine and cosine values to compute the discrete cosine transform and its inverse. You can now set the blocks to compute sines and cosines in one of the following ways:

- **Table lookup** — Look up sine and cosine values in a speed-optimized table
- **Trigonometric function calls** — Make sine and cosine function calls

Digital Filter Design — Supports More Filter Structures

The Digital Filter Design block, which previously supported only the direct form II transposed structure, now supports all of the following structures:

- Direct form I
- Direct form II
- Direct form I transposed
- Direct form II transposed
- Direct form FIR
- Direct form FIR transposed

Dyadic Analysis Filter Bank and Dyadic Synthesis Filter Bank Enhancements

The Dyadic Analysis Filter Bank block and Dyadic Synthesis Filter Bank block (referred to as "analysis block" and "synthesis block") share several changes and enhancements described below.

New Wavelet Option. Both blocks now allow you to specify wavelet-based highpass and lowpass filters in addition to the previously supported user-defined filters. Use of wavelets requires the Wavelet Toolbox.

New Option for Single-Port Inputs and Outputs. Previously, the analysis block output each subband on a separate output port and the synthesis block accepted each subband through a separate input port. Now the analysis block provides an option to output a single vector or matrix of concatenated subbands, and the synthesis block provides an option to accept such inputs through a single input port.

Frame-Based Support Only. The analysis block now accepts only frame-based input signals. The synthesis block now always outputs frame-based signals. To decompose or reconstruct a sample-based signal using filter banks, you can create your own filter banks using the new Two-Channel Analysis Subband Filter and Two-Channel Synthesis Subband Filter blocks in the Multirate Filters library (see "Two-Channel Analysis Subband Filter and Two-Channel Synthesis Subband Filter" on page 1-8).

Filter Realization Wizard — New Filter Design Options, Better Fixed-Point Support, More Structures

The Filter Realization Wizard block has several significant enhancements described below.

New Interface Provides Filter Design and Analysis Options. The Filter Realization Wizard block interface is now a part of the Filter Design and Analysis Tool (FDATool) GUI. Previously, the block required you to specify the filter by typing in its coefficients (you had to predesign the filter elsewhere). You now have the option to design your filter within the block; the extensive set of filter design and analysis tools in FDATool allow the block to automatically implement your filter design without you having to type its coefficients.

Enhanced Fixed-Point Support. The Filter Realization Wizard block now better supports fixed-point filters due to the following enhancements:

- **New fixed-point filter design capabilities** — The block now allows you to design fixed-point filters by using the **Set Quantization Parameters** panel in the new block interface. Use of this panel requires the Filter Design Toolbox.
- **Better fixed-point filter implementation** — The block now implements a much better fixed-point filter when you install the Fixed-Point Blockset and Filter Design Toolbox. To implement a good fixed-point filter, you should first design a fixed-point filter as described above and then set the block to implement the filter using Fixed-Point Blockset blocks.

Corresponding dfilt and qfilt Methods. `dfilt` and `qfilt` objects now have a new `realizemdl` method. This method allows you to access the filter realization capabilities of the Filter Realization Wizard block from the command line.

Supports More Filter Structures. The block can now realize filters using any of the following filter structures:

- Direct form I
- Direct form II
- Direct form I transposed
- Direct form II transposed
- Second order sections for direct form I and II, and their transposes
- Direct form FIR
- Direct form FIR transposed
- Direct form asymmetric FIR
- Direct form symmetric FIR
- Lattice ARMA
- Lattice AR
- Lattice MA (same as lattice minimum phase)
- Lattice all-pass
- Lattice maximum phase
- Cascade
- Parallel

You may not be able to directly access some of the above structures through the block interface, but you *can* access all of them by creating a `qfilt` or `dfilt` object with the desired structure, and then importing the filter into the block.

Random Source — Better Random Signals and New Option

The Random Source block now outputs much better random signals due to the following algorithmic enhancements:

- **Uses improved random generators** — The block now uses the same random generator algorithms as the current `rand` and `randn` functions. These generators yield much better random signals than the MATLAB Version 4 generators that were previously used by the Random Source block.
- **Better automatic seed generation** — When you opt to specify a single initial seed for a multichannel output, the block automatically generates an initial seed for each channel (using the seed you provide). The block now generates these initial seeds using an improved method that results in better random signals.
- **Better complex random signal generation** — The block now generates complex random values using an improved method that results in better complex random signals.

The Random Source block also now offers a second method of computing Gaussian (normally distributed) random signals based on the central limit theorem. When set to use this new method, the block computes Gaussian random values by adding and scaling uniformly distributed random signals. The block allows you to specify the number of uniform values to sum.

Sine Wave — Accepts Zero and Negative Frequencies

The Sine Wave block now accepts zero and negative frequency values in addition to positive frequency values.

New and Enhanced Demos

You can access all DSP Blockset demos (in the Help browser **Demo** tab) by typing `demo blockset dsp` at the command line. To open the new and enhanced demos, you can also click the links in the following table in the MATLAB Help browser (*not* in a Web browser).

New and Enhanced Demos	Location in DSP Blockset Entry of Help Browser Demo Tab	Enhancement Description
GSM Digital Down Converter	`Communications`	New
FIR Interpolation	`Filtering`	These demos now provide a fixed-point version that you can run when you install the Fixed-Point Blockset.
Denoising	`Wavelets`	
Wavelet Transmultiplexer (WTM)	`Wavelets`	
Multi-Level PR Filter Bank	`Wavelets`	Formerly a 1-level filter bank demo, this demo now demonstrates a three-level perfect reconstruction wavelet-based filter bank. This demo also provides a new fixed-point version that you can run when you install the Fixed-Point Blockset.

Audio Blocks Relocated to New Block Library

The four audio blocks, formerly in the DSP Sources and DSP Sinks libraries, are now in the new Windows (WIN32) library (a sublibrary of the new Platform-specific I/O library).

New DSP Blockset Block Libraries

- **Platform-specific I/O** — New block library that contains sublibraries for use with specific platforms such as the Windows operating system. To see the library, type `dsppio`, or type `simulink` and navigate to the library using the Simulink Library Browser.
- **Windows (WIN32)** — Sublibrary of the Platform-specific I/O library. Contains blocks (listed below) for use with 32-bit Windows operating systems. To see the library, type `dspwin32`, or type `simulink` and navigate to the library using the Simulink Library Browser.

Blocks Relocated to the Windows (WIN32) Library

- To Wave File
- To Wave Device
- From Wave File
- From Wave Device

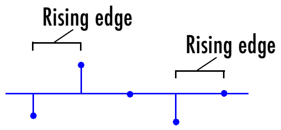
New Options for Event Detection

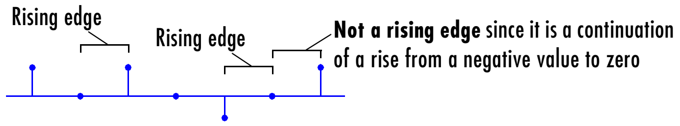
The following blocks perform an operation when an event-detecting input port (such as a clock or reset port) detects an event:

- Counter
- Cumulative Product
- Cumulative Sum
- Histogram
- Integer Delay
- Maximum
- Mean
- Minimum
- N-Sample Enable
- Queue
- RMS
- Stack
- Standard Deviation
- Variance

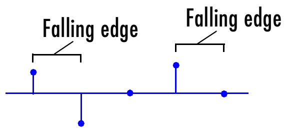
You can now set the event-detecting ports of the above blocks to respond to one of the following events:

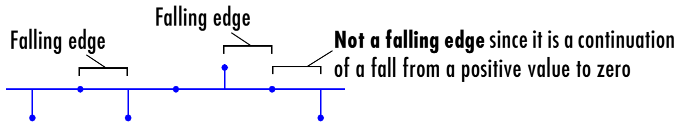
- **Non-zero sample** — When the input sample is nonzero
- **Rising edge** — When the input does one of the following:
  - Rises from a negative value to a positive value or zero
  - Rises from zero to a positive value, where the rise is not a continuation of a rise from a negative value to zero (see the following figure)




Rising edge
Rising edge
Not a rising edge since it is a continuation of a rise from a negative value to zero

- **Falling edge** — When the input does one of the following:
  - Falls from a positive value to a negative value or zero
  - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to zero (see the following figure)




Falling edge
Falling edge
Not a falling edge since it is a continuation of a fall from a positive value to zero

- **Either edge** — When an input is a **Rising edge** or **Falling edge** (as described above)

Major Bug Fixes

DSP Blockset 5.0 includes several bug fixes made since Version 4.1. This section describes the particularly important Version 5.0 bug fixes.

If you are viewing these Release Notes in PDF form, please refer to the HTML form of the Release Notes, using either the Help browser or the MathWorks Web site and use the link provided.

If you are upgrading from a release earlier than Release 12.1, then you should also see “Major Bug Fixes” on page 2-10.

Upgrading from an Earlier Release

The following topics describe the upgrade issues involved in moving from the DSP Blockset 4.1 to Version 5.0:

- “New Default Setting Enables Boolean Data Type Support” on page 1-19
- “Replaced Filtering Blocks” on page 1-20
- “Wavelet Analysis and Wavelet Synthesis Blocks Replaced” on page 1-20
- “Cumulative Sum Block Behaves Differently” on page 1-21
- “Contiguous Copy Block Obsolete” on page 1-21
- “Audio Blocks Relocated” on page 1-21

If you are upgrading from a version earlier than 4.0, then you should see “Upgrading from an Earlier Release” on page 3-12 of the DSP Blockset 4.0 Release Notes.

New Default Setting Enables Boolean Data Type Support

The Simulink default settings now *enable* Boolean data type support. This allows you to take advantage of the full Boolean data type support in DSP Blockset 5.0.

In some cases, you may want to override the Simulink default and *disable* Boolean support. For instance, for each of the following blocks, the default data type of at least one output has now changed from double precision to Boolean:

- Counter
- Edge Detector
- Event-Count Comparator
- Multiphase Clock
- N-Sample Enable
- Queue
- Stack

If you have a model that uses previous versions of the above blocks, their new default Boolean data type can potentially break your model. If the change in data type does break your model, you can fix this by disabling Boolean support; then the output data type of the blocks will remain double precision, and your model will behave just as it did before.

For more information about disabling Boolean support, see the following topics in the DSP Blockset documentation:

- Steps to disable Boolean support
- The effects of enabling and disabling Boolean support

For more information about the new Boolean data type support in DSP Blockset 5.0, see the following topics:

- "Full Boolean Data Type Support" on page 1-4 — Description of the new full support of the Boolean data type in DSP Blockset 5.0.
- A list of all DSP Blockset blocks that support Boolean data types in the DSP Blockset documentation.

Replaced Filtering Blocks

The new Digital Filter block in the Filter Designs library replaces the following blocks from DSP Blockset 4.1:

- Biquadratic Filter
- Time-Varying Lattice Filter
- Time-Varying Direct-Form II Transpose Filter

Your models that contain these replaced blocks will still work, and you can still access these blocks by typing `dsparch3` at the MATLAB command line.

However, when creating new models, use the new Digital Filter block, which can implement various filters including those that the above three blocks implement: biquadratic direct from II transposed filters, time-varying lattice filters, and time-varying direct form II transposed filters.

See "Digital Filter" on page 1-6 for more information about the new Digital Filter block.

Wavelet Analysis and Wavelet Synthesis Blocks Replaced

The Wavelet Analysis and Wavelet Synthesis blocks, formerly in the Multirate Filters library, are now replaced by new or enhanced blocks as summarized in the following table.

New or Enhanced Block	**Replaces...**
Dyadic Analysis Filter Bank	Wavelet Analysis block in frame-based operation
Dyadic Synthesis Filter Bank	Wavelet Synthesis block in frame-based operation
Two-Channel Analysis Subband Filter	Wavelet Analysis block in sample-based or frame-based operation
Two-Channel Synthesis Subband Filter	Wavelet Analysis block in sample-based or frame-based operation

You can still access the Wavelet Analysis and Wavelet Synthesis blocks by typing `dspmlti3` at the command line.

For more information about the new and enhanced replacement blocks, see the following topics:

- "Two-Channel Analysis Subband Filter and Two-Channel Synthesis Subband Filter" on page 1-8
- "Dyadic Analysis Filter Bank and Dyadic Synthesis Filter Bank Enhancements" on page 1-11

Cumulative Sum Block Behaves Differently

The Cumulative Sum block now behaves differently when given frame-based inputs and set to sum along columns. In this situation, the block now computes the running sum of each column of a frame-based input, where the running sum is independent of the running sums of previous inputs.

To get the previous behavior for frame-based inputs when set to sum along columns, set the block to sum along channels.

Contiguous Copy Block Obsolete

The Contiguous Copy block, formerly in the Signal Attributes library, is now obsolete. Your models that currently use the block will still work, but you can no longer access the block in the DSP Blockset 5.0 libraries.

Audio Blocks Relocated

The following blocks, formerly in the DSP Sources and DSP Sinks libraries, are now in the new Windows (WIN32) library:

- From Wave Device
- From Wave File
- To Wave Device
- To Wave File

For more information, see "Audio Blocks Relocated to New Block Library" on page 1-15.

Platform Limitations

Limitations for HP and IBM

On the HP and IBM platforms, the DSP Blockset 5.0 has the same simulation limitations as Simulink and the same code generation limitations as Real-Time Workshop.

Limitation for Windows

On Windows platforms, the DSP Blockset 5.0 does not support Borland C++ 5.6. The DSP Blockset still supports Borland C++ 5.5.

Known Software Problems

This section includes a link to a description of known software and documentation problems in Version 5.0.

If you are viewing these Release Notes in PDF form, please refer to the HTML form of the Release Notes, using either the Help browser or the MathWorks Web site and use the link provided.

For a list of bugs reported in the previous release that remain open, see “Known Software Problems” on page 2-13 of the Release Notes for DSP Blockset 4.1.

DSP Blockset 4.1 Release Notes

New Features

This section introduces the new features and enhancements added in the DSP Blockset 4.1 (Release 12.1) since DSP Blockset 4.0 (Release 12.0).

If you are upgrading from a release earlier than Release 12.0, then you should see “New Features” on page 3-2.

The DSP Blockset 4.1 new features are discussed in detail in the following sections:

- “New Digital Filter Design Block” on page 2-2
- “Major Individual Block Enhancements” on page 2-2
- “Blocks with Enhanced Data Type Support” on page 2-5
- “Blocks with Tunability Enhancements” on page 2-6
- “Enhancements to Blocks Implemented with S-Functions” on page 2-7
- “Real-Time Workshop Generated C Code Enhancements” on page 2-8
- “Changes to Filtering Block Libraries” on page 2-9

New Digital Filter Design Block

The new Digital Filter Design block integrates the Filter Design and Analysis Tool graphical user interface (FDATool) with the DSP Blockset. Double-clicking the block opens FDATool, which you can use to quickly design, import, or directly specify a digital FIR or IIR filter. Using the specified filter, the block filters the input signal and outputs the result.

The Digital Filter Design block replaces many of the blocks from DSP Blockset 4.0 (Release 12). For details, see “Changes to Filtering Block Libraries” on page 2-9.

To implement a filter you have already designed, use the Digital Filter block, introduced in DSP Blockset 5.0 (Release 13).

Major Individual Block Enhancements

The following sections summarize the major individual block enhancements for DSP Blockset 4.1:

- “FFT Block”
- “IFFT Block”

- “Magnitude FFT Block”
- “Overlap-Add FFT Filter and Overlap-Save FFT Filter Blocks”
- “Random Source Block”
- “Sine Wave Block”
- “Unwrap Block”
- “Window Function Block”

FFT Block

- You can now set the block to compute twiddle factors by table lookup or by calling trigonometric math library functions. In table lookup mode, the block precomputes and stores a table that you can optimize for memory or speed. For an N-point FFT, a table optimized for speed yields a length-3N/4 table, and a table optimized for memory yields a length-N/4 table.
- A new option lets you set the block to output in bit-reversed or linear order. Outputting in bit-reversed order reduces computation time by eliminating the output data descrambling step.
- The block processes real-valued inputs using more efficient algorithms.
- The block generates smaller Real-Time Workshop C code with improved run-time efficiency through use of in-place algorithms.
- The block now supports single-precision (32-bit) floating-point input and output signals (in table lookup mode) for both simulation and Real-Time Workshop C code generation.

For more information, see the FFT block reference page.

IFFT Block

- You can now set the block's twiddle factor computation method as in the FFT block (see the earlier FFT block enhancement description).
- A new option lets you provide inputs in both linear and bit-reversed order. Using inputs in bit-reversed order reduces block computation time by eliminating the initial input data scrambling step.
- You can now explicitly specify conjugate-symmetric input signals to significantly increase processing efficiency and to get a real-valued output signal.

- You can now skip normalization by the IFFT length, N, when you do not need to output a normalized signal, increasing processing efficiency.
- The block generates smaller Real-Time Workshop C code with improved run-time efficiency through use of in-place algorithms.
- The block now supports single-precision (32-bit) floating-point input and output signals (in table lookup mode) for both simulation and Real-Time Workshop C code generation.

For more information, see the IFFT block reference page.

Magnitude FFT Block

- You can now set the block to compute the magnitude FFT or magnitude squared FFT
- The block uses the new, more efficient FFT block algorithms.

For more information, see the Magnitude FFT block reference page.

Overlap-Add FFT Filter and Overlap-Save FFT Filter Blocks

By using the enhanced FFT and IFFT blocks, the Overlap-Add FFT Filter and Overlap-Save FFT Filter blocks are more efficient. (Both blocks now skip data descrambling and scrambling operations by processing data in bit-reversed order, and use specialized, more efficient algorithms for real-valued input signals.)

Random Source Block

- You can now set the repeatability of the block output to **Not repeatable**, **Repeatable**, or **Specify seed**. The default mode is **Not repeatable**, so that when you add new Random Source blocks from the block library to a model, the blocks automatically generate different outputs (without requiring you to specify a seed).
- The block now uses Simulink run-time parameters, greatly enhancing its Real-Time Workshop generated C code (see "Enhancements to Blocks Implemented with S-Functions" on page 2-7).

Sine Wave Block

You can now optimize the Sine Wave block's table lookup computation method for speed or memory. For speed optimization, the block uses a full-length table, while for memory optimization, it uses a 1/4-length table.

Unwrap Block

You can now set the Unwrap block to phase unwrap across successive inputs for both frame-based and sample-based input signals.

Window Function Block

- The Window Function block now generates smaller, more efficient Real-Time Workshop C code by using run-time parameters (see "Enhancements to Blocks Implemented with S-Functions" on page 2-7).
- The block now supports single-precision (32-bit) floating-point input and output signals.

Blocks with Enhanced Data Type Support

Many blocks now support all Simulink built-in data types, as well as the fixed-point data type and properly defined Simulink custom data types in both simulation and in Real-Time Workshop C code generation. To see which data types a particular block supports, check the "Supported Data Types" section of its reference page.

The data signal input and output ports (noncontrol input and output ports) of the following blocks now support all built-in, fixed-point, and other properly defined Simulink custom data types:

- Buffer
- Check Signal Attributes
- Contiguous Copy
- Convert 1-D to 2-D
- Convert 2-D to 1-D
- Delay Line
- Display
- Downsample
- Extract Diagonal
- Extract Triangular Matrix
- Flip
- Frame Status Conversion
- Inherit Complexity
- Integer Delay
- Matrix Viewer
- Multiport Selector
- Pad
- Permute Matrix (also, `index` input port accepts signed and unsigned integer and double- and single-precision floating-point)
- Repeat
- Sample and Hold
- Signal To Workspace
- Spectrum Scope
- Submatrix
- Time Scope
- Toeplitz
- Transpose
- Triggered Delay line
- Triggered To Workspace
- Unbuffer
- Upsample
- Variable Integer Delay
- Variable Selector (also, `index` input port accepts signed and unsigned integer and double- and single-precision floating-point)
- Vector Scope
- Zero Pad

The data signal input and output ports (noncontrol input and output ports) of the following blocks now support all Simulink built-in data types:

- Constant Diagonal Matrix
- DSP Constant
- Signal From Workspace
- Triggered Signal From Workspace

Blocks with Tunability Enhancements

The following blocks have enhanced tunability:

- Chirp — **Target time** parameter is now tunable.
- Counter — **Maximum count** parameter is now tunable.
- Mean — **Reset port** parameter is now tunable.
- Pad — **Value** parameter is now tunable.

Enhancements to Blocks Implemented with S-Functions

Many of the DSP Blockset blocks are S-function blocks (implemented with Simulink S-functions). The following sections summarize some of the major DSP Blockset 4.1 enhancements that improve S-function block performance and help you to write better S-functions.

Use of Simulink Run-Time Parameters

Many S-function blocks now generate better optimized Real-Time Workshop C code by using Simulink run-time parameters (RTPs) for both mask dialog and internal parameters. Blocks that use RTPs include those with filter coefficient parameters and initial condition parameters.

Better Memory Management for Simulation and Code Generation

Memory management enhancements for S-function block simulation and code generation now provide simulation code requiring less memory, and smaller Real-Time Workshop generated C code that is more efficient at run-time. Memory management enhancements include the use of contiguous input and output arrays, reusable inputs, overwritable arrays, and in-place algorithms.

New S-Function Simulation Support Directories

Various extensible C-callable library functions, macros, and definitions are now included in the following directories:

- `toolbox/dspblks/include` — header files (prototypes and definitions)
- `toolbox/dspblks/src/sim` — simulation library source code
- `toolbox/dspblks/lib` — prebuilt library (`.lib`) files for certain combinations of platforms and compilers only

Real-Time Workshop Generated C Code Enhancements

DSP Blockset 4.1 enhancements that provide better Real-Time Workshop generated C code include the following:

- DSP Blockset can now build and link C-callable run-time libraries, providing for smaller, more efficient Real-Time Workshop generated C code.
- The use of run-time parameters in blocks implemented with S-functions (see "Enhancements to Blocks Implemented with S-Functions" on page 2-7) provides for the following Real-Time Workshop generated C code enhancements for blocks implemented using S-functions:
  - Parameter values are organized in a common structure.
  - Parameter values are located in a common header file (for example, `<model>_prm.h`).
  - Redundant copies of parameter values are reused or "pooled" in `<model>_prm.h` (when there are multiple instances of a block with the same run-time parameter values).

Changes to Filtering Block Libraries

The new **Filter Designs** block library in DSP Blockset 4.1 replaces the old **Filter Structures** and **Filter Designs** block libraries in DSP Blockset 4.0, and contains the following blocks:

- Analog Filter Design
- Biquadratic Filter
- Digital Filter Design
- Direct-Form II Transpose Filter
- Filter Realization Wizard
- Overlap-Add FFT Filter
- Overlap-Save FFT Filter
- Time-Varying Lattice Filter
- Time-Varying Direct-Form II Transpose Filter

The new Digital Filter Design block in the **Filter Designs** block library (see "New Digital Filter Design Block" on page 2-2) replaces the following DSP Blockset 4.0 blocks:

- Digital FIR Filter Design
- Digital FIR Raised Cosine Filter Design
- Digital IIR Filter Design
- Least Squares FIR Filter Design
- Remez FIR Filter Design

Your models that contain these replaced blocks will still work, but they are no longer accessible in the Filter Designs library. You can still access these blocks by typing `dspddes3` at the MATLAB command line, but you should use the Digital Filter Design block to design and implement filters in your new models.

Note The Digital Filter Design block allows you to design a filter, and then implement it. To implement a filter you already designed, use the Digital Filter block, introduced in DSP Blockset 5.0 (Release 13).

Major Bug Fixes

The following sections describe some of the major DSP Blockset 4.1 bug fixes:

- “Real-Time Workshop C Code Generated from DSP Blockset”
- “Nontunable Block Parameters”
- “Chirp Block”
- “Matrix Viewer Block”
- “Random Source Block”
- “Stack and Queue Blocks”
- “Window Function Block”

Real-Time Workshop C Code Generated from DSP Blockset

- Real-Time Workshop C code generated from the blockset now uses the `sizeof` function rather than hard-coding word sizes for nonstandard code generation targets (such as those that do not treat characters as 8-bit bytes).
- The generated C code for most blocks no longer generates compiler warnings for most supported targets.

Nontunable Block Parameters

All nontunable block parameters are now uneditable in the Simulink mask UI during simulations.

Chirp Block

The Chirp block now has correct behavior in bidirectional sweep mode, and correctly interprets target frequency and sweep time parameters.

Matrix Viewer Block

The Matrix Viewer block now displays the color bar.

Random Source Block

The Random Source block seed initialization better supports multiple copies of the block in one Simulink model.

Also, the block no longer sets the seed of the MATLAB random number generator function during block initialization.

Stack and Queue Blocks

The Stack and Queue blocks now correctly handle simultaneously triggered control ports (such as the `Push` and `Pop` ports).

Window Function Block

The Window Function block now errors out for non-floating-point inputs to avoid possible segmentation violations.

Upgrading from an Earlier Release

There are no issues for upgrading from DSP Blockset 4.0 (Release 12) to DSP Blockset 4.1 (Release 12.1).

For information about upgrading from an earlier version than DSP Blockset 4.0, see “Upgrading from an Earlier Release” on page 3-12.

Known Software Problems

The following sections describe some of the known DSP Blockset problems and limitations:

- "Real-Time Workshop C Code Generation Limitations"
- "Nonzero Simulation Start Times Not Supported"
- "Triggered and Enabled Subsystem Support"
- "Digital Filter Design Block Limitations"
- "Magnitude FFT Block Limitations"

Real-Time Workshop C Code Generation Limitations

C code generated from DSP Blockset blocks using Real-Time Workshop has the following limitations:

- DSP Blockset generally does not support the **Loop rolling threshold** parameter (visible when the **Category** parameter is set to **General code generation options**) in the **Real-Time Workshop** tab of the Simulink **Simulation Parameter**s dialog.
- Currently, the DSP Blockset run-time library `dsp_rt.lib` does not always rebuild when you change compiler or linker options. Thus, before changing compiler and linker options, you should delete `dsp_rt.lib` and all associated object files. You can change compiler and linker options in the Simulink **Simulation Parameters** dialog's **Real-Time Workshop** tab by setting `make` command arguments such as `OPTS` or `OPT_OPTS` in the **Make command** parameter (for example, `make_rtw OPTS="-DMYDEFINE=1"`).

Nonzero Simulation Start Times Not Supported

The DSP Blockset does not support nonzero simulation start times. When running a simulation with DSP Blockset blocks, you must set the **Start time** option in the Simulink **Simulation Parameters** dialog's **Solver** tab to `0`.

Triggered and Enabled Subsystem Support

Triggered or enabled subsystems currently do not support the following blocks:

- Buffer
- Counter
- Downsample
- Dyadic Analysis Filter Bank
- Dyadic Synthesis Filter Bank
- FIR Decimation
- FIR Interpolation
- Histogram
- Kalman Adaptive Filter
- LMS Adaptive Filter
- LU Solver
- Maximum
- Mean
- Minimum
- N-Sample Enable
- N-Sample Switch
- Overlap-Add FFT Filter
- Overlap-Save FFT Filter
- Permute Matrix
- Queue
- Repeat
- RLS Adaptive Filter
- RMS
- Stack
- Standard Deviation
- Time-Varying Direct-Form II Transpose Filter
- Time-Varying Lattice Filter
- Unbuffer
- Upsample
- Variable Fractional Delay
- Variable Integer Delay
- Variable Selector
- Variance
- Wavelet Analysis
- Wavelet Synthesis

Digital Filter Design Block Limitations

The Digital Filter Design block has the following limitations:

- When you install the Filter Design Toolbox, FDATool, opened from the MATLAB command line, gets a new tab, **Set Quantization Parameters**. However, the Digital Filter Design block's FDATool does not currently support the **Set Quantization Parameters** tab.
- The Digital Filter Design block currently allows you to edit nontunable FDATool filter specification parameters. When you edit nontunable

parameters during a simulation, the filter still updates in FDATool to reflect your changes. However, during simulation, FDATool only updates the block icon (which displays the frequency response of the updated filter). FDATool sends the new filter coefficients to the block only after the simulation stops running.

For details about the Digital Filter Design block, see the previous section, “New Digital Filter Design Block” on page 2-2.

Magnitude FFT Block Limitations

The Magnitude FFT block currently does not support 2-D sample-based 1-by-N row vector inputs.

DSP Blockset 4.0 Release Notes

New Features

This section introduces the new features and enhancements added in the DSP Blockset 4.0 since the DSP Blockset 3.1 (Release 11.0).

If you are upgrading from a release earlier than Release 11.0, then you should see "DSP Blockset Releases Prior to Version 4.0" on page 4-2.

Version 4.0 of the DSP Blockset offers complete support for the Simulink 4.0 enhancements that provide seamless propagation and processing of both matrix and frame-based signals. A number of new blocks are also provided.

This section is organized into the following subsections:

- "Matrix Support" on page 3-2
- "Frame Support" on page 3-4
- "New and Enhanced Blocks" on page 3-5
- "Library Reorganization" on page 3-7

Matrix Support

Matrix support is now provided natively within Simulink, which allows the DSP Blockset to generate and process matrix signals in a much more natural manner than was previously possible.

Generating Matrix Signals

Matrix signals can now be generated in the same way as vector or scalar signals, using appropriate source blocks such as Signal From Workspace or DSP Constant. Where you would have previously specified a vector or scalar, you can now use standard MATLAB notation to specify a matrix. For example, enter `[1 2 3;4 5 6]` in the **Constant value** parameter of the DSP Constant block to generate the 2-by-3 output matrix.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

When **Signal dimensions** is selected from the model window **Format** menu, Simulink displays the dimensions of an M-by-N matrix signal as a pair of bracketed integers, `[MxN]`. Simulink *does not display* any size information for a scalar signal. Dimension information for a signal can also be displayed in a model by attaching a Probe block with **Probe signal dimensions** selected.

Row vectors and column vectors are matrices with M=1 or N=1, respectively. These are generated by the DSP Constant block when a vector is specified in the **Constant value** parameter and the **Interpret vector parameters as 1-D** check box is *not* selected.

In addition, Simulink recognizes one-dimensional (1-D) vectors, which are ordered lists of values. These are generated by the DSP Constant block when a vector is specified in the **Constant value** parameter and the **Interpret vector parameters as 1-D** check box is selected. When **Signal dimensions** is selected from the model window **Format** menu, Simulink displays the size of a 1-D vector signal as an unbracketed integer. For convenience, DSP Blockset blocks treat 1-D vectors as column vectors for most operations.

Processing Matrix Signals

The most noticeable benefit of the new matrix enhancements is that you no longer need to manually specify the size of matrix inputs to a block. All DSP Blockset blocks now automatically detect input dimensions, and will generate an error if those dimensions are not appropriate for the specific operation.

Since input dimensions are detected automatically by new (Version 4.0) blocks, a matrix output of an older block may need to be altered before you use is as an input to a new block, and vice versa. See "Adding Release 12 Blocks to Release 11 Models" on page 3-12 for more information.

The interpretation of sample-based and frame-based matrices is the same as in previous releases. An M-by-N sample-based matrix represents M*N independent channels, each containing one sample. An M-by-N frame-based matrix represents M consecutive samples from each of N independent channels. That is, each column of a frame-based matrix represents M consecutive samples from a single channel.

Frame Support

Support for frame-based signals is also now provided natively within Simulink, which allows for straightforward propagation of frame-based data throughout a model.

Generating Frame-Based Signals

Frame-based signals are generated in much the same way that they were in previous DSP Blockset releases. In constant blocks such as DSP Constant or Constant Diagonal Matrix, the **Frame-based output** check box confers frame-based status when selected. In other source blocks, such as Signal From Workspace, Chirp, and Random Source, the output is frame based whenever the **Samples per frame** setting is greater than 1.

Note that 1-D vectors cannot be frame based. See the DSP Constant block reference for information about the interaction of the **Frame-based output** and **Interpret vector parameters as 1-D** parameters.

Processing Frame-Based Signals

Frame-based status is now propagated automatically throughout a model. Most blocks do not change the frame status of a signal; frame-based signals remain frame based, and sample-based signals remain sample based. The exceptions to this rule are blocks, like Buffer and Frame Status Conversion, that are specifically intended to alter the frame status of a signal.

Since frame status is now automatically detected by the new (Version 4.0) blocks, a frame-based output from an older block may need to be altered before you use it as an input to a new block, and vice versa. See “Adding Release 12 Blocks to Release 11 Models” on page 3-12 for more information.

As mentioned above, the interpretation of sample-based and frame-based matrices is the same as in previous releases. An M-by-N sample-based matrix represents M*N independent channels, each containing one sample. An M-by-N frame-based matrix represents M consecutive samples from each of N independent channels. For convenience, certain vector-oriented blocks such as Autocorrelation treat sample-based 1-by-N matrices (i.e., row vectors) as N-by-1 matrices (i.e., column vectors). See the documentation for the particular block in which you are interested.

Simulink displays sample-based signals using a single line, $\rightarrow$, and frame-based signals using a double line, $\Rightarrow$. The sample period (T_S) of a sample-based signal, or the frame period (T_f) of a frame-based signal, can be displayed by connecting a Probe block to the line. As always, the sample period of a frame-based signal is T_f / M, where M is the signal's frame size (i.e., the number of rows in the frame-based M-by-N matrix).

New and Enhanced Blocks

Table 3-1 lists the blocks that are new in Version 4.0, as well as those (*) that received substantial enhancements above and beyond the general upgrade for matrix and frame support. Library levels are separated by a forward slash.

Table 3-1: New and Enhanced Blocks in the DSP Blockset 4.0

Block Library	Block Name	Description
DSP Sources	Chirp*	Generate a swept-frequency cosine. Version 4.0 adds bidirectional sweep.
	Constant Ramp	Generate a ramp signal with length based on input dimensions.
	Window Function*	Compute a window, and/or apply a window to an input signal. Version 4.0 can compute user-defined windows.
DSP Sinks	Signal To Workspace	Replaces To Workspace.
	Spectrum Scope	Replaces FFT Frame Scope and Buffered FFT Frame Scope.
	Vector Scope	Replaces Time Frame Scope, User-Defined Frame Scope, and Frequency Frame Scope.
Estimation / Linear Prediction	Autocorrelation LPC	Determine the coefficients of an Nth-order forward linear predictor.
Math Functions / Matrices and Linear Algebra / Linear System Solvers	SVD Solver	Solve the equation AX = B using singular value decomposition.
Math Functions / Matrices and Linear Algebra / Matrix Factorizations	Singular Value Decomposition	Compute the singular value decomposition of a matrix.
Math Functions / Matrices and Linear Algebra / Matrix Inverses	Cholesky Inverse	Compute the inverse of a Hermitian positive definite matrix using Cholesky factorization.
	LDL Inverse	Compute the inverse of a Hermitian positive definite matrix using LDL factorization.
	LU Inverse	Compute the inverse of a square matrix using LU factorization.
	Pseudoinverse	Compute the Moore-Penrose pseudoinverse.
Math Functions / Matrices and Linear Algebra / Matrix Operations	Identity Matrix	Generate a matrix with ones on the main diagonal and zeros elsewhere.
	Submatrix*	Select a subset of elements (submatrix) from a matrix input. Version 4.0 offers a major redesign of the block's interface.

Table 3-1: New and Enhanced Blocks in the DSP Blockset 4.0 (Continued)

Block Library	Block Name	Description
Math Functions / Polynomial Functions	Least Squares Polynomial Fit	Compute the coefficients of the polynomial that best fits the input data in a least-squares sense.
	Polynomial Stability Test	Determine whether all roots of the input polynomial are inside the unit circle using the Schur-Cohn algorithm.
Signal Management / Buffers	Buffer*	Rebuffer the input sequence to a smaller or larger frame size. In Version 4.0, the Buffer block incorporates all the capabilities of the former Rebuffer block. See "Minor Enhancements" below for more information.
	Delay Line	Rebuffer a sequence of inputs with a one-sample shift. Previously called Shift Register.
	Triggered Delay Line	Rebuffer a sequence of inputs with a one-sample shift. Previously called Triggered Shift Register.
Signal Management / Indexing	Multiport Selector	Distribute rows or columns of an input to multiple output ports.
Signal Management / Signal Attributes	Check Signal Attributes	Generate an error if input attributes differ from (or match) those specified.
	Convert 1-D to 2-D	Reshape a 1-D or 2-D input to a 2-D matrix with the specified dimensions.
	Convert 2-D to 1-D	Convert a 2-D matrix input to a 1-D vector.
	Frame Status Conversion	Specify the frame status of the output, sample-based or frame-based.
Signal Operations	Pad	Alter the input size by padding or truncating rows and/or columns.
Simulink / Signals & Systems	Matrix Concatenation	Concatenate inputs horizontally or vertically.

Minor Enhancements

Rowwise Operations. A number of blocks were enhanced to operate along both the row and column dimensions of inputs. Among the improved blocks are Zero Pad, Flip, Difference, and Cumulative Sum. All these blocks now allow you to select **Rows** or **Columns** as the dimension along which to operate.

Buffers Library Reorganization. The Buffers library has undergone a minor reorganization. The Buffer block now incorporates all the capabilities of the former Rebuffer block, which has been removed. The former Shift Register and Triggered Shift Register blocks have also been renamed to Delay Line and Triggered Delay Line, respectively.

Library Reorganization

The DSP Blockset libraries have been reorganized for clarity and accessibility. The revised library hierarchy is shown below:

- DSP Sinks
- DSP Sources
- Estimation
  - Linear Prediction
  - Parametric Estimation
  - Power Spectrum Estimation
- Filtering
  - Adaptive Filters
  - Filter Designs
  - Filter Structures
  - Multirate Filters
- Math Functions
  - Math Operations
  - Matrices and Linear Algebra
    - Linear System Solvers
    - Matrix Factorizations
    - Matrix Inverses
    - Matrix Operations
  - Polynomial Functions
- Quantizers
- Signal Management
  - Buffers
  - Indexing
  - Signal Attributes
  - Switches and Counters
- Signal Operations
- Statistics
- Transforms

Use the Simulink Library Browser to access the blockset directly through this hierarchical library list.

As a result of the reorganization, a number of existing blocks have been moved to new locations. Table 3-2 below lists the changes.

Table 3-2: New Block Locations in Version 4.0

Old Block Name	Old Library Location	New Block Name	New Library Location
Analog Filter Design	Filtering / Filter Designs	same	Filtering / Filter Structures
Analytic Signal	General DSP / Signal Operations	same	Transforms
Autocorrelation	Math Functions / Vector Functions	same	Statistics
Backward Substitution	Math Functions / Linear Algebra	same	Math Functions / Matrices and Linear Algebra / Linear System Solvers
Biquadratic Filter	Filtering / Filter Realizations	same	Filtering / Filter Structures
Buffered FFT Frame Scope	DSP Sinks	Spectrum Scope	same
Cholesky Factorization	Math Functions / Linear Algebra	same	Math Functions / Matrices and Linear Algebra / Matrix Factorizations
Cholesky Solver	Math Functions / Linear Algebra	same	Math Functions / Matrices and Linear Algebra / Linear System Solvers
Commutator	General DSP / Switches and Counters	obsolete	obsolete
Complex Exponential	Math Functions / Elementary Functions	same	Math Functions / Math Operations
Constant Diagonal Matrix	DSP Sources, Math Functions / Matrix Functions	same	Math Functions / Matrices and Linear Algebra / Matrix Operations
Contiguous Copy	Math Functions / Elementary Functions	same	Signal Management / Signal Attributes
Convert Complex DSP To Simulink	Math Functions / Elementary Functions	obsolete	obsolete
Convert Complex Simulink To DSP	Math Functions / Elementary Functions	obsolete	obsolete
Convolution	Math Functions / Vector Functions	same	Signal Operations
Correlation	Math Functions / Vector Functions	same	Statistics
Create Diagonal Matrix	Math Functions / Matrix Functions	same	Math Functions / Matrices and Linear Algebra / Matrix Operations
Cumulative Sum	Math Functions / Vector Functions	same	Math Functions / Math Operations

Table 3-2: New Block Locations in Version 4.0 (Continued)

Old Block Name	Old Library Location	New Block Name	New Library Location
dB	Math Functions / Elementary Functions	dB Conversion	Math Functions / Math Operations
dB Gain	Math Functions / Elementary Functions	same	Math Functions / Math Operations
Detrend	General DSP / Signal Operations	same	Statistics
Difference	Math Functions / Vector Functions	same	Math Functions / Math Operations
Direct-Form II Transpose Filter	Filtering / Filter Realizations	same	Filtering / Filter Structures
Discrete Constant	DSP Sources	DSP Constant	same
Distributor	General DSP / Switches and Counters	obsolete	obsolete
Downsample	General DSP / Signal Operations	same	Signal Operations
Extract Diagonal	Math Functions / Matrix Functions	same	Math Functions / Matrices and Linear Algebra / Matrix Operations
Extract Triangular Matrix	Math Functions / Matrix Functions	same	Math Functions / Matrices and Linear Algebra / Matrix Operations
FFT Frame Scope	DSP Sinks	Spectrum Scope	same
Filter Realization Wizard	Filtering / Filter Realizations	same	Filtering / Filter Structures
Flip	Math Functions / Vector Functions	same	Signal Management / Indexing
Forward Substitution	Math Functions / Linear Algebra	same	Math Functions / Matrices and Linear Algebra / Linear System Solvers
Frequency Frame Scope	DSP Sinks	Vector Scope	same
Inherit Complexity	Math Functions / Elementary Functions	same	Signal Management / Signal Attributes
Integer Delay	General DSP / Signal Operations	same	Signal Operations
LDL Factorization	Math Functions / Linear Algebra	same	Math Functions / Matrices and Linear Algebra / Matrix Factorizations
LDL Solver	Math Functions / Linear Algebra	same	Math Functions / Matrices and Linear Algebra / Linear System Solvers
Levinson Solver	Math Functions / Linear Algebra	same	Math Functions / Matrices and Linear Algebra / Linear System Solvers
LPC	General DSP / Signal Operations	Autocorrelation LPC	Estimation / Linear Prediction
LU Factorization	Math Functions / Linear Algebra	same	Math Functions / Matrices and Linear Algebra / Matrix Factorizations

Table 3-2: New Block Locations in Version 4.0 (Continued)

Old Block Name	Old Library Location	New Block Name	New Library Location
LU Solver	Math Functions / Linear Algebra	same	Math Functions / Matrices and Linear Algebra / Linear System Solvers
Matrix 1-Norm	Math Functions / Matrix Functions	same	Math Functions / Matrices and Linear Algebra / Matrix Operations
Matrix Constant	DSP Sources, Math Functions / Matrix Functions	DSP Constant	DSP Sources
Matrix From Workspace	DSP Sources, Math Functions / Matrix Functions	From Workspace	DSP Sources
Matrix Multiplication	Math Functions / Matrix Functions	Matrix Multiply	Math Functions / Matrices and Linear Algebra / Matrix Operations
Matrix Product	Math Functions / Matrix Functions	same	Math Functions / Matrices and Linear Algebra / Matrix Operations
Matrix Scaling	Math Functions / Matrix Functions	same	Math Functions / Matrices and Linear Algebra / Matrix Operations
Matrix Square	Math Functions / Matrix Functions	same	Math Functions / Matrices and Linear Algebra / Matrix Operations
Matrix Sum	Math Functions / Matrix Functions	same	Math Functions / Matrices and Linear Algebra / Matrix Operations
Matrix To Workspace	DSP Sinks, Math Functions / Matrix Functions	To Workspace	DSP Sinks
Multiphase Clock	DSP Sources, General DSP / Switches and Counters	same	DSP Sources
Normalization	Math Functions / Vector Functions	same	Math Functions / Math Operations
Overlap-Add FFT Filter	Filtering / Filter Realizations	same	Filtering / Filter Structures
Overlap-Save FFT Filter	Filtering / Filter Realizations	same	Filtering / Filter Structures
Partial Unbuffer	General DSP / Buffers	Buffer / Unbuffer	same
Permute Matrix	Math Functions / Matrix Functions	same	Math Functions / Matrices and Linear Algebra / Matrix Operations
Polynomial Evaluation	Math Functions / Elementary Functions	same	Math Functions / Polynomial Functions
QR Factorization	Math Functions / Linear Algebra	same	Math Functions / Matrices and Linear Algebra / Matrix Factorizations
QR Solver	Math Functions / Linear Algebra	same	Math Functions / Matrices and Linear Algebra / Linear System Solvers
Rebuffer	General DSP / Buffers	Buffer/Unbuffer	same

Table 3-2: New Block Locations in Version 4.0 (Continued)

Old Block Name	Old Library Location	New Block Name	New Library Location
Reciprocal Condition	Math Functions / Linear Algebra	same	Math Functions / Matrices and Linear Algebra / Matrix Operations
Repeat	General DSP / Signal Operations	same	Signal Operations
Reshape	Math Functions / Matrix Functions	same	Simulink / Signals & Systems
Sample and Hold	General DSP / Switches and Counters	same	Signal Operations
Shift Register	General DSP / Buffers	Delay Line	same
Submatrix	Math Functions / Matrix Functions	same	Math Functions / Matrices and Linear Algebra / Matrix Operations
Time Frame Scope	DSP Sinks	Vector Scope	same
Time-Varying Direct-Form II Transpose Filter	Filtering / Filter Realizations	same	Filter Structures
Time-Varying Lattice Filter	Filtering / Filter Realizations	same	Filter Structures
Toeplitz	Math Functions / Matrix Functions	same	Math Functions / Matrices and Linear Algebra / Matrix Operations
Transpose	Math Functions / Matrix Functions	same	Math Functions / Matrices and Linear Algebra / Matrix Operations
Triggered Matrix To Workspace	DSP Sinks	Triggered To Workspace	same
Triggered Shift Register	General DSP / Buffers	Triggered Delay Line	same
Unwrap	Math Functions / Vector Functions	same	Signal Operations
Upsample	General DSP / Signal Operations	same	Signal Operations
User-Defined Frame Scope	DSP Sinks	Vector Scope	same
Variable Fractional Delay	General DSP / Signal Operations	same	Signal Operations
Variable Integer Delay	General DSP / Signal Operations	same	Signal Operations
Variable Selector	Math Functions / Elementary Functions	same	Signal Management / Indexing
Window Function	DSP Sources, General DSP / Signal Operations	same	Signal Operations
Zero Pad	General DSP / Signal Operations	same	Signal Operations

Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the DSP Blockset 3.1 to the DSP Blockset 4.0.

Adding Release 12 Blocks to Release 11 Models

All DSP Blockset Version 4.0 blocks support the Simulink 4.0 matrix and frame enhancements. Thus, when you add new (Version 4.0) blocks to a model containing older blocks, some of the signals may need to be altered. Otherwise, an older block may not correctly recognize the output from a new block, and vice versa. See "Using Version 4.0 Blocks with Blocks from Earlier Versions" "Checking Block Versions" and "Finding Replacement Blocks" below for details.

Using Version 4.0 Blocks with Blocks from Earlier Versions

Input dimension and frame status are detected automatically by new (Version 4.0) blocks but not by older blocks from previous releases. Thus, when both new and older blocks are used in a model, some signals may need to be passed through conversion blocks so the dimension and frame status of signals are recognized by their receiving blocks.

Using Outputs from Older Blocks as Inputs to New Blocks. All outputs of blocks from previous releases are 1-D vectors. You can use an output of an older block as an input to a new block without altering the signal. In this case, the new block will usually treat the 1-D input vector as a sample-based 2-D column vector (check the documentation for the particular block in which you are interested). You can also change the 1-D output of an older block to a 2-D sample- or frame-based matrix before feeding it into a new block: simply insert the Convert 1-D to 2-D block in the DSP Blockset Signal Attributes library. See the documentation for the Convert 1-D to 2-D block for more information.

Using Outputs from New Blocks as Inputs to Older Blocks. Older blocks can only accept 1-D inputs. To use an output from a new block as an input to an older block, pass the new output through the Convert 2-D to 1-D block found in the DSP Blockset Signal Attributes library. You can set the frame status and dimensions of the resulting 1-D signal in the parameters dialog box of the older block. See the documentation for the Convert 2-D to 1-D block for more information.

Checking Block Versions

When you begin adding Version 4.0 blocks to an older model, use the `dsp_links` function to check the versions of all blocks in that model. When you type

```
dsp_links
```

at the command line, the blocks in current model are color coded to reflect the version of the DSP Blockset to which they are linked. By default, Version 4.0 blocks are displayed in green. If the color coding reveals that a Version 4.0 block is providing the input to a Version 3 or Version 2 block (displayed in yellow or red by default), the older block should be replaced with a current version. Alternatively, you can use conversion blocks as described previously in "Using Version 4.0 Blocks with Blocks from Earlier Versions" on page 3-12.

Finding Replacement Blocks

In most cases, you can simply replace the older block with the current block of the same name. In general, you should open the block's dialog box and verify that the settings are correct for the intended operation. (Many block dialogs have undergone minor refinements.)

In certain cases, a block may have changed name or location in Version 4.0. If you cannot find the block you are looking for, check Table 3-2 to see if it may have been renamed or relocated.

DSP Blockset Notes from Earlier Releases

DSP Blockset Releases Prior to Version 4.0

The following sections include information about DSP Blockset versions prior to DSP Blockset 4.0; this information formerly appeared in the `Readme.m` file that was distributed with the product in past releases:

- "DSP Blockset 3.0" on page 4-3
- "DSP Blockset 2.2" on page 4-9
- "DSP Blockset 2.0" on page 4-14

DSP Blockset 3.0

The following sections have the complete contents of the Readme.m file of DSP Blockset Version 2.2:

Infrastructure Changes

Multichannel frame-based processing

Blocks now support a frame-based processing mode which allows you to more faithfully model real-time DSP applications. Frame-based processing also provides significant performance gains, both during simulation and in the generated code (via Real Time Workshop)

All blocks which retain state information over time now provide optional multichannel frame-based processing, including all filters, delays, running statistics, etc.

New intrinsic complex data types

Complex data is now an intrinsic data type, and blocks and lines now automatically recognize when data is real or complex. This greatly reduces the risk of unintentionally sending complex data to a block which only accepts real inputs.

All blocks now accept both real and complex data at their input ports, including combinations of real and complex inputs.

The internal format for complex data has fundamentally changed, from a concatenation of the real followed by the imaginary part, to an interleaving of the real and imaginary parts. This leads to increased efficiency when processing complex data and generating code.

The DSP Blockset Complex library is now obsolete, and all basic complex blocks have moved to the Simulink library. For example, complex math may now be performed with the core Simulink math blocks.

Only source and sinks require sample time specifications

Enhancements to the Simulink S-Function API provides for automatic propagation of sample time information, *removing* the requirement for sample times to be entered in block dialogs.

Dynamic dialogs

Masks now support *dynamic dialogs*, where dialog controls may now change dynamically in response to user entries, simplifying block dialogs and reducing the chances for entry errors.

Upgrading Models

All old (v2.2 and prior) models are supported and will run in this release while you make the transition to Version 3.0. Note that there is no automatic model upgrade facility:

- A new utility (`dsp_links`) helps to locate and summarize usage of DSP library blocks used in models.
- You cannot directly mix the old complex data format with the new intrinsic complex data type; conversion blocks are provided in the Elementary Functions library, which can translate between the two complex formats while transitioning older models to Version 3.0.

DSP Default Settings

A script has been added to the DSP Blockset that changes the default Simulink model settings to values more appropriate for DSP systems.

Include a call to the `dspstartup.m` script in your own `startup.m` file in order to automatically install these defaults each time you run MATLAB.

New Blocks

The following libraries have new blocks:

- “Sources” on page 4-5
- “Sinks” on page 4-5
- “Elementary Functions” on page 4-5
- “Matrix Functions” on page 4-6
- “Linear Algebra” on page 4-6
- “Buffers” on page 4-6
- “Switches and Counters” on page 4-6
- “Parametric Estimation” on page 4-6
- “Power Spectrum Estimation” on page 4-7
- “Filter architectures” on page 4-7
- “Adaptive Filters” on page 4-7
- “Multirate Filters” on page 4-7

Sources

- Discrete Constant
- Triggered Signal From Workspace
- Sine Wave
- From Wave File (PC only)
- From Wave Device (PC only)

Sinks

- Time/Frequency Frame Scopes
- To Wave File (PC only)
- To Wave Device (PC only)

Elementary Functions

- Convert Complex: DSP to Simulink
- Convert Complex: Simulink to DSP
- Contiguous Copy

- Inherit Complexity
- Variable Selector

Matrix Functions

- Extract Triangular Matrix
- Extract Diagonal
- Create Diagonal Matrix
- Permute Matrix
- Matrix Scaling
- Matrix Sum (sum)
- Matrix Product (prod)

Linear Algebra

- QR, LU, LDL, Cholesky factorization
- QR, LU, LDL, Cholesky, triangular solvers
- Reciprocal Condition estimation

Buffers

- Rebuffer
- Stack
- Queue

Switches and Counters

- Multiphase clock
- General up/down counter
- Event count comparator
- Edge detector

Parametric Estimation

- Yule-Walker AR Estimator
- Burg AR Estimator, supporting both polynomial and reflection coefficient outputs
- Covariance AR Estimator

- Modified Covariance AR Estimator

Power Spectrum Estimation

- Short-Time FFT
- Magnitude FFT
- Yule-Walker Method
- Burg Method
- Covariance Method
- Modified Covariance Method

Filter architectures

- Biquadratic (second-order section) filter
- Time-varying direct-form II transpose filter
- Time-varying lattice filter

Adaptive Filters

Enabled versions of LMS, RLS, and Kalman adaptive filters

Multirate Filters

- Dyadic Analysis Filter Bank
- Dyadic Synthesis Filter Bank

Block Enhancements

- All blocks automatically accept real and complex data at their input ports, including combinations of real/complex inputs.
- All blocks which retain state information over time now provide optional multichannel frame-based processing, including:
  - all filters (all static, adaptive, and multirate blocks)
  - all delay blocks (both static and variable blocks)
  - statistics blocks (min/max/etc.)
- All frame-based blocks now support multichannel operation, including: FFT, Windows, Unbuffer, etc.

- Brand new Frame Scope blocks for displaying the results of frame-based simulations. These blocks support multiple data channels and are highly customizable.
- Removed input port width requirement from Window block
- Added optional value/index ports to min/max statistics blocks
- Unbuffering now provides general output size/overlap parameters.
- Filter Realization Wizard supports creation of filters which employ blocks from the Fixed-Point Blockset in all filter realizations.

DSP Blockset 2.2

The following sections have the complete contents of the Readme.m file of DSP Blockset Version 2.2:

Enhancements to Old Blocks

Changing Block Parameters During Simulation

Some blocks have parameters which cannot be changed during simulation. In past releases, changes to these parameters were simply ignored. Now, attempts to change these parameters will generate a warning.

FIR Rate Conversion

The FIR Rate conversion block has been re-implemented to improve computational performance. Additionally, it can now handle vectorized inputs.

Buffer and Unbuffer

All Buffer and Unbuffer blocks now handle vectorized inputs. Additionally, the Buffer blocks can now take initial conditions. Similar improvements have been made to the Commutator and Distributor blocks.

Delay

Now supports specification of initial conditions from the block dialog.

Diagonal Matrix

Now supports specification of a non-constant diagonal.

Zero Pad and Complex Zero Pad

Now supports a checkbox to disable truncation of input vector width

Vector Scope Blocks

All vector scope blocks (in DSP Sinks) provide an additional option to record the current scope position into the block dialog.

New Blocks

Many new blocks have been added in V2.2. Take some time to explore the block library, and the demonstration examples. In particular, the blockset now includes new blocks in the following libraries:

- "DSP Sinks" on page 4-10
- "General DSP" on page 4-11
- "Transforms" on page 4-11
- "Buffers" on page 4-11
- "Switches and Counters" on page 4-11
- "Math Functions" on page 4-11
- "Complex" on page 4-12
- "Matrix" on page 4-12
- "Statistics" on page 4-12
- "Filters" on page 4-12
- "Multirate Filters" on page 4-12
- "Spectral Analysis" on page 4-12

DSP Sinks

- Triggered To Workspace
- Triggered Complex To Workspace

- Triggered Matrix To Workspace
- Triggered Complex Matrix To Workspace

General DSP

- Signal Operations
- Levinson-Durbin
- Complex Levinson-Durbin
- Complex Delay
- Variable Integer Delay
- Variable Fractional Delay
- LPC
- Complex LPC

Transforms

- DCT
- Real DCT
- IDCT
- Real IDCT
- Complex Cepstrum
- Real Cepstrum

Buffers

- Shift Register
- Triggered Shift Register

Switches and Counters

- N-Sample Enable w/Reset
- Sample and Hold

Math Functions

- Vector Math
- Autocorrelation

- Complex Autocorrelation
- Normalization
- Complex Normalization

Complex

- Real To Complex
- Complex Gain

Matrix

- Complex Diagonal Matrix

Statistics

- Sort
- Median
- Histogram
- Running Histogram

Filters

- Filter Realizations
- Multichannel IIR Filter (Frame)
- Time Varying IIR Filter
- Time Varying FIR Filter
- Filter Realization Wizard

Multirate Filters

- FIR Rate Conversion (Frame)

Spectral Analysis

- Yule-Walker AR
- Burg Method

New Demos

- Audio Effects: Flanging

- Audio Effects: Reverberation
- Sigma-Delta A/D Conversion
- Spectral Analysis: Comparison of methods
- Waveform Coding: ADPCM
- Time-Code Receiver

New Tools

Filter Realization Wizard

This is a GUI which automatically constructs filter realizations based on a selected architecture and corresponding coefficients. The filters are generated using gain, sum, and delay blocks, and may be constructed in real or complex form. Architectures include Direct-Form I and II, Symmetric FIR, and AR/MA/ARMA Lattices.

Obsolete Functions

The following functions are obsolete. They are shipping with Version 2.2 solely for backwards compatibility with prior releases:

- `sbuffer` — Vector buffering is now implemented in `sbufferc`.
- `selmath1` — Elementary math blocks are now available in Simulink. See the dspelmat library for these blocks.
- `selmath2` — Same as `selmath1`.
- `slength` — Vector length is now implemented by the Simulink Width block.
- `srfft` — Real FFT is now implemented by `srfftc`.
- `ssink` — Unused output terminator is replaced by the Simulink terminator block.
- `unbuff` — Vector unbuffering is now implemented by `sunguffc`.
- `szerofil` — Zero fill (upsample) replaced by `szerofil2`.
- `joincon` — Complex "join" icon function.
- `mtrxicon` — Matrix icon function.

DSP Blockset 2.0

The following sections have the complete contents of the `Readme.m` file of DSP Blockset Version 2.0:

Block Libraries

The blockset now makes extensive use of Simulink Libraries, which provides a block copy-by-reference facility. Libraries may be distinguished from models by the name appearing in the window title bar, and the disabled Simulation menu. Masked blocks copied from any DSP Blockset library are copied by reference, so future changes made to the library implementation of the block will automatically occur wherever the block has been used. You may still modify an individual copy of a library block, but you must break the link to the library in order to do so. You may also construct your own block libraries. See the Using Simulink documentation for more information on Simulink libraries.

New Versions of Previous Blocks

Elementary Math Blocks

Many of the Elementary Math blocks appearing in DSP Blockset Version 1.0a have migrated into the Simulink/Nonlinear block library, yielding higher performance for both simulation and code generation.

The new blocks are

- Trigonometric Function block — `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `atan2`, `sinh`, `cosh`, `tanh`
- Math Function block — `exp`, `log`, `10^u`, `log10`, `square`, `sqrt`, `pow`, `reciprocal`, `hypot`, `rem`, `mod`
- Rounding Function block — `floor`, `ceiling`, `round`, `fix`

These blocks were previously implemented as C-MEX S-functions, and were implemented as one function per block. The three new Simulink blocks are implemented with pop-up menu function selection for better usability. Copies of these Simulink blocks may be found in the DSP Blockset Math Functions/ Elementary Math block library for easy access.

Note that you may continue to use the earlier versions of these blocks, but the older CMEX S-function implementations will continue to be employed. Such usage is now considered obsolete, and a warning will be generated. You may suppress the warnings by typing `warning off` from the MATLAB command line, or you may replace instances of the obsolete blocks with copies of the new blocks.

Length Block

The Length block from Version 1.0a has been migrated into the Simulink Connections library where it is now named the Width block. A copy of the Width block now appears in the Math Functions/Vector Math library, along with a Complex Width counterpart, for convenience.

Note that you may continue to use the earlier version of this blocks, but the older CMEX S-function implementation will continue to be employed. Such usage is now considered obsolete, and a warning will be generated. You may suppress the warnings by typing `warning off` from the MATLAB command line, or you may replace instances of the obsolete block with copies of the new block.

Buffer Block

Changes which will affect the behavior of simulations involving buffering and/ or unbuffering operations have taken place in Version 2.0. Essentially, these blocks have been modified such that their behaviors are simpler and more consistent with real-time code practices.

Assume a buffer block configured with buffer size N and overlap OV. The buffer block now outputs an initial buffer of length N at time 0 (i.e., the start of simulation), while acquiring the next (N-OV) number of new samples. After (N-OV) samples have been acquired, at time (N-OV+1), the latest buffer will be output, and the acquisition of new samples resumed.

Contrast this with the earlier buffer operation, in which the buffer did not output an initial buffer of data; the first output was made at time (N-OV).

Unbuffer Block

The unbuffer block now supports only the "pipeline" mode of execution in order to guarantee consistency between simulation and real-time code execution.

There are now two unbuffer blocks. One supports "complete" unbuffering only, in which the block unbuffers the entirety of the input vector. The second supports "partial" unbuffering, in which just a portion of the input vector may be specified on which to perform the unbuffering operation.

Enhancements to Old Blocks

Performance Enhancements

Many blocks and S-Functions have been re-implemented to improve their performance, both in terms of computational load and data memory requirements.

FIR Decimation and Interpolation

These blocks have been re-implemented using multirate polyphase filter architectures, coded as C MEX S-functions, yielding much higher performance.

New Blocks and Libraries

The blockset has numerous block additions in comparison to V1.0a. Take some time to explore the block library, and the demonstration examples. In particular, the blockset now includes

- Adaptive filters
- Multirate filters
- Multichannel IIR filters
- Matrix library

- Overlap-add and overlap-save filters
- Switches and timers

Additionally, numerous changes to library block interfaces have been made in order to provide a smaller and more usable set of blocks without any loss of functionality. For example, many of the filter design blocks now have pop-up menus for selection of filter characteristic (lowpass/highpass/etc.) and type (Butterworth/Elliptic/etc.), instead of separate blocks for each combination of parameters.

Real-Time Workshop (RTW)

The DSP Blockset no longer requires template makefiles in order to generate code with the Real-Time Workshop. In addition, loop rolling is now performed automatically for loop counts above a configurable threshold. See the Real-Time Workshop documentation for more information.

Other

Sample Times with Nonzero Offsets

Use of nonzero sample time offsets are discouraged in this and future releases of the DSP Blockset. In future releases, support for this feature will be phased out of the DSP Blockset, and any nonzero sample time offsets encountered will generate warnings and be ignored. Sample time offsets should be accomplished via sample rate conversions and delays.