Simulink Release Notes

The “Simulink 5.0 Release Notes” on page 1-1 describe the changes introduced in the latest version of Simulink. The following topics are discussed in these Release Notes:

- “New Features” on page 1-2
- “Major Bug Fixes” on page 1-10
- “Platform Limitations for HP and IBM” on page 1-11
- “Upgrading from an Earlier Release” on page 1-12
- “Known Software and Documentation Problems” on page 1-13

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

- “Simulink 4.1 Release Notes” on page 2-1
- “Simulink 4.0 Release Notes” on page 3-1

If you are upgrading from a release prior to Release 11.1, see the Release 11.1 New Features Guide. Note that this is a PDF document.

Printing the Release Notes

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

Contents

1 Simulink 5.0 Release Notes

2 Simulink 4.1 Release Notes

Simulink 5.0 Release Notes

New Features

Note Simulink 5.0 incorporates changes introduced in Simulink 4.1.1, which was initially released in Web-downloadable form after Release 12.1 was released, but before Release 13. These Release Notes describe those changes, as well as other changes introduced after Version 4.1.1.

Simulink 5.0 introduce features and enhancements in the following areas:

- "Block Enhancements" on page 1-2
- "Simulation Enhancements" on page 1-6
- "Modeling Enhancements" on page 1-7

If you are upgrading from a release earlier than Release 12.1, then you should also see "New Features" on page 2-2 in the Simulink 4.1 Release Notes.

Block Enhancements

Simulink 5.0 includes the following block-related enhancements:

- "Fixed-Point Block Library" on page 1-3
- "Look-Up Table Editor" on page 1-3
- "Model Verification Block Library" on page 1-4
- "Signal Builder Block" on page 1-4
- "DocBlock" on page 1-4
- "Rate Transition Block" on page 1-4
- "Block Library Reorganization" on page 1-4
- "Model Linearization Blocks" on page 1-4
- "Data Read/Write Block Navigation" on page 1-5
- "Enhanced S-Function Builder" on page 1-5
- "Miscellaneous Block Enhancements" on page 1-5

Fixed-Point Block Library

Simulink now includes the latest version (4.0) of the Fixed-Point Blockset. The library was previously available only as a separately installed option. You must have a Fixed-Point Blockset license to run models containing fixed-point blocks in fixed-point mode. However, you can open, edit, and run such models in floating-point mode, regardless of whether you have a Fixed-Point Blockset license. This change facilitates sharing of fixed-point models in large organizations by eliminating the need for all users in a group to have a Fixed-Point Blockset license in order to run or modifymodels containing fixed-point blocks. See "Installation and Licensing" in the Fixed-Point Blockset release notes for information on how to run models containing fixed-point blocks when you do not have a Fixed-Point Blockset license.

This release also unifies many core Simulink and Fixed-Point Blockset blocks that have similar functionality. For example, the Sum block in the Simulink Math Operations library and the Sum block in the Fixed-Point Blockset Math library are now the same block. As a result, you no longer have to replace any of the unified blocks when switching from built-in to fixed-point data types and vice versa. You can change the data types of the blocks simply by selecting the appropriate settings on their parameter dialog boxes. See "Unified Simulink and Fixed-Point Blockset Blocks" in the Fixed-Point Blockset release notes for more information and for a list of blocks that this release unifies.

Note When you open an existing model, Simulink 5.0 updates the model to use the unified version of a standard or Fixed-Point Blockset block wherever an instance of that block occurs in the model. Simulink sets the parameters of the unified block to preserve the behavior of the original block. For example, wherever your existing model contains a Sum block from the Fixed-Point Blockset library, Simulink replaces the Fixed-Point Blockset version with a unified Sum block set to operate as a fixed-point block. This automatic updating ensures that your existing model runs the same in Simulink 5.0 as it did in previous releases of Simulink.

Look-Up Table Editor

The Look-Up Table Editor allows you to find and edit the contents of look-up tables used by look-up table blocks. See "Look-Up Table Editor" in the online Simulink documentation for more information.

Model Verification Block Library

Simulink now includes a library of model verification blocks that enable you to create self-validating models. For example, you can use the blocks to test that signals do not exceed specified limits during simulation. When you are satisfied that a model is correct, you can turn error-checking off by disabling the model verification blocks. You do not have to physically remove them from the model. The library includes set of blocks preconfigured to check for common types of errors, for example, signals that exceed a specified upper or lower bound. See "Model Verification" in the online Simulink documentation for more information.

Signal Builder Block

The new Signal Builder block allows you to create interchangeable groups of signal sources and quickly switch the groups into and out of a model. The Signal Builder block's signal editor allows you to define the waveforms of the signals output by the block. You can specify any waveform that is piecewise linear. Signal groups can greatly facilitate testing a model, especially when used in conjunction with Simulink assertion blocks and the optional Model Coverage Tool. See "Working with Signal Groups" for more information.

DocBlock

The new DocBlock block allows you to create text that documents a model and save that text with the model.

Rate Transition Block

Simulink now includes a Rate Transition block that allows you to specify the data transfer mechanism between two rates of a multirate system. See Rate Transition in the online Simulink block reference for more information.

Block Library Reorganization

The Simulink Block Library has been reorganized to simplify accessing blocks with related functionality. See for more information.

Model Linearization Blocks

This release introduces two blocks that generate linear models from a Simulink model at various times during a simulation. The Time-Based Linearization block generates linear models at specified time steps. The Trigger-Based

Linearization block generates models when triggered by events appearing at its trigger port.

Data Read/Write Block Navigation

To come

Enhanced S-Function Builder

The S-Function Builder has been enhanced to generate S-functions with the following additional capabilities

- Multiple ports
- Support for all builtin datatypes
- Support for 2-D signals
- Support for complex signals

See “Building S-Functions Automatically” for more information.

Miscellaneous Block Enhancements

This release introduces the following enhancements to Simulink blocks.

Math Function Block. This release significantly speeds up the simuluation of the Math Function block’s exponential math functions. All functions now support both double- and single-precision floating-point inputs and outputs. The `mod` and `rem` functions also support inputs and outputs of all integer types. The `transpose` and `hermitian` functions support all data types. When optimizations are enabled, the conjugate operation on a real signal invokes the block reduction optimization, as that case is a no-op. In-place multiplies for the magnitude^2 operation are used for reused block I/O on real signals.

Gain Block . The Gain block now performs block reduction when block reduction is on, `inline parameters=ON`, and the gain is both nontunable and unity.

Width Block . The Width block now includes a parameter to specify the datatype of the output.

Real Data Type Support. The following blocks now operate on both double precision and single precision floating point signals:

- Dot Product

- Trignometric
- Matrix Inversion

Block Data Type Table

To view a table that summarizes the data types supported by the blocks in the Simulink and Fixed-Point block libraries, execute the following command at the MATLAB command line:

```
showblockdatatypetable
```

Simulation Enhancements

Simulink 5.0 includes the following new features and enhancements to simulation of Simulink models.

- "Invalid Loop Highlighting" on page 1-6
- "Algebraic Loop Highlighting" on page 1-6
- "Conditional Input Branch Execution" on page 1-7
- "Reorganized Simulation Diagnostics" on page 1-7
- "Enhanced Diagnostic Viewer" on page 1-7

Invalid Loop Highlighting

Simulink now detects and highlights several kinds of invalid loops:

- Loops that create invalid function-call connections or an attempt to modify the input/output arguments of a function call
- Loops containing non-latched triggered subsystems
- Self-triggering subsystems
- Loops containing action subsystems in a cycle

This makes it is easier to identify and fix the loop. See "Avoiding Invalid Loops" for more information.

Algebraic Loop Highlighting

Simulink now optionally highlights algebraic loops when you update or simulate a model. See "Highlighting Algebraic Loops" for more information. The `ashow` debug command without any arguments now lists all of a model's algebraic loops in the MATLAB command window.

Conditional Input Branch Execution

This release introduces a new optimization called conditional input branch execution. Previously, when simulating models containing Switch or Multiport Switch blocks, Simulink executed all blocks required to compute all inputs to each switch at each time step. In this release, Simulink, by default, executes only the blocks required to compute the control input and the data input selected by the control input at each time step. Similarly, code generated from the model by Real-Time Workshop executes only the code needed to compute the control input and the selected data input. This optimization speeds simulation and execution of code generated from the model. See "Optimizations" for more information.

Reorganized Simulation Diagnostics

The **Diagnostics Pane** of the **Simulation Parameters** dialog box now groups diagnostics by functionality. This makes it easier to find and configure related diagnostics.

Enhanced Diagnostic Viewer

This release introduces an enhanced Diagnostic Viewer. Improvements include

- Identical appearance on UNIX and Windows
- Hyperlinks to Simulink, Stateflow, and Real-Time Workshop objects that caused the errors displayed in the viewer
- Sortable error list

  Clicking a column head sorts the error list by the contents of that column.
- Configurable content

  The **View** menu allows you to choose which information to display in the viewer.
- Selectable font size

  The **FontSize** menu allows you to choose the size of the font used to display error messages.

See "Simulation Diagnostic Viewer" for more information.

Modeling Enhancements

The following enhancements facilitate creation of Simulink models.

Enhanced Mask Editor

This release introduces changes to the Mask Editor designed to improve usability. Changes include

- Block parameter information moves from the **Initialization** pane to a new pane entitled **Parameters**.
- The **Parameters** pane allows you to specify a callback function to be called when the value of a parameter changes.
- The **Parameters** pane allows you to specify via check boxes whether a parameter is visible on the masked block’s dialog box and whether a parameter is tunable.
- The **Icon** pane provides a list of examples of all the types of drawing commands that can be used to draw the block’s icon.

See “Creating Masked Subsystems” in the online Simulink documentation for more information.

Including Symbols and Greek Letters in Block Diagrams

This release allows you to include symbols, Greek letters, and other formatting in annotations, masked subsystem port labels, and masked subsystem icon text. You do this by including TeX formatting commands in the annotation, port label, or icon text.

Production Hardware Characteristics

Production hardware characteristics is a new setting on the **Advanced** pane of the **Simulation parameters** dialog box. This setting, intended for use in modeling, simulating, and generating code for digital systems, allows you to specify the sizes of the data types supported by the system being modeled.

Simulink uses this information to automate the choice of data types for signals output by some blocks. See "The Advanced Pane" for more information.

True Color Support

This release allows you to use any color supported by your system as the foreground or background colors of a block diagram. See "Specifying Block Diagram Colors" in the online documentation for more information.

Print Details

This command generates an HTML report detailing the contents of the currently selected model (see "Generating a Model Report" in the online documentation for more information).

Boolean Logic Signals

In previous releases, the `Boolean logic signals` optimization was off by default for new models (see "Optimizations" in the online Simulink documentation for a description of this option). In the current release, the optimization is on by default for new models. This change does not affect existing models.

Model Discretizer

The Model Discretizer tool selectively replaces continuous Simulink blocks with discrete equivalents. Discretization is critical in digital controller design for dynamic systems and for hardware in the loop simulations. You can use this tool to prepare continuous models for use with the Real-Time Workshop Embedded Coder, which supports only discrete blocks. See "Model Discretizer" in the online documentation for more information.

Major Bug Fixes

Simulink 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 “Bug Fixes” on page 2-10.

Platform Limitations for HP and IBM

The following are platform limitations for Simulink 5.0 for the HP and IBM platforms that are new limitations, as of Version 5.0.

- New version of the Mask Editor
- New version of the Diagnostic Viewer
- The Model Discretizer

Note The Release 12 and 12.1 platform limitations for Simulink for the HP and IBM platforms still apply to Release 13. These are listed below.

The following Java-dependent Simulink features, introduced in Simulink 4.1, are *not* available on the HP and IBM platforms.

- Simulink Data Class Designer
- S-Function Builder
- Look-Up Table Editor

In addition, the following Simulink features are not supported on the HP and IBM platforms:

- Simulink Editor's **Find** dialog

  Use the `find_system` command instead.
- GUI interface to the Simulink Debugger

  Use the command-line interface instead.
- The **View Changes** dialog box for modified library links

  Instead, select the modified link and execute `ld=get_param(gcb,'LinkData')` to get a structure that lists the parameter differences between the library and local instance of the block. Edit this structure and execute `set_param(gcb,'LinkData',ld)` to apply the changes.
- **Parameter** dialog for the Configurable Subsystem block.

  Use the `set_param` command instead to set the block's parameters.
- Model Discretizer

Upgrading from an Earlier Release

This section describes an upgrade issue involved in moving from Simulink 4.1 to Version 5.0.

If you are upgrading from a version earlier than 4.1, then you should see "Upgrading from an Earlier Release" on page 2-12 in the Simulink 4.1 Release Notes.

BlockInstanceData Function Deprecated

S-functions should no longer call the `BlockInstanceData` function. All data used by a block should be declared using data type work vectors (e.g., `DWORK`).

Known Software and Documentation 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 2-15.

Simulink 4.1 Release Notes

New Features

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

For information about Simulink features that are incorporated from recent releases, see “New Features” on page 3-2 in the Simulink 4.0 Release Notes.

This section about new Simulink features is organized into the following subsections:

- “Simulink Editor” on page 2-2
- “Modeling Enhancements” on page 2-4
- “Simulink Debugger” on page 2-7
- “Block Library” on page 2-8

Simulink Editor

This section describes enhancements to the Simulink Editor.

Undo Move

In Simulink 4.1, the **Undo** command on the Simulink **Edit** menu restores blocks, annotations, lines, and nodes that have moved to their original locations (see “Undoing a Command” in Using Simulink).

Undo Subsystem Creation

In Simulink 4.1, the **Undo** command on the Simulink **Edit** menu restores blocks that have been grouped into a subsystem to their original level in the model (see “Undoing Subsystem Creation” in Using Simulink).

Autoconnecting Blocks

This version makes connecting blocks significantly easier. To connect a set of source blocks to a target block, simply select the source blocks, hold down the **Ctrl** key and left--click the target block. Simulink draws connecting lines between the source blocks and the destination block, neatly routing lines around intervening blocks. To connect a source block to a set of target blocks, select the target blocks, hold down the **Ctrl** key and left--click the source block. To connect two blocks, select the source block, and left-click the destination block while holding down the **Ctrl** key. Simulink connects as many ports on the two blocks as possible (see "Autoconnecting Blocks" in *Using Simulink*).

Autorouting Signal Lines

Simulink now routes signal lines around intervening blocks when you connect them either interactively (by dragging the connecting lines or using autoconnect) or programmatically via the `add_line` command's new `'autorouting'` option (see "Autorouting Option Added to add_line Command" on page 2-4).

Displaying Storage Class on Lines

This version adds an item to the **Format** menu, which toggles the display of (nonAuto) storage class on signal lines (see "RTW Storage Class" in *Using Simulink* for more information).

Save Models in Release 11 Format

This release can save post-Release 11 models in Release 11 format. Simulink 3 (Release 11) can load and run converted models that do not use any post-Release 11 features of Simulink. Simulink 3 can load converted models that use post-Release 11 features but may not be able to simulate the model correctly. Use the **Save as** option from the Simulink **File** menu or the following command to save a model in Release 11 format.

```
slsaveas(SYS)
```

See "Saving a Model in Simulink 3 (R11) format" in *Using Simulink* for more information.

Modeling Enhancements

This section describes enhancements to Simulink dynamic system modeling tools.

Autorouting Option Added to add_line Command

The `add_line` command now optionally routes lines around intervening blocks and annotations. For example, the following command autoroutes a connection between two blocks in the `vdp` model.

```
add_line('vdp','Product/1','Mu/1','autorouting','on')
```

The autorouting option is off by default. See `add_line` in Using Simulink for more information.

S-Function Builder

The S-Function Builder generates an S-function from specifications that you enter in a dialog box. It provides an easy way for you to incorporate existing code into a Simulink model.

add_param, delete_param

With this version, you can add custom parameters to your block diagrams.

```
    add_param('modelname','MyParameterName','value')
    delete_param('modelname','MyParameterName')
```

You can also use the model handle in place of the model name. See `add_param` and `delete_param` in Using Simulink for more information.

Connection Callbacks

With this version, you can use `set_param` to set callbacks on ports that are triggered by changes in the ports' connectivity. The callback function parameter is named `ConnectionCallback`. When the port's connectivity changes (addition/deletion of line connected to the port, connection of new block to the port, etc.), Simulink invokes the callback function with the port handle as its argument. See "Port Callback Parameters" in Using Simulink for more information.

Saving Block User Data in Model Files

This version adds a new block parameter, named `UserDataPersistent`, that is `off` by default. Setting this parameter `on`, e.g.,

```
set_param(block-name,'UserDataPersistent','on')
```

causes Simulink to include a block's user data (i.e., the value of the block's `UserData` parameter) in the model file when you save a model. Simulink encodes the user data as ASCII characters and saves the encoded data in a new section of the model file called `MatData`. This mechanism works with all forms of MATLAB data, including arrays, structures, objects, and Simulink data objects. See "Associating User Data with Blocks" in *Using Simulink* for more information.

Absolute Tolerance Enhancements

This version adds a dialog item for setting the absolute tolerance for each state in the State-Space block, the Transfer Fcn block, and the Zero-Pole block. With this enhancement, you can now specify the absolute tolerance for solving every continuous state in your model.

Block Reduction Enhancements

S-functions may now request that they be eliminated from the compiled model. To do this, call `ssSetBlockReduction(true)` inside the S-function. This is an advanced feature provided for customers writing S-functions who want to optimize the generated code produced for their S-function. Graphical connectivity is now remapped during block reduction, eliminating a source of error during reduction (e.g., a memory reference error used to occur if Simulink eliminated a block connected to a scope). Block reduction is now `on` by default, and a Simulink preference has been added for the option.

Boolean Logic Signals Preference

The Simulink Preferences dialog box now allows you to specify the use of Boolean logic signals by default. See "Setting Simulink Preferences" in *Using Simulink* for more information.

Subsystem Semantics Demos

Typing `sl_subsys_semantics` at the MATLAB prompt now displays a set of models that illustrate the semantics of various types of subsystem blocks. The demos include formal definitions of function-call subsystems.

Enhanced Engine Model Demos

The top and bottom dead center detection in the `engine` and `enginewc` demo models now use a reset integrator. In previous versions, the models used a triggered subsystem to detect angular position. This method resulted in inefficiencies and a slower, less accurate solution. In addition, self-triggering subsystems are now illegal in Simulink.

Setting Block Sorting Priority on Virtual Subsystems

In Simulink 4.0, it was an error to specify a priority on a virtual subsystem. In Simulink 4.1, you can specify priorities on virtual subsystems.

Using ~ in Filenames on UNIX

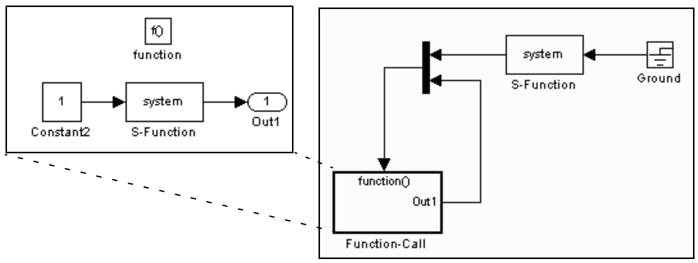
Now all filename fields in Simulink support the mapping of the ~ character in filenames. For example, in a To File block, you can specify `~/outdir/file.mat`. On most systems, this will expand to `/home/$USER/outdir/file.mat`.

Improved Warning About Slow Signals Feeding the Enable Port of an Enabled Subsystem Containing Fast Blocks

In a multitasking environment, deterministic results cannot be guaranteed if a slow signal feeds the enable port of an enabled subsystem that contains fast blocks. In previous versions, Simulink did not issue a warning in some cases where this may occur.

Flagging Function-Call Subsystem Cycles

In previous versions, Simulink allowed you to build models containing function-call-cycles, i.e., function-call subsystems that directly or indirectly call themselves.



Such models cannot be correctly simulated. Accordingly, Simulink now displays an error message when you attempt to run or update a diagram containing function-call cycles.

Simulink Debugger

This section describes enhancements to the Simulink debugger.

Enhancement to Sorted List Display

The Simulink debugger (`sldebug`) sorted list command, `slist`, now displays the names of the S-functions residing inside S-function blocks.

Improved Messages in Accelerated Mode

The `trace`, `break`, `zcbreak`, `nanbreak`, and `minor` commands now indicate that they are disabled when in accelerator mode and you need to switch to normal mode to activate them. The spacing of several messages has been fixed so the text aligns correctly.

Breakpoints on a Function-Call Subsystem

You can now put a break point on a function-call subsystem. Simulink breaks when the subsystem is executed. In Release 12, entering the `quit` command while at a breakpoint within a function-call subsystem wouldn't always quit the debugger. Now the `quit` command ends the debugging session once the initiating (calling) Stateflow chart or S-function finishes executing its time step.

Displaying and Probing Virtual Blocks

The `display` and `probe` commands now work for virtual blocks.

Stepping Stateflow Charts

You can now step execution of a model into a Stateflow chart.

Block Library

This section describes enhancements to the Simulink block libraries.

Unified Pulse Generator

This version merges the Discrete Pulse Generator block into the Pulse Generator block. The combined block has two modes: time-based and sample-based (discrete). Time-based mode varies the step size when a variable step solver is being used to ensure that simulation steps occur at pulse on/off transitions. When a fixed step solver is used, the time-based mode computes a fixed step size that ensures that a simulation step occurs at every pulse transition. The Pulse Generator block also outputs a pulse of any real data type in sample-based as well as time-based mode.

Control Flow Blocks

Simulink 4.1 adds an If block and Switch Case block that can drive conditionally executed subsystems that contain instances of the new Action Port block. Action subsystems are similar to enabled subsystems, except that all blocks must run at the same rate as the If or Switch Case block.

This version also adds a For Iterator block and a While Iterator block. When placed in a subsystem, these blocks cause all of the blocks in the system to run multiple cycles during a time step. The block cycle in a For Iterator subsystem runs a specified number of times. The block cycle in a While Iterator subsystem runs until a specified condition is false. A user can limit execution of a While Iterator subsystem to a specified number of iterations to avoid infinite loops.

The new Assignment block allows a model to assign values to specified elements of a signal.

Bus Creator

Simulink 4.1 adds a Bus Creator block that combines the output of multiple blocks into a single signal bus. A model can use the existing Signal Selector block to extract signals from the bus. The block's dialog box allows you to assign names to signals on the bus or allow the signals to inherit their names from their sources. When you double-click on a signal name in the block dialog, the source block is highlighted. There is no execution overhead in the use of bus creator/bus selector blocks.

Sine Wave Block Enhancements

The Sine Wave block now supports a bias factor that eliminates the need to sum with a Constant block. The Sine Wave block also has a new computational mode. This mode (called sample-based) eliminates the dependence on absolute time.

Enhanced Flip-Flop Blocks

Simulink Extras (`simulink_extras.mdl`) contains a set flip-flop blocks. These blocks now use the new triggered subsystem latching semantics. In addition, the S-R Flip-Flop block now models a physical `NOR` gate (i.e., `S=1`, `R=1 => Q=0`, `Q!=0`, the undefined state).

Additional Data Type Support

The Discrete-Time Integrator and Rounding Function blocks now handle `single` as well as `double` values. The Transport Delay, Unit Delay, Variable Transport Delay, Memory, Merge, and Outport blocks can specify nonzero initial conditions when operating on fixed-point signals.

Simulink Block Library Reorganization

The Simulink Block Library contains a new Subsystems sublibrary. The new library contains most of the new control flow blocks as well as subsystem and subsystem-related blocks that used to reside in the Signals & Systems library. The subsystems in the new library each contain the minimum set of blocks needed to create a functioning subsystem, e.g., an input port and an output port.

Scope Enhancements

The Scope block includes the following enhancements:

- A floating version of the Scope added to the Sinks block library
- Floating Scope saves the signals selected for display in the model file
- The Scope’s toolbar buttons for toggling between floating/nonfloating mode, restoring saved axes, locking/unlocking axes, and displaying the **Signal Selector**

Bug Fixes

This section lists fixes to bugs that occurred in the previous version of Simulink.

- Variable sample time S-functions

  Simulink no longer crashes when an S-function with variable sample time is placed in an atomic subsystem.
- Bus selector detection of duplicated names

  A bug related to the detection of a duplicated name in a bus that was feeding a Bus Selector block was fixed.
- Optimize block memory use

  In Simulink 4.0, the Continuous and Discrete Transfer Function blocks and the Discrete Filter block used more memory than they needed to, particularly for the case of many poles. They now use an optimal amount of memory.
- Miscellaneous fixes to the model loader

  Miscellaneous bug fixes have been performed on the model loader:
  - The loader and saver now retain any comment lines (i.e., lines that begin with #) that are found at the top of the model file.
  - The loader does not crash on Windows NT when file sizes are integer multiples of 4096.
  - The loader does not hang on corrupt models in which blocks with duplicate names are found.
- Profiler fixes

  The Simulink profiler now saves its files in the temporary directory. See the MATLAB command `tempdir`. The help was also updated.
- Chirp block fix

  The Chirp block now sweeps through frequencies correctly from the initial frequency at the simulation start time to the target frequency at the target time.
- Function-call subsystem bug fixes

  This version fixes several bugs related to the execution orders of function-call subsystems.

- Sorting bug fix

  Previous versions incorrectly computed the direct feedthrough setting for nonvirtual subsystems in triggered/function-call subsystems. This resulted in incorrect execution (sorting) orders. Now all nonvirtual subsystems within triggered subsystems have their direct feedthrough (needs input) flags set for all input ports. This is needed because a nonvirtual subsystem with a triggered sample time executes both its output and update methods together within the context of the model's output method.
- Fixed handling of grounded/unconnected inputs feeding certain blocks

  Simulink 4.0 incorrectly handled grounded or unconnected inputs to level-1 and level-2 S-functions requiring contiguous inputs and to some Matrix blocks. This has been fixed in Simulink 4.1.

Upgrading from an Earlier Release

This section discusses upgrade issues in moving from Simulink 4.0 to Simulink 4.1.

See “Upgrading from an Earlier Release” on page 3-10 in the Simulink 4.0 Release Notes for upgrade issues involved in moving from Simulink 3.0 (Release 11.0) to Simulink 4.1.

Running Simulink 4.1 Models in Simulink 4.0

Simulink 4.0 can run models created or saved by Simulink 4.1 as long as the models do not use features introduced in the new version, including new block types and block parameters. In particular, you should not attempt to use Simulink 4.0 to simulate or even open models that use the new Simulink control flow blocks. Opening such models cause Simulink 4.0 to crash.

Simulink Block Library Reorganization

The Simulink Block Library contains a new Subsystems sublibrary. The new library contains most of the new control flow blocks as well as subsystem and subsystem-related blocks that used to reside in the Signals & Systems library. The subsystems in the new library each contain the minimum set of blocks needed to create a functioning subsystem, e.g., an input port and an output port.

Direct Feedthrough Compensation Deprecated

If an S-function needs the current value of its input to compute its output, it must set its direct feedthrough flag to true. Previously, if a direct feedthrough S-function failed to do this, Simulink tried to provide a valid signal to the S-function’s `mdlOutput` (M-file `flag=3`) or `mdlGetTimeOfNextVarHit` (M-file `flag=4`) methods. This special compensation mode for S-functions was flawed. For this reason, the current version deprecates the mode, though making it available as an option. In this version, by default, if an S-function sets its direct feedthrough flag to false during initialization, Simulink sets the S-function’s input signal to `NULL` (or a `NaN` signal for M-file S-functions) during the `mdlOutput` or `mdlGetTimeOfNextVarHit` methods. Thus, in this version, models with S-function(s) may produce segmentation violations. See `*matlabroot*/simulink/src/sfuntmpl_directfeed.txt` for more information.

S-Functions Sorted Like Built-In Blocks

When sorting blocks, Simulink now treats S-function blocks the way it treats built-in blocks. This means that S-functions now work correctly in nonvirtual subsystems when there is a direct feedback connection (in Simulink 4.0 and prior, this wasn't the case). It also means that models compile (update diagram) faster. As a side effect, the execution order for S-functions that incorrectly set the direct feedthrough flag differs from that used in previous versions of Simulink. Consequently, models that contain invalid S-functions may produce different answers in this version of Simulink.

Added Latched Triggered Subsystems

Now triggered subsystems enable you to implement software triggering, hardware triggering, or a combination of the two. Software triggering is defined as

```
if (trigger_signal_edge_detected) {
   out(t) = f(in(t));
}
```

Hardware triggering is defined as

```
if (trigger_signal_edge_detected) {
   out(t) = f(in(t-h));   // h == last step size
}
```

Previous to this version, triggered subsystems provided software triggering and a form of hardware triggering when a cycle involving triggered subsystems existed. Now, you must explicitly specify whether or not you'd like software or hardware triggering. This is done by selecting `'Latch (buffer) input'` on the Inport blocks in a triggered subsystem.

Each input port of a triggered subsystem configures whether or not the input should be latched. A latched input provides the hardware-triggering semantics for that input port. Type `sl_subsys_semantics` at the MATLAB prompt for more information.

Self-Triggering Subsystems Are No Longer Allowed

Before this version, you could define the output of a triggered subsystem to directly feed back into the trigger port of the subsystem (with potentially other additive signals). This resulted in an implicit delay. Now you must explicitly define the delay by inserting a memory block.

Improved Invalid Model Configuration Diagnostics

This version of Simulink does a better job of detecting and flagging invalid modeling constructs in Simulink models. Consequently models that ran in previous versions of Simulink (sometimes producing incorrect results) may not run in the current release. The changes include:

- Direct feedthrough compensation no longer occurs by default for S-functions (see “Direct Feedthrough Compensation Deprecated” on page 2-12).
- S-functions are now sorted like built-in blocks (see “S-Functions Sorted Like Built-In Blocks” on page 2-13).
- Simulink no longer inserts implicit latches in triggered subsystems that directly or indirectly trigger themselves (see “Self-Triggering Subsystems Are No Longer Allowed” on page 2-13, above). Instead it signals an error when it detects a triggered subsystem loop with unlatched inputs. To avoid the error, you must select the **Latch** option on the triggered subsystem’s input ports.
- Simulink now signals an error when it detects invalid configurations of function-call subsystems. See the Subsystem Examples block in the Subsystems library for examples of illegal modeling constructs involving function-call subsystems. You can disable this diagnostic by setting the `Invalid FcnCall Connection` parameter on the **Diagnostics** pane of the **Simulation Parameters** dialog box to `none` or `warning`.

Known Software Problems

This section updates the Simulink 4.1 documentation set, reflecting known Simulink 4.1 software problems.

Accelerator Mode Does Not Support Inline Parameters

Accelerator mode does not support the **Inline parameters** option on the **Advanced** pane of the **Simulation Parameters** dialog box. Block outputs may appear at different times in accelerated mode than they do when the model is running in normal mode with the **Inline parameters** option selected.

Turn the New Wrap Lines Option Off

The MATLAB Command Window has a new **Wrap lines** option. Many Simulink error messages are very long. This can cause some display problems. Therefore, when using Simulink, you should turn the **Wrap lines** option off using the **Preferences** setting. For more information on this issue, see the Technical Support Solution 29082 from the MathWorks Web page.

Simulink 4.0 Release Notes

New Features

This section introduces the new features and enhancements added in Simulink 4.0 since Simulink 3.0 (Release 11.0).

This section about new Simulink features is organized into the following subsections:

Simulink Editor

This section describes enhancements to the Simulink Editor.

Preferences

The Simulink **Preferences** dialog box allows you to specify default settings for many options (see “Setting Simulink Preferences” in Using Simulink).

Text Alignment

Simulink 4.0 allows you to choose various alignments for annotation text. To choose an alignment for an annotation, select the annotation and then select **Text Alignment** from the editor menubar or context (right-click) menu (see “Annotations” in Using Simulink).

UNIX Context Menus

The UNIX version of Simulink 4.0 now has context menus for block diagrams. Click the right button on your mouse to display the menu.

Library Link Enhancements

Simulink 4.0 optionally displays an arrow in each block that represents a library link in a model. Simulink 4.0 also allows you to modify a link in a model and propagate the changes back to the library (see “Modifying a Linked Subsystem” in Using Simulink).

Note Simulink displays “Parameterized Link” on the parameter dialog box of a masked subsystem whose parameters differ from the library reference block to which the masked subsystem is linked. This feature, which is not documented in Using Simulink, allows you to determine quickly whether a library link differs from its reference.

Find Dialog Box

The **Find** dialog box enables you to search Simulink models and Stateflow charts for objects that satisfy specified search criteria. You can use the dialog box to find annotations, blocks, signals, states, state transitions, etc. To invoke the **Find** dialog, select **Find** from the Simulink **Edit** menu (see “Searching for Objects” in Using Simulink).

Model Browser

The Model Browser’s toolbar includes the following new buttons:

- Show Library Links

  Shows library links as nodes in the browser tree.
- Look Under Masks

  Shows the contents of masked blocks as nodes in the browser tree.

Single Window Mode

Simulink now provides two modes for opening subsystems. In multiwindow mode, Simulink opens each subsystem in a new window. In single-window mode, Simulink closes the parent and opens the subsystem (see “Window Reuse” in *Using Simulink*).

Keyboard Navigation

Simulink 4.0 provides the following new keyboard shortcuts.

Key	Action
Tab	Selects the next block in the block diagram.
Shift+Tab	Selects the previous block in the block diagram.

Key	Action
Ctrl+Tab	Cycles between the browser tree pane and the diagram pane when the model browser is enabled.
Enter	Opens the currently selected subsystem.
Esc	Opens the parent of the current subsystem.

Enhanced Library Browser

The Library Browser incorporates the following new features:

- Blocks no longer appear as browser tree nodes. Instead, they appear as icons in the preview pane.
- The preview pane has moved from beneath the library tree pane to beside the tree pane. You can create instances of blocks displayed in the preview pane by dragging them from the preview pane and dropping them in a model.
- Splitter bars now divide the browser's panes, allowing the panes to be independently resized.
- Double-clicking a block's icon opens the block's parameter dialog box with all fields disabled. This allows you to inspect, but not modify, a library block's parameters.
- Double-clicking a library block opens the library in the preview pane.
- You can now insert a block in the topmost model on your screen by right-clicking the block in the preview pane and selecting **Insert in...** from the context menu that appears. If no model is open or the topmost model is a locked library, the Library Browser offers to create a model in which to insert the block.
- The browser now contains a menu with **File**, **Edit**, and **Help** options.
- The block help text pane has moved from the bottom of the Library Browser to the top.
- Selecting **Find** from the Library Browser's **Edit** menu displays a modeless **Find** dialog box.
- The browser's search feature is much faster and supports regular expressions.

Help Menus

Simulink 4.0 adds a Help menu to the menu bar on model and library windows. The help item on a block context menu displays a help page for the block. The help item on the model context menu displays the first page of the Using Simulink book.

Modeling Enhancements

Hierarchical Variable Scoping

This release extends the ability of Simulink to resolve references to variables in masked subsystems. Previously Simulink could resolve references only to variables in a block's local workspace.With this release, Simulink will resolve references to variables located anywhere within the workspace hierarchy containing the block (see "The Mask Workspace" in *Using Simulink*).

Note In some cases, hierarchical scoping will cause some models to behave differently in the current release than in previous releases of Simulink.

Matrix Signals

Many Simulink blocks can now accept or output matrix signals. A matrix signal is a two-dimensional array of signal elements represented by a matrix. Each matrix element represents the value of the corresponding signal element at the current time step. In addition to matrix signals, Simulink also supports scalar (dimensionless) signals and vector signals (one-dimensional arrays of signals). Simulink can optionally thicken (select **Wide Lines** from the **Format** menu) and display the dimensions of lines (select **Line Dimensions** from the **Format** menu) that carry vector or matrix signals. When you select the **Line Dimensions** option, Simulink displays a label of the form `[r x c]` above a matrix signal line, where `r` is the number of rows and `c` is the number of columns. For example, the label `[2 x 3]` indicates that the line carries a two-row by three-column matrix signal.

You can use Simulink source blocks, such as a Sine Wave or a Constant block, to generate matrix signals. For example, to create a time-invariant matrix signal, insert a Constant block in your model and set its Constant Value parameter to any MATLAB expression that evaluates to a matrix, e.g., `[1 2;`

3 4], that represents the desired signal. See “Working with Signals” in Using Simulink for more information.

Simulink Data Objects

Simulink data objects allow a model to capture user-defined information about parameters and signals, such as minimum and maximum values, units, and so on (see “Working with Data Objects” in Using Simulink).

Block Execution Order

Simulink now optionally displays the execution order of each block on the model’s block diagram (see “Displaying Block Execution Order” in Using Simulink).

Simulink Debugger

This section describes enhancements to the Simulink debugger.

GUI Debugger Interface

Simulink 4.0 introduces a graphical user interface (GUI) for the Simulink Debugger. For more information, see “Simulink Debugger” in the online help for Simulink (see “Simulink Debugger” in Using Simulink).

Block Library

This section describes enhancements to the Simulink block libraries.

Product Block

The Product block now supports both element-by-element and matrix multiplication and inversion of inputs. The block’s parameter dialog includes a new **Multiplication** parameter that allows you to specify whether the block should multiply or invert inputs element-by-element or matrix-by-matrix.

Gain Block

The Gain block now supports matrix as well as element-wise multiplication of the input signal by a gain factor. Both input signals and gain factors can be matrices. The block’s parameter dialog includes a new **Multiplication** parameter that allows you to choose the following options:

- `K.*u` (element-wise product)

- `K*u` (matrix product with the gain as the left operand)
- `u*K` (matrix product with the gain as the right operand)

Math Function Block

The Math Function block adds two new matrix-specific functions: transpose and Hermitian. The first function outputs the transpose of the input matrix. The second function outputs the complex conjugate transpose (Hermitian) of the input matrix.

Reshape Block

Simulink 4.0 introduces the Reshape block, which changes the dimensionality of its input signals, based on an **Output dimensionality** parameter that you specify. For example, the block can change an n-element vector to a 1-by-N or N-by-1 matrix signal and vice versa. You can find the Reshape block in the Simulink Signals & Systems library.

Multiplexing Matrix Signals

The Simulink Mux, Demux, and Bus Selector blocks have been enhanced to support multiplexing of matrix signals.

Function Call Iteration Parameter

Simulink 4.0 adds a **Number of iterations** parameter to the Function Call Generator block. This parameter allows you to specify the number of times the target block is called per time step.

Probing Signal Dimensionality

The Probe block now optionally outputs the dimensionality of the signal connected to its input.

Configurable Subsystem

The Configurable Subsystem block has been reimplemented to make it easier to use. The configurable subsystem block now has a **Blocks** menu that allows you to choose which block the subsystem represents. To display the menu, select the configurable subsystem and then **Blocks** from the Simulink editor's **Edit** or context (right click) menu.

Look-Up Table Blocks

This release provides four new Look-Up Table (LUT) blocks.

- Direct Look-Up Table (n-D)
- Look-Up Table (n-D)
- Prelook-Up Index Search
- Interpolation (n-D) Using PreLook-Up

The blocks reside in the Simulink Functions and Tables block library.

Polynomial Block

The Polynomial block outputs a polynomial function of its input. The block resides in the Simulink Functions and Tables block library.

Signal Specification

The Signal Specification block allows you to specify the attributes that the input signal must satisfy. If the input signal does not meet the specification, the block generates an error.

ADA S-Functions

Simulink now supports S-functions coded in ADA. See “Creating Ada S-Functions” in *Writing S-Functions* for more information.

Bitwise Logical Operator Block

The Bitwise Logical Operator block is a new block that logically masks, inverts, or shifts the bits of an unsigned integer signal. See the online Simulink documentation for details.

Atomic Subsystems

Simulink 4.0 allows you to designate subsystems as *atomic* as opposed to *virtual*. An *atomic subsystem* is a true subsystem. When simulating a model, Simulink executes all blocks contained by an atomic subsystem block before executing the next block of the containing model (or atomic subsystem).

By declaring a subsystem atomic, you guarantee that Simulink completes execution of the subsystem before executing any other blocks at the same level in the model hierarchy. See “Atomic Versus Virtual Subsystems” in *Using Simulink* for more information.

Note Conditionally executed subsystems are inherently atomic. Simulink does not allow you to specify them as atomic or virtual.

SB2SL

SB2SL Extends Code Generation Support

SB2SL, which is included as part of Simulink, allows you to translate SystemBuild SuperBlocks to Simulink models.

For Release 12, SB2SL 2.1 has been enhanced to provide more complete support for use with the Real-Time Workshop. If you use the Real-Time Workshop 4.0 to generate code for models you have converted from SystemBuild to Simulink (using SB2SL), then code is generated for most translated blocks in the model.

The blocks that do *not* support code generation through the Real-Time Workshop 4.0 are:

- ConditionBlock
- Decoder
- Encoder
- GainScheduler
- Interp Table (Archive library)
- ShiftRegister

Note SB2SL 2.1 also includes a number of important bug fixes.

Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from Simulink 3.0 (Release 11.0) to Simulink 4.0.

Port Name Property

In previous releases, the name property of ports and lines referred to the label of the line connected to the port. In the current release, a port's name property refers to the port's (and line's) name, which, in the current release, can differ from the line's label.If you need to get the line's label, invoke

```
get_param(p, 'label')
```

where p is the handle of the port.