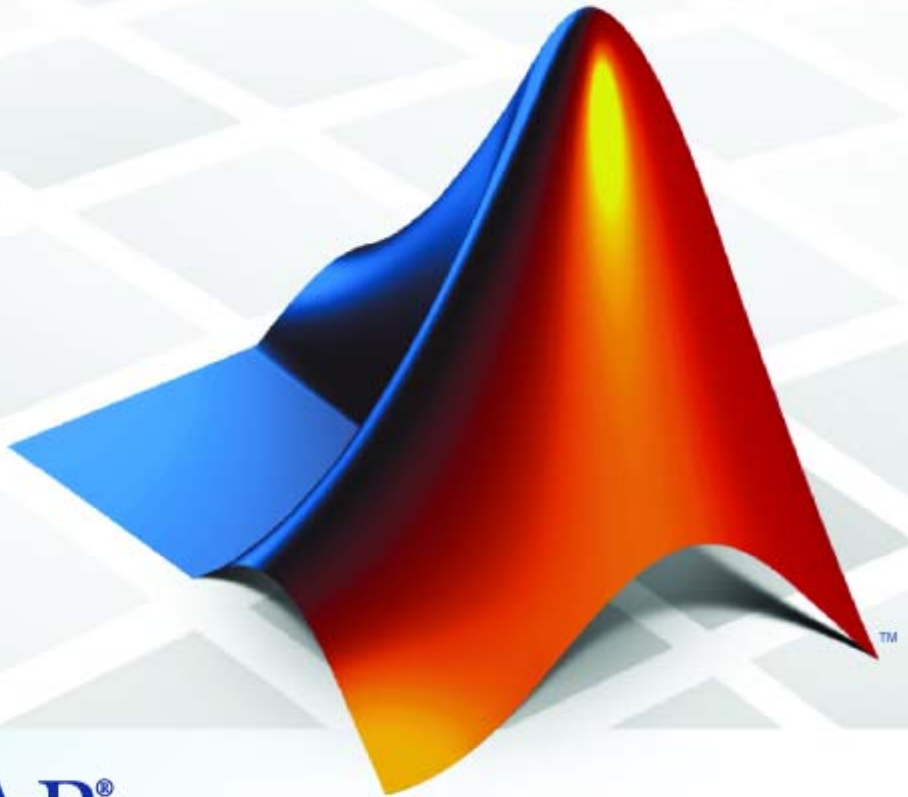


Simulink® 7

Graphical User Interface



MATLAB®
& SIMULINK®

How to Contact The MathWorks



www.mathworks.com
comp.soft-sys.matlab
www.mathworks.com/contact_TS.html

Web
Newsgroup
Technical Support



suggest@mathworks.com
bugs@mathworks.com
doc@mathworks.com
service@mathworks.com
info@mathworks.com

Product enhancement suggestions
Bug reports
Documentation error reports
Order status, license renewals, passcodes
Sales, pricing, and general information



508-647-7000 (Phone)



508-647-7001 (Fax)



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

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

Simulink® Graphical User Interface

© COPYRIGHT 1990–2008 by The MathWorks, Inc.

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

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

Trademarks

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

Patents

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

Revision History

September 2007	Online only	New for Simulink 7.0 (Release 2007b)
March 2008	Online only	Revised for Simulink 7.1 (Release 2008a)

Configuration Parameters Dialog Box

Configuration Parameters Dialog Box Overview (p. 1-3)	Brief overview of the Configuration Parameters dialog box
Model Configuration Pane (p. 1-6)	Parameters for specifying the name and description of your configuration set.
Solver Pane (p. 1-9)	Parameters for specifying the solver configuration
Data Import/Export Pane (p. 1-76)	Parameters for specifying data to import and export
Optimization Pane (p. 1-107)	Parameters for setting up optimizations
Diagnostics Pane: Solver (p. 1-168)	Parameters for specifying action to take when abnormal solver settings are detected
Diagnostics Pane: Sample Time (p. 1-192)	Parameters for specifying diagnostic actions to take when Simulink® software detects an abnormal condition or compilation error related to model sample times.
Diagnostics Pane: Data Validity (p. 1-208)	Parameters for specifying diagnostic actions to take when Simulink software detects a condition that could compromise the integrity of data defined by the model.

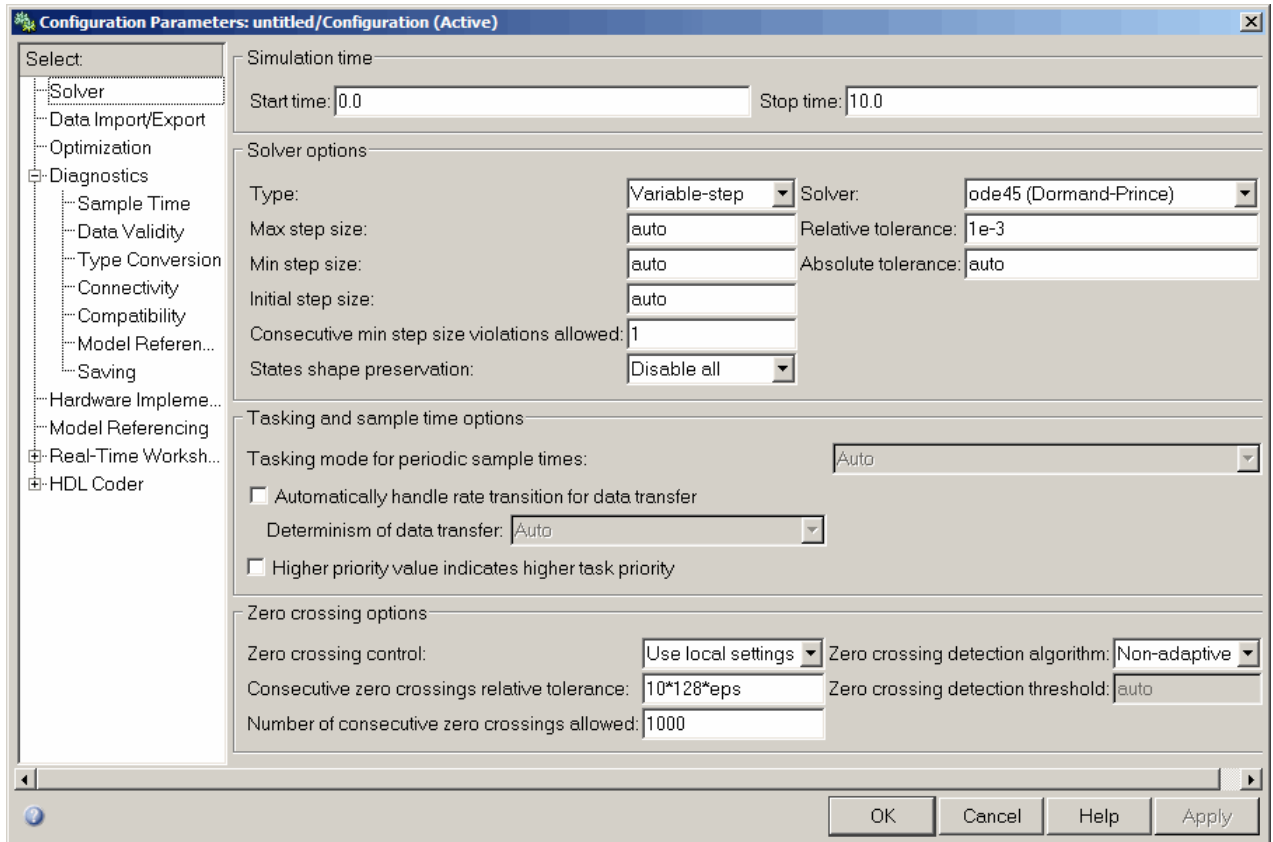
Diagnostics Pane: Type Conversion (p. 1-249)	Parameters for specifying diagnostic actions to take when Simulink software detects a data type conversion problem while compiling the model.
Diagnostics Pane: Connectivity (p. 1-256)	Parameters for specifying diagnostic actions to take when Simulink software detects a problem with block connections while compiling the model.
Diagnostics Pane: Compatibility (p. 1-280)	Parameters for specifying diagnostic actions to take when Simulink software detects an incompatibility between the current version of Simulink software and the model when updating or simulating the model.
Diagnostics Pane: Model Referencing (p. 1-293)	Parameters for specifying diagnostic actions to take when Simulink software detects an incompatibility relating to a model reference hierarchy
Diagnostics Pane: Saving (p. 1-307)	Parameters for specifying diagnostic actions to take when Simulink software saves a block diagram containing disabled library links or parameterized library links
Hardware Implementation Pane (p. 1-313)	Parameters for defining the hardware environment, including embedded and emulation hardware, for simulation and code generation
Model Referencing Pane (p. 1-374)	Parameters for specifying model referencing

Configuration Parameters Dialog Box Overview

The **Configuration Parameters** dialog box specifies the settings for a model's active *configuration set*. These parameters determine the type of solver used, import and export settings, and other values that determine how the model runs. See *Configuration Sets* for more information.

Note You can also use the Model Explorer to modify settings for the active configuration set or any other configuration set. See *The Model Explorer* for more information.

To display the dialog box, select **Simulation > Configuration Parameters** in the Model Editor, or press **Ctrl+E**. The dialog box appears.



The dialog box groups the configuration parameters into various categories. To display the parameters for a specific category, click the category in the **Select** tree on the left side of the dialog box.

In most cases, Simulink® software does not apply changes until you click **OK** or **Apply** at the bottom of the dialog box. The **OK** button applies your changes and dismisses the dialog box. The **Apply** button applies your changes but leaves the dialog box open.

Note Each of the parameters in the **Configuration Parameters** dialog box can also be set via the `sim` and `simset` commands. Each parameter description includes the corresponding command line information.

Model Configuration Pane

In this section...
“Model Configuration Overview” on page 1-6
“Name” on page 1-7
“Description” on page 1-8

Model Configuration Overview

View or edit the name and description of your configuration set.

In the Model Explorer you can edit the name and description of your configuration sets.

In the Model Explorer or Simulink® Preferences window you can edit the description of your template configuration set, Model Configuration Preferences. Go to the Model Configuration Preferences to edit the template Configuration Parameters to be used as defaults for new models.

When editing the Model Configuration preferences, you can click **Restore to Default Preferences** to restore the default configuration settings for creating new models. These underlying defaults cannot be changed.

Name

Specify the name of your configuration set.

Settings

Default: Configuration (for Active configuration set) or Configuration Preferences (for default configuration set).

Edit the name of your configuration set.

In the Model Configuration Preferences, the name of the default configuration is always Configuration Preferences, and cannot be changed.

Description

Specify a description of your configuration set.

Settings

No Default

Enter text to describe your configuration set.

Solver Pane

Simulation time	
Start time: <input type="text" value="0.0"/>	Stop time: <input type="text" value="10.0"/>
Solver options	
Type: <input type="text" value="Variable-step"/>	Solver: <input type="text" value="ode45 (Dormand-Prince)"/>
Max step size: <input type="text" value="auto"/>	Relative tolerance: <input type="text" value="1e-3"/>
Min step size: <input type="text" value="auto"/>	Absolute tolerance: <input type="text" value="auto"/>
Initial step size: <input type="text" value="auto"/>	
Consecutive min step size violations allowed: <input type="text" value="1"/>	
States shape preservation: <input type="text" value="Disable all"/>	
Tasking and sample time options	
Tasking mode for periodic sample times: <input type="text" value="Auto"/>	
<input type="checkbox"/> Automatically handle rate transition for data transfer	
<input type="checkbox"/> Higher priority value indicates higher task priority	
Zero crossing options	
Zero crossing control: <input type="text" value="Use local settings"/>	Zero crossing location algorithm: <input type="text" value="Non-adaptive"/>
Consecutive zero crossings relative tolerance: <input type="text" value="10*128*eps"/>	Zero crossing location threshold: <input type="text" value="auto"/>
Number of consecutive zero crossings allowed: <input type="text" value="1000"/>	

In this section...

- “Solver Overview” on page 1-11
- “Start time” on page 1-13
- “Stop time” on page 1-15
- “Type” on page 1-17
- “Solver” on page 1-20
- “Max Step Size” on page 1-27
- “Initial Step Size” on page 1-29
- “Min Step Size” on page 1-31
- “Relative tolerance” on page 1-33
- “Absolute tolerance” on page 1-35
- “Maximum order” on page 1-38
- “Solver reset method” on page 1-40

In this section...

“Consecutive min step size violations allowed” on page 1-42

“States shape preservation” on page 1-44

“Tasking mode for periodic sample times” on page 1-46

“Automatically handle rate transition for data transfer” on page 1-48

“Deterministic data transfer” on page 1-50

“Higher priority value indicates higher task priority” on page 1-52

“Zero crossing control” on page 1-54

“Consecutive zero crossings relative tolerance” on page 1-56

“Number of consecutive zero crossings allowed” on page 1-58

“Zero crossing location algorithm” on page 1-60

“Zero crossing location threshold” on page 1-62

“Periodic sample time constraint” on page 1-64

“Fixed step size (fundamental sample time)” on page 1-67

“Sample time properties” on page 1-69

“Extrapolation order” on page 1-72

“Number Newton’s iterations” on page 1-74

Solver Overview

Specify the simulation start and stop time, and the solver configuration for the simulation. Use the Solver pane to set up a solver for a model's active configuration set.

A solver computes a dynamic system's states at successive time steps over a specified time span, using information provided by the model.

Configuration

- 1 Select a solver type from the **Type** list.
- 2 Select a solver from the **Solver** list.
- 3 Set the parameters displayed for the selected type and solver combination.
- 4 Apply the changes.

Tips

- Simulation time is not the same as clock time. For example, running a simulation for 10 seconds usually does not take 10 seconds. Total simulation time depends on factors such as model complexity, solver step sizes, and computer speed.
- Fixed-point solver type is required for code generation, unless you use an S-function or RSim target.
- Variable-step solver type can significantly shorten the time required to simulate models in which states change rapidly or which contain discontinuities.

See Also

- Solvers
- Choosing a Solver
- Specifying a Simulation Start and Stop Time

- Configuration Parameters Dialog Box
- Solver Pane

Start time

Specify the start time for the simulation or generated code as a double-precision value, scaled to seconds.

Settings

Default: 0.0

- A start time other than 0.0 is an offset, and must be less than or equal to the stop time. An example of when you might use an offset is to set up a delay to accommodate some type of initialization.
- The values of block parameters with initial conditions must match the initial condition settings at the specified start time.
- Simulation time is not the same as clock time. For example, running a simulation for 10 seconds usually does not take 10 seconds. Total simulation time depends on factors such as model complexity, solver step sizes, and computer speed.

Command-Line Information

Parameter: StartTime

Type: string

Value: any valid value

Default: '0.0'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	0.0

See Also

- [Specifying a Simulation Start and Stop Time](#)
- [Configuration Parameters Dialog Box](#)
- [Solver Pane](#)

Stop time

Specify the stop time for the simulation or generated code as a double-precision value, scaled to seconds.

Settings

Default: 10

- Stop time must be greater than or equal to the start time.
- Specify `inf` to run a simulation or generated program until you explicitly pause or stop it.
- If the stop time is the same as the start time, the simulation or generated program runs for one step.
- Simulation time is not the same as clock time. For example, running a simulation for 10 seconds usually does not take 10 seconds. Total simulation time depends on factors such as model complexity, solver step sizes, and computer speed.
- If your model includes blocks that depend on absolute time and you are creating a design that runs indefinitely, see [Blocks That Depend on Absolute Time](#).

Command-Line Information

Parameter: `StopTime`

Type: string

Value: any valid value

Default: '10.0'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	Any positive value

See Also

- [Blocks That Depend on Absolute Time](#)
- [Using Blocks to Stop or Pause a Simulation](#)
- [Specifying a Simulation Start and Stop Time](#)
- [Configuration Parameters Dialog Box](#)
- [Solver Pane](#)

Type

Select the type of solver you want to use to simulate your model.

Settings

Default: Variable-step

Variable-step

Step size varies from step to step, depending on model dynamics. A variable-step solver:

- Reduces step size when model states change rapidly, to maintain accuracy.
- Increases step size when model states change slowly, to avoid unnecessary steps.

Variable-step is recommended for models in which states change rapidly or that contain discontinuities. In these cases, a variable-step solver requires fewer time steps than a fixed-step solver to achieve a comparable level of accuracy. This can significantly shorten simulation time.

Fixed-step

Step size remains constant throughout the simulation.

Required for code generation, unless you use an S-function or RSim target.

Note The solver computes the next time as the sum of the current time and the step size.

Dependencies

Selecting Variable-step enables the following parameters:

- **Solver**
- **Max step size**
- **Min step size**

- **Initial step size**
- **Relative tolerance**
- **Absolute tolerance**
- **Zero crossing control**
- **Consecutive min step size violations allowed**
- **Consecutive zero crossings relative tolerance**
- **Number of consecutive zero crossings allowed**

Selecting Fixed-step enables the following parameters:

- **Solver**
- **Periodic sample time constraint**
- **Fixed-step size (fundamental sample time)**
- **Tasking mode for periodic sample times**
- **Higher priority value indicates higher task priority**
- **Automatically handle data transfers between tasks**

Command-Line Information

Parameter: SolverType

Type: string

Value: 'Variable-step' | 'Fixed-step'

Default: 'Variable-step'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	Fixed-step

See Also

- [Solvers](#)
- [Choosing a Solver](#)
- [Determining Step Size for Discrete Systems](#)
- [Configuration Parameters Dialog Box](#)
- [Solver Pane](#)

Solver

Select the solver you want to use to compute the model's states during simulation or code generation.

Settings

The available solvers change depending on which solver Type you selected:

- “Fixed-step Solvers” on page 1-20
- “Variable-step Solvers” on page 1-22

Fixed-step Solvers. Default: ode3 (Bogacki-Shampine)

ode3 (Bogacki-Shampine)

Computes the model's state at the next time step as an explicit function of the current value of the state and the state derivatives, using the Bogacki-Shampine Formula integration technique to compute the state derivatives. In the following example, X is the state, DX is the state derivative, and h is the step size:

$$X(n+1) = X(n) + h * DX(n)$$

discrete (no continuous states)

Computes the time of the next time step by adding a fixed step size to the current time.

Use this solver for models with no states or discrete states only, using a fixed step size. Relies on the model's blocks to update discrete states.

The accuracy and length of time of the resulting simulation depends on the size of the steps taken by the simulation: the smaller the step size, the more accurate the results but the longer the simulation takes.

Note The fixed-step discrete solver cannot be used to simulate models that have continuous states.

ode5 (Dormand-Prince)

Computes the model's state at the next time step as an explicit function of the current value of the state and the state derivatives, using the Dormand-Prince Formula integration technique to compute the state derivatives. In the following example, X is the state, DX is the state derivative, and h is the step size:

$$X(n+1) = X(n) + h * DX(n)$$

ode4 (Runge-Kutta)

Computes the model's state at the next time step as an explicit function of the current value of the state and the state derivatives, using the Fourth-Order Runge-Kutta (RK4) Formula integration technique. In the following example, X is the state, DX is the state derivative, and h is the step size:

$$X(n+1) = X(n) + h * DX(n)$$

ode2 (Heun)

Computes the model's state at the next time step as an explicit function of the current value of the state and the state derivatives, using the Heun's Method integration technique. In the following example, X is the state, DX is the state derivative, and h is the step size:

$$X(n+1) = X(n) + h * DX(n)$$

ode1 (Euler)

Computes the model's state at the next time step as an explicit function of the current value of the state and the state derivatives, using the Euler's Method integration technique. In the following example, X is the state, DX is the state derivative, and h is the step size:

$$X(n+1) = X(n) + h * DX(n)$$

ode14x (extrapolation)

Computes the model's state at the next time step as an *implicit* function of the state and state derivative at the next time step, using a combination of Newton's method and extrapolation from the current value. In the following example, X is the state, DX is the state derivative, and h is the step size:

$$X(n+1) - X(n) - h * DX(n+1) = 0$$

This solver requires more computation per step than an explicit solver, but is more accurate for a given step size.

Variable-step Solvers. Default: ode45 (Dormand-Prince)

ode45 (Dormand-Prince)

Computes the model's state at the next time step using an explicit Runge-Kutta (4,5) formula (the Dormand-Prince pair) for numerical integration.

ode45 is a one-step solver, needing only the solution to the preceding time point.

Use ode45 as a first try for most problems.

discrete (no continuous states)

Computes the time of the next step by adding a step size that varies depending on the rate of change of the model's states.

Use this solver for models with no states or discrete states only, using a variable step size.

ode23 (Bogacki-Shampine)

Computes the model's state at the next time step using an explicit Runge-Kutta (2,3) formula (the Bogacki-Shampine pair) for numerical integration.

ode23 is a one-step solver, needing only the solution to the preceding time point.

ode23 is more efficient than ode45 at crude tolerances and in the presence of mild stiffness.

ode113 (Adams)

Computes the model's state at the next time step using a variable-order Adams-Bashforth-Moulton PECE numerical integration technique.

ode113 is a multistep solver, generally needing the solutions at several preceding time points to compute the current solution.

ode113 can be more efficient than ode45 at stringent tolerances.

ode15s (stiff/NDF)

Computes the model's state at the next time step using variable-order numerical differentiation formulas (NDFs). These are related to but are more efficient than the backward differentiation formulas, BDFs (also known as Gear's method).

ode15s is a multistep solver, generally needing the solutions at several preceding time points to compute the current solution.

ode15s is efficient for stiff problems. Try this solver if ode45 fails or is inefficient.

ode23s (stiff/Mod. Rosenbrock)

Computes the model's state at the next time step using a modified Rosenbrock formula of order 2.

ode23s is a one-step solver, needing only the solution to the preceding time point.

ode23s is more efficient than ode15s at crude tolerances, and can solve stiff problems for which ode15s is ineffective.

ode23t (Mod. stiff/Trapezoidal)

Computes the model's state at the next time step using an implementation of the trapezoidal rule using a "free" interpolant.

ode23t is a one-step solver, needing only the solution to the preceding time point.

Use ode23t if the problem is only moderately stiff and you need a solution without numerical damping.

ode23tb (stiff/TR-BDF2)

Computes the model's state at the next time step using a multistep implementation of TR-BDF2, an implicit Runge-Kutta formula with a trapezoidal rule first stage, and a second stage consisting of a backward differentiation formula of order two. By construction, the same iteration matrix is used in evaluating both stages.

ode23tb is more efficient than ode15s at crude tolerances, and can solve stiff problems for which ode15s is ineffective.

Tips

- Identifying the optimal solver for a model requires experimentation, for an in-depth discussion, see Choosing a Solver.
- The optimal solver balances acceptable accuracy with the shortest simulation time.
- Simulink® software uses a discrete solver for any model with no states or discrete states only, even if you specify a continuous solver.
- A smaller step size increases accuracy, but increases simulation time.
- The degree of computational complexity increases for ode_n , as n increases.
- As computational complexity increases, accuracy of results increases.

Dependencies

Selecting the `ode1` (Euler), `ode2` (Huen), `ode3` (Bogacki-Shampine), `ode4` (Runge-Kutta), `ode5` (Dormand-Prince), or `discrete` (no continuous states) fixed-step solvers enables the following parameters:

- **Periodic sample time constraint**
- **Fixed-step size (fundamental sample time)**
- **Tasking mode for periodic sample times**
- **Higher priority value indicates higher task priority**
- **Automatically handle data transfers between tasks**

Selecting `ode14x` (extrapolation) enables the following parameters:

- **Periodic sample time constraint**
- **Fixed-step size (fundamental sample time)**
- **Extrapolation order**
- **Number Newton's iterations**
- **Tasking mode for periodic sample times**
- **Higher priority value indicates higher task priority**
- **Automatically handle data transfers between tasks**

Selecting the discrete (no continuous states) variable-step solver enables the following parameters:

- **Max step size**
- **Zero crossing control**
- **Consecutive zero crossings relative tolerance**
- **Number of consecutive zero crossings allowed**

Selecting ode45 (Dormand-Prince), ode23 (Bogacki-Shampine), ode113 (Adams), or ode23s (stiff/Mod. Rosenbrock) enables the following parameters:

- **Max step size**
- **Min step size**
- **Initial step size**
- **Relative tolerance**
- **Absolute tolerance**
- **Zero crossing control**
- **Consecutive min step size violations allowed**
- **Consecutive zero crossings relative tolerance**
- **Number of consecutive zero crossings allowed**

Selecting ode15s (stiff/NDF), ode23t (Mod. stiff/Trapezoidal), or ode23tb (stiff/TR-BDF2) enables the following parameters:

- **Max step size**
- **Min step size**
- **Initial step size**
- **Relative tolerance**
- **Absolute tolerance**
- **Maximum order**

- **Zero crossing control**
- **Solver reset method**
- **Consecutive min step size violations allowed**
- **Consecutive zero crossings relative tolerance**
- **Number of consecutive zero crossings allowed**

Command-Line Information

Parameter: Solver

Type: string

Value: 'VariableStepDiscrete' | 'ode45' | 'ode23' | 'ode113' | 'ode15s' | 'ode23s' | 'ode23t' | 'ode23tb' | 'FixedStepDiscrete' | 'ode5' | 'ode4' | 'ode3' | 'ode2' | 'ode1' | 'ode14x'

Default: 'ode45'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	discrete (no continuous states)

See Also

- Solvers
- Choosing a Solver
- Determining Step Size for Discrete Systems
- Configuration Parameters Dialog Box
- Solver Pane

Max Step Size

Specify the largest time step that the solver can take.

Settings

Default: auto

- For the discrete solver, the default value (auto) is the model's shortest sample time.
- For continuous solvers, the default value (auto) is determined from the start and stop times. If the stop time equals the start time or is `inf`, Simulink software chooses 0.2 seconds as the maximum step size. Otherwise, it sets the maximum step size to

$$h_{max} = \frac{t_{stop} - t_{start}}{50}$$

Tips

- Generally, the default maximum step size is sufficient. If you are concerned about the solver's missing significant behavior, change the parameter to prevent the solver from taking too large a step.
- If the time span of the simulation is very long, the default step size might be too large for the solver to find the solution.
- If your model contains periodic or nearly periodic behavior and you know the period, set the maximum step size to some fraction (such as 1/4) of that period.
- In general, for more output points, change the refine factor, not the maximum step size. For more information, see [Specifying Output Options](#).

Dependencies

This parameter is enabled only if the solver **Type** is set to `Variable-step`.

Command-Line Information

Parameter: MaxStep
Type: string
Value: any valid value
Default: 'auto'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Determining Step Size for Discrete Systems
- Specifying Output Options
- Configuration Parameters Dialog Box
- Solver Pane

Initial Step Size

Specify the size of the first time step that the solver takes.

Settings

Default: auto

By default, the solver selects an initial step size by examining the derivatives of the states at the start time.

Tips

- Be careful when increasing the initial step size. If the first step size is too large, the solver might step over important behavior.
- The initial step size parameter is a *suggested* first step size. The solver tries this step size but reduces it if error criteria are not satisfied.

Dependencies

This parameter is enabled only if the solver **Type** is set to Variable-step.

Command-Line Information

Parameter: InitialStep

Type: string

Value: any valid value

Default: 'auto'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- [Determining Step Size for Discrete Systems](#)
- [Improving Simulation Performance and Accuracy](#)
- [Configuration Parameters Dialog Box](#)
- [Solver Pane](#)

Min Step Size

Specify the smallest time step that the solver can take.

Settings

Default: auto

- The default value (auto) sets a minimum step size on the order of machine precision and an unlimited number of warnings.
- You can specify either a real number greater than zero, or a two-element vector where the first element is the minimum step size and the second element is the maximum number of minimum step size warnings before issuing an error.

Tips

- If the solver takes a smaller step to meet error tolerances, it issues a warning indicating the current effective relative tolerance.
- Setting the second element to zero results in an error the first time the solver must take a step smaller than the specified minimum. This is equivalent to changing the **Min step size violation** diagnostic to error on the **Diagnostics** pane (see Min step size violation).
- Setting the second element to -1 results in an unlimited number of warnings. This is also the default if the input is a scalar.

Dependencies

This parameter is enabled only if the solver **Type** is set to Variable-step.

Command-Line Information

Parameter: MinStep
Type: string
Value: any valid value
Default: 'auto'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Determining Step Size for Discrete Systems
- Min step size violation
- Configuration Parameters Dialog Box
- Solver Pane

Relative tolerance

Specify the largest acceptable solver error, relative to the size of each state during each time step. If the relative error exceeds this tolerance, the solver reduces the time step.

Settings

Default: 1e-3

- The relative tolerance is a percentage of the state's value.
- The default value (1e-3) means that the computed state is accurate to within 0.1%.

Tips

- The acceptable error at each time step is a function of both **Relative tolerance** and **Absolute tolerance**. For more information about how these settings work together, see [Specifying Variable-Step Solver Error Tolerances](#).
- During each time step, the solver computes the state values at the end of the step and also determine the local error, the estimated error of these state values. If the error is greater than the acceptable error for any state, the solver reduces the step size and tries again.
- The default relative tolerance value is sufficient for most applications. Decreasing the relative tolerance value can slow down the simulation.
- To check the accuracy of a simulation after you run it, you can reduce the relative tolerance to 1e-4 and run it again. If the results of the two simulations are not significantly different, you can feel confident that the solution has converged.

Dependencies

This parameter is enabled only if you set:

- Solver **Type** to `Variable-step`.
- **Solver** to a continuous variable step solver.

This parameter works along with **Absolute tolerance** to determine the acceptable error at each time step. For more information about how these settings work together, see [Specifying Variable-Step Solver Error Tolerances](#).

Command-Line Information

Parameter: RelTol
Type: string
Value: any valid value
Default: '1e-3'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- [Specifying Variable-Step Solver Error Tolerances](#)
- [Improving Simulation Performance and Accuracy](#)
- [Configuration Parameters Dialog Box](#)
- [Solver Pane](#)

Absolute tolerance

Specify the largest acceptable solver error, as the value of the measured state approaches zero. If the absolute error exceeds this tolerance, the solver reduces the time step.

Settings

Default: auto

- The default value (auto) initially sets the absolute tolerance for each state to 1e-6. As the simulation progresses, the absolute tolerance for each state is reset to the maximum value that the state has assumed thus far, times the relative tolerance for that state.

For example, if a state goes from 0 to 1 and the **Relative tolerance** is 1e-3, then by the end of the simulation the **Absolute tolerance** is set to 1e-3.

- If the computed setting is not suitable, you can determine an appropriate setting yourself.

Tips

- The acceptable error at each time step is a function of both **Relative tolerance** and **Absolute tolerance**. For more information about how these settings work together, see [Specifying Variable-Step Solver Error Tolerances](#).
- The Integrator, Transfer Fcn, State-Space, and Zero-Pole blocks allow you to specify absolute tolerance values for solving the model states that they compute or that determine their output. The absolute tolerance values that you specify in these blocks override the global setting in the Configuration Parameters dialog box.
- You might want to override the **Absolute tolerance** setting using blocks if the global setting does not provide sufficient error control for all of your model's states, such as if they vary widely in magnitude.
- If you set the **Absolute tolerance** too low, the solver may take too many steps around near-zero state values, slowing down the simulation.
- To check the accuracy of a simulation after you run it, you can reduce the absolute tolerance and run it again. If the results of the two simulations

are not significantly different, you can feel confident that the solution has converged.

- If your simulation results do not seem accurate, and your model has states whose values approach zero, the **Absolute tolerance** may be too large. Reduce the **Absolute tolerance** to force the simulation to take more steps around areas of near-zero state values.

Dependencies

This parameter is enabled only if you set:

- Solver **Type** to Variable-step.
- **Solver** to a continuous variable step solver.

This parameter works along with **Relative tolerance** to determine the acceptable error at each time step. For more information about how these settings work together, see [Specifying Variable-Step Solver Error Tolerances](#).

Command-Line Information

Parameter: AbsTol
Type: string
Value: any valid value
Default: 'auto'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- [Specifying Variable-Step Solver Error Tolerances](#)
- [Improving Simulation Performance and Accuracy](#)
- [Configuration Parameters Dialog Box](#)
- [Solver Pane](#)

Maximum order

Select the order of the numerical differentiation formulas (NDFs) used in the ode15s solver.

Settings

Default: 5

- 5 Specifies that the solver use fifth order NDFs.
- 1 Specifies that the solver use first order NDFs.
- 2 Specifies that the solver use second order NDFs.
- 3 Specifies that the solver use third order NDFs.
- 4 Specifies that the solver use fourth order NDFs.

Tips

- Although the higher order formulas are more accurate, they are less stable.
- If your model is stiff and requires more stability, reduce the maximum order to 2 (the highest order for which the NDF formula is A-stable).
- As an alternative, you can try using the ode23s solver, which is a lower order (and A-stable) solver.

Dependencies

This parameter is enabled only if **Solver** is set to ode15s.

Command-Line Information

Parameter: MaxOrder

Type: integer

Value: 1 | 2 | 3 | 4 | 5

Default: 5

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- [Specifying Variable-Step Solver Error Tolerances](#)
- [Improving Simulation Performance and Accuracy](#)
- [Configuration Parameters Dialog Box](#)
- [Solver Pane](#)

Solver reset method

Select how the solver behaves during a reset, such as when it detects a zero crossing.

Settings

Default: Fast

Fast

Specifies that the solver not recompute the Jacobian matrix at a solver reset.

Robust

Specifies that the solver recompute the Jacobian matrix needed by the integration step at every solver reset.

Tips

- Selecting Fast speeds up the simulation. However, it can result in incorrect solutions in some cases.
- If you suspect that the simulation is giving incorrect results, try the Robust setting. If there is no difference in simulation results between the fast and robust settings, revert to the fast setting.

Dependencies

This parameter is enabled only if you select one of the following solvers:

- ode15s (Stiff/NDF)
- ode23t (Mod. Stiff/Trapezoidal)
- ode23tb (Stiff/TR-BDF2)

Command-Line Information

Parameter: SolverResetMethod

Type: string

Value: 'Fast' | 'Robust'

Default: 'Fast'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Choosing a Solver
- Configuration Parameters Dialog Box
- Solver Pane

Consecutive min step size violations allowed

Specify the maximum number of consecutive minimum step size violations allowed during simulation.

Settings

Default: 1

- A minimum step size violation occurs when a variable-step continuous solver takes a smaller step than that specified by the **Min step size** property (see Min step size).
- Simulink software counts the number of consecutive violations that it detects. If the count exceeds the value of **Consecutive min step size violations allowed**, Simulink software displays either a warning or error message as specified by the **Min step size violation** diagnostic (see Min step size violation).

Dependencies

This parameter is enabled only if you set:

- Solver **Type** to Variable-step.
- **Solver** to a continuous variable step solver.

Command-Line Information

Parameter: MaxConsecutiveMinStep

Type: string

Value: any valid value

Default: '1'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	No impact

See Also

- Choosing a Solver
- Min step size violation
- Min step size
- Configuration Parameters Dialog Box
- Solver Pane

States shape preservation

At each time step use derivative information to improve integration accuracy.

Settings

Default: Disable all

Disable all

Do not perform states shape preservation on any signals

Enable all

Perform states shape preservation on all signals.

Tips

- The default setting (Disable all) usually provides good accuracy for most models.
- Setting to Enable all will increase accuracy in those models having signals who's derivative exhibits a high rate of change, but simulation time may be increased.

Dependencies

This parameter is enabled only if a continuous step solver is used.

Command-Line Information

Parameter: ShapePreserveControl

Type: string

Value: 'EnableAll' | 'DisableAll'

Default: 'DisableAll'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	No impact

See Also

- Zero-Crossing Detection
- Solver Pane

Tasking mode for periodic sample times

Select how blocks with periodic sample times execute.

Settings

Default: Auto

Auto

Specifies that single-tasking execution is used if:

- Your model contains one sample time.
- Your model contains a continuous and a discrete sample time, and the fixed-step size is equal to the discrete sample time.

Selects multitasking execution for models operating at different sample rates.

SingleTasking

Specifies that all blocks are processed through each stage of simulation (for example, calculating output and updating discrete states) together.

MultiTasking

Specifies that groups of blocks with the same execution priority are processed through each stage of simulation (for example, calculating output and updating discrete states) based on task priority. Multitasking mode helps to create valid models of real-world multitasking systems, where sections of your model represent concurrent tasks.

Tip

The **Multitask rate transition** parameter on the **Diagnostics > Sample Time** pane allows you to adjust error checking for sample rate transitions between blocks that operate at different sample rates.

Dependency

This parameter is enabled by selecting Fixed-step solver type.

Command-Line Information

Parameter: SolverMode

Type: string

Value: 'Auto' | 'SingleTasking' | 'MultiTasking'

Default: 'Auto'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Rate Transition block
- Model Execution and Rate Transitions
- Single-Tasking and Multitasking Execution Modes
- Sample Rate Transitions
- Single-Tasking and Multitasking Execution of a Model: an Example
- Configuration Parameters Dialog Box
- Solver Pane

Automatically handle rate transition for data transfer

Specify whether Simulink software automatically inserts hidden Rate Transition blocks between blocks that have different sample rates to ensure the integrity of data transfers between tasks and optionally ensure determinism of data transfers for periodic tasks.

Settings

Default: Off



Inserts hidden Rate Transition blocks between blocks when rate transitions are detected. Handles rate transitions for asynchronous and periodic tasks. Simulink software adds the hidden blocks configured to ensure data integrity for data transfers. Selecting this option also enables the parameter **Deterministic data transfer**, which allows you to control the level of data transfer determinism for periodic tasks.



Does not insert hidden Rate Transition blocks when rate transitions are detected. If Simulink software detects invalid transitions, you must adjust the model such that the sample rates for the blocks in question match or manually add a Rate Transition block.

See Rate Transition Block Options in the Real-Time Workshop® documentation for further details.

Tips

- Selecting this parameter allows you to handle rate transition issues automatically. This saves you from having to manually insert Rate Transition blocks to avoid invalid rate transitions, including invalid asynchronous-to-periodic and asynchronous-to-asynchronous rate transitions, in multirate models.
- For asynchronous tasks, Simulink software configures the inserted blocks to ensure data integrity but not determinism during data transfers.

Command-Line Information

Parameter: AutoInsertRateTranBlk

Type: string

Value: 'on' | 'off'

Default: 'off'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	Off

See Also

- Rate Transition Block Options
- Configuration Parameters Dialog Box
- Solver Pane

Deterministic data transfer

Control whether the Rate Transition block parameter **Ensure deterministic data transfer (maximum delay)** is set for auto-inserted Rate Transition blocks

Default: Whenever possible

Always

Specifies that the block parameter **Ensure deterministic data transfer (maximum delay)** is always set for auto-inserted Rate Transition blocks.

If Always is selected and a model needs to auto-insert a Rate Transition block to handle a rate transition that is *not* between two periodic sample-times related by an integer multiple, Simulink errors out.

Whenever possible

Specifies that the block parameter **Ensure deterministic data transfer (maximum delay)** is set for auto-inserted Rate Transition blocks whenever possible. If an auto-inserted Rate Transition block handles data transfer between two periodic sample-times that are related by an integer multiple, **Ensure deterministic data transfer (maximum delay)** is set; otherwise, it is cleared.

Never (minimum delay)

Specifies that the block parameter **Ensure deterministic data transfer (maximum delay)** is never set for auto-inserted Rate Transition blocks.

Note Clearing the Rate Transition block parameter **Ensure deterministic data transfer (maximum delay)** can provide reduced latency for models that do not require determinism. See the description of **Ensure deterministic data transfer (maximum delay)** on the Rate Transition block reference page for more information.

Dependencies

This parameter is enabled only if **Automatically handle rate transition for data transfer** is checked.

Command-Line Information

Parameter: InsertRTBMode

Type: string

Value: 'Always' | 'Whenever possible' | 'Never (minimum delay)'

Default: 'Whenever possible'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	'Whenever possible'

See Also

- Rate Transition Block Options
- Configuration Parameters Dialog Box
- Solver Pane

Higher priority value indicates higher task priority

Specify whether the real-time system targeted by the model assigns higher or lower priority values to higher priority tasks when implementing asynchronous data transfers

Settings

Default: Off

On

Real-time system assigns higher priority values to higher priority tasks, for example, 8 has a higher task priority than 4. Rate Transition blocks treat asynchronous transitions between rates with lower priority values and rates with higher priority values as low-to-high rate transitions.

Off

Real-time system assigns lower priority values to higher priority tasks, for example, 4 has a higher task priority than 8. Rate Transition blocks treat asynchronous transitions between rates with lower priority values and rates with higher priority values as high-to-low rate transitions.

Command-Line Information

Parameter: PositivePriorityOrder

Type: string

Value: 'on' | 'off'

Default: 'off'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Rate Transitions and Asynchronous Blocks
- Configuration Parameters Dialog Box
- Solver Pane

Zero crossing control

Enables zero-crossing detection during variable-step simulation of the model. For most models, this speeds up simulation by enabling the solver to take larger time steps.

Settings

Default: Use local settings

Use local settings

Specifies that zero-crossing detection be enabled on a block-by-block basis. You can enable zero-crossing detection on a block-by-block basis. For a list of those blocks, see

To specify zero-crossing detection for one of these blocks, open the block's parameter dialog box and select the **Enable zero crossing detection** option.

Enable all

Enables zero-crossing detection for all blocks in the model.

Disable all

Disables zero-crossing detection for all blocks in the model.

Tips

- For most models, enabling zero-crossing detection speeds up simulation by allowing the solver to take larger time steps.
- If a model has extreme dynamic changes, disabling this option can speed up the simulation but can also decrease the accuracy of simulation results. See Zero Crossing Detection for more information.
- Selecting Enable all or Disable all overrides the local zero-crossing detection setting for individual blocks.

Dependencies

This parameter is enabled only if the solver **Type** is set to Variable-step.

Selecting either Use local settings or Enable all enables the following parameters:

- **Consecutive zero crossings relative tolerance**
- **Number of consecutive zero crossings allowed**
- **Zero crossing threshold algorithm**

Command-Line Information

Parameter: ZeroCrossControl

Type: string

Value: 'UseLocalSettings' | 'EnableAll' | 'DisableAll'

Default: 'UseLocalSettings'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Zero-Crossing Detection
- Number of consecutive zero crossings allowed
- Consecutive zero crossings violation
- Consecutive zero crossings relative tolerance
- Configuration Parameters Dialog Box
- Solver Pane

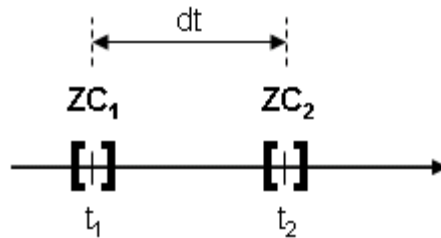
Consecutive zero crossings relative tolerance

Specify a tolerance factor that controls how closely zero-crossing events must occur to be considered consecutive.

Settings

Default: $10 \cdot 128 \cdot \text{eps}$

- Simulink software defines zero crossings as consecutive if the time between events is less than a particular interval. The following figure depicts a simulation timeline during which Simulink software detects zero crossings ZC_1 and ZC_2 , bracketed at successive time steps t_1 and t_2 .



Simulink software determines that the zero crossings are consecutive if

$$dt < \text{RelTolZC} * t_2$$

where dt is the time between zero crossings and RelTolZC is the **Consecutive zero crossings relative tolerance**.

- Simulink software counts the number of consecutive zero crossings that it detects. If the count exceeds the value of **Number of consecutive zero crossings allowed**, Simulink software displays either a warning or error as specified by the **Consecutive zero crossings violation** diagnostic (see Consecutive zero crossings violation).

Tips

- Simulink software resets the counter each time it detects nonconsecutive zero crossings (successive zero crossings that fail to meet the relative tolerance setting), therefore, decreasing the relative tolerance value may afford your model's behavior more time to recover.

- If your model experiences excessive zero crossings, you can also increase the **Number of consecutive zero crossings allowed** to increase the threshold at which Simulink software triggers the **Consecutive zero crossings violation** diagnostic.

Dependencies

This parameter is enabled only if **Zero crossing control** is set to either Use local settings or Enable all.

Command-Line Information

Parameter: ConsecutiveZCsStepRelTol

Type: string

Value: any valid value

Default: '10*128*eps'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Zero Crossing Detection
- Zero Crossing Control
- Number of consecutive zero crossings allowed
- Consecutive zero crossings violation
- Configuration Parameters Dialog Box
- Solver Pane

Number of consecutive zero crossings allowed

Specify the number of consecutive zero crossings that can occur before Simulink software displays a warning or error.

Settings

Default: 1000

- Simulink software counts the number of consecutive zero crossings that it detects. If the count exceeds the specified value, Simulink software displays either a warning or error as specified by the **Consecutive zero crossings violation** diagnostic (see Consecutive zero crossings violation).
- Simulink software defines zero crossings as consecutive if the time between events is less than a particular interval (see Consecutive zero crossings relative tolerance).

Tips

- If your model experiences excessive zero crossings, you can increase this parameter to increase the threshold at which Simulink software triggers the **Consecutive zero crossings violation** diagnostic. This may afford your model's behavior more time to recover.
- Simulink software resets the counter each time it detects nonconsecutive zero crossings, therefore, decreasing the relative tolerance value may also afford your model's behavior more time to recover.

Dependencies

This parameter is enabled only if **Zero crossing control** is set to either Use local settings or Enable all.

Command-Line Information

Parameter: MaxConsecutiveZCs

Type: string

Value: any valid value

Default: '1000'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Zero-Crossing Detection
- Zero-Crossing Control
- Consecutive zero crossings violation
- Consecutive zero crossings relative tolerance
- Configuration Parameters Dialog Box
- Solver Pane

Zero crossing location algorithm

Specifies the algorithm to detect zero-crossings when a variable-step solver is used.

Settings

Default: Non-adaptive

Adaptive

Use an improved zero-crossing algorithm which dynamically activates and deactivates zero crossing bracketing. With this algorithm you can set a zero crossing tolerance. See “Zero crossing location threshold” on page 1-62 to learn how to set the zero crossing tolerance.

Non-adaptive

Use the nonadaptive zero crossing algorithm present in the Simulink software prior to Version 7.0 (R2008a). This option is provided for backward compatibility.

Tips

- The Adaptive zero crossing algorithm is especially useful in systems having strong “chattering”, or Zeno behavior. In these systems this algorithm yields shorter simulation run times compared to the non-adaptive algorithm. See Zero Crossing Detection for more information.

Dependencies

- This parameter is enabled only if the solver **Type** is set to Variable-step.
- Selecting Adaptive enables the **Zero crossing location threshold** parameter.

Command-Line Information

Parameter: ZeroCrossAlgorithm

Type: string

Value: 'Non-adaptive' | 'Adaptive'

Default: 'Non-adaptive'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Zero-Crossing Detection
- Number of consecutive zero crossings allowed
- Consecutive zero crossings violation
- Consecutive zero crossings relative tolerance
- Configuration Parameters Dialog Box
- Solver Pane

Zero crossing location threshold

Specifies the deadband region used during the detection of zero crossings. Signals falling within this region are defined as having crossed through zero.

The zero crossing tolerance is a real number, greater than or equal to zero.

Settings

Default: Auto

Auto

The zero crossing tolerance is determined automatically by the adaptive algorithm.

String

Use the specified value for the zero crossing tolerance. The value must be a real number equal to or greater than zero.

Tips

- Do not confuse the **Zero Crossing Tolerance** with the **Consecutive zero crossings relative tolerance** parameter.
- Entering too small of a small value for the **Zero Crossing Location Tolerance** parameter will result in long simulation run times and increased difficulty in detecting a zero crossing.
- Entering a large **Zero Crossing Location Tolerance** value may improve the simulation speed (especially in those systems with large chattering). Making the value too large may reduce simulation accuracy.

Dependency

This parameter is enabled if the **Zero crossing algorithm** is set to Adaptive.

Command-Line Information

Parameter: ZcDetectionTol

Type: string

Value: 'auto' | any real number greater than or equal to zero

Default: 'auto'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Zero-Crossing Detection
- Number of consecutive zero crossings allowed
- Consecutive zero crossings violation
- Consecutive zero crossings relative tolerance
- Configuration Parameters Dialog Box
- Solver Pane

Periodic sample time constraint

Select constraints on the sample times defined by this model. If the model does not satisfy the specified constraints during simulation, Simulink software displays an error message.

Settings

Default: Unconstrained

Unconstrained

Specifies no constraints. Selecting this option causes Simulink software to display a field for entering the solver step size.

Use the **Fixed step size (fundamental sample time)** option to specify solver step size.

Ensure sample time independent

Specifies that Model blocks inherit sample time from the context in which they are used. This option is required if you use the Model block in a triggered subsystem. For more information, see:

- Model Block Sample Times
- Inherited Sample Time for Referenced Models
- Function Call Models

Simulink software checks to ensure that this model can inherit its sample times from a model that references it without altering its behavior. Models that specify a step size (i.e., a base sample time) cannot satisfy this constraint. For this reason, selecting this option causes Simulink software to hide the group's step size field (see Fixed-step size (fundamental sample time)).

Specified

Specifies that Simulink software check to ensure that this model operates at a specified set of prioritized periodic sample times. Use the **Sample time properties** option to specify and assign priorities to model sample times.

Executing Multitasking Models explains how to use this option for multitasking models.

Tips

During simulation, Simulink software checks to ensure that the model satisfies the constraints. If the model does not satisfy the specified constraint, Simulink software displays an error message.

Dependencies

This parameter is enabled only if the solver **Type** is set to Fixed-step.

Selecting Unconstrained enables the following parameters:

- **Fixed-step size (fundamental sample time)**
- **Tasking mode for periodic sample times**
- **Higher priority value indicates higher task priority**
- **Automatically handle data transfers between tasks**

Selecting Specified enables the following parameters:

- **Sample time properties**
- **Tasking mode for periodic sample times**
- **Higher priority value indicates higher task priority**
- **Automatically handle data transfers between tasks**

Command-Line Information

Parameter: SampleTimeConstraint

Type: string

Value: 'unconstrained' | 'STIndependent' | 'Specified'

Default: 'unconstrained'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	Specified or Ensure sample time independent

See Also

- Model Block Sample Times
- Inherited Sample Time for Referenced Models
- Function Call Models
- Fixed-step size (fundamental sample time)
- Executing Multitasking Models
- Configuration Parameters Dialog Box
- Solver Pane

Fixed step size (fundamental sample time)

Specify the step size used by the selected fixed-step solver.

Settings

Default: auto

- Entering auto (the default) in this field causes Simulink software to choose the step size.
- If the model specifies one or more periodic sample times, Simulink software chooses a step size equal to the least common denominator of the specified sample times. This step size, known as the fundamental sample time of the model, ensures that the solver will take a step at every sample time defined by the model.
- If the model does not define any periodic sample times, Simulink software chooses a step size that divides the total simulation time into 50 equal steps.

Dependencies

This parameter is enabled only if the **Periodic sample time constraint** is set to Unconstrained.

Command-Line Information

Parameter: FixedStep

Type: string

Value: any valid value

Default: 'auto'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Modeling Dynamic Systems
- Configuration Parameters Dialog Box
- Solver Pane

Sample time properties

Specify and assign priorities to the sample times that this model implements.

Settings

No Default

- Enter an Nx3 matrix with rows that specify the model's discrete sample time properties in order from fastest rate to slowest rate.
- Faster sample times must have higher priorities.

Format.

```
[period, offset, priority]
```

period	The time interval (sample rate) at which updates occur during the simulation.
offset	A time interval indicating an update delay. The block is updated later in the sample interval than other blocks operating at the same sample rate.
priority	Execution priority of the real-time task associated with the sample rate.

See Specifying Sample Time for more detail and options for specifying sample time.

Example.

```
[[0.1, 0, 10]; [0.2, 0, 11]; [0.3, 0, 12]]
```

- Declares that the model should specify three sample times.
- Sets the fundamental sample time period to 0.1 second.
- Assigns priorities of 10, 11, and 12 to the sample times.
- Assumes higher priority values indicate lower priorities — the **Higher priority value indicates higher task priority** option is not selected.

Tips

- If the model's fundamental rate differs from the fastest rate specified by the model, specify the fundamental rate as the first entry in the matrix followed by the specified rates, in order from fastest to slowest. See *Determining Step Size for Discrete Systems*.
- If the model operates at one rate, enter the rate as a three-element vector in this field — for example, [0.1, 0, 10].
- When you update a model, Simulink software displays an error message if what you specify does not match the sample times defined by the model.
- If **Periodic sample time constraint** is set to Unconstrained, Simulink software assigns priority 40 to the model base sample rate. If **Higher priority value indicates higher task priority** is selected, Simulink software assigns priorities 39, 38, 37, and so on, to subrates of the base rate. Otherwise, it assigns priorities 41, 42, 43, and so on, to the subrates.
- Continuous rate is assigned a higher priority than is the discrete base rate regardless of whether **Periodic sample time constraint** is Specified or Unconstrained.

Dependencies

This parameter is enabled by selecting Specified from the **Periodic sample time constraint** list.

Command-Line Information

Parameter: SampleTimeProperty

Type: structure

Value: any valid matrix

Default: []

Note If you specify SampleTimeProperty at the command line, you must enter the sample time properties as a structure with the following fields:

- SampleTime
- Offset
- Priority

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	Period, offset, and priority of each sample time in the model; faster sample times must have higher priority than slower sample times

See Also

- Determining Step Size for Discrete Systems
- Specifying Sample Time
- Configuration Parameters Dialog Box
- Solver Pane

Extrapolation order

Select the extrapolation order used by the ode14x solver to compute a model's states at the next time step from the states at the current time step.

Settings

Default: 4

- 1 Specifies first order extrapolation.
- 2 Specifies second order extrapolation.
- 3 Specifies third order extrapolation.
- 4 Specifies fourth order extrapolation.

Tip

Selecting a higher order produces a more accurate solution, but is more computationally intensive per step size.

Dependencies

This parameter is enabled by selecting ode14x (extrapolation) from the **Solver** list.

Command-Line Information

Parameter: ExtrapolationOrder

Type: integer

Value: 1 | 2 | 3 | 4

Default: 4

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- [Choosing a Fixed-Step Solver](#)
- [Configuration Parameters Dialog Box](#)
- [Solver Pane](#)

Number Newton's iterations

Specify the number of Newton's method iterations used by the ode14x solver to compute a model's states at the next time step from the states at the current time step.

Settings

Default: 1

Minimum: 1

Maximum: 2147483647

More iterations produce a more accurate solution, but is more computationally intensive per step size.

Dependencies

This parameter is enabled by selecting ode14x (extrapolation) from the **Solver** list.

Command-Line Information

Parameter: NumberNewtonIterations

Type: integer

Value: any valid number

Default: 1

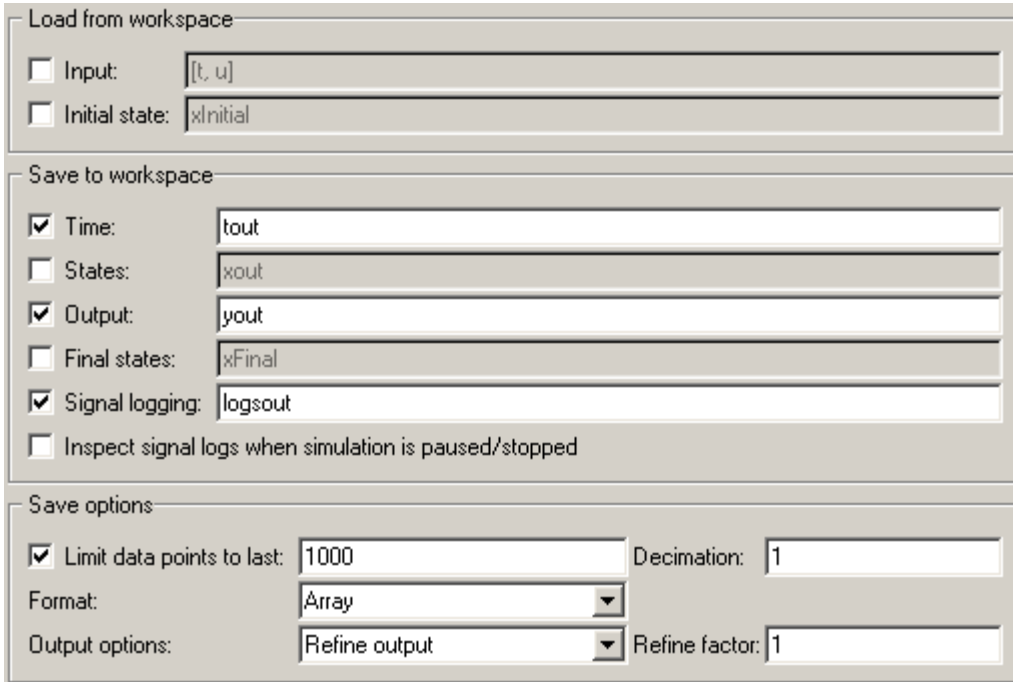
Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- [Choosing a Fixed-Step Solver](#)
- [Configuration Parameters Dialog Box](#)
- [Solver Pane](#)

Data Import/Export Pane



Load from workspace

Input: [t, u]

Initial state: xInitial

Save to workspace

Time: tout

States: xout

Output: yout

Final states: xFinal

Signal logging: logstdout

Inspect signal logs when simulation is paused/stopped

Save options

Limit data points to last: 1000 Decimation: 1

Format: Array

Output options: Refine output Refine factor: 1

In this section...

“Data Import/Export Overview” on page 1-78

“Input” on page 1-79

“Initial State” on page 1-81

“Time” on page 1-83

“States” on page 1-85

“Output” on page 1-87

“Final states” on page 1-89

“Signal logging” on page 1-91

“Inspect signal logs when simulation is paused/stopped” on page 1-93

In this section...

“Limit data points to last” on page 1-95

“Decimation” on page 1-97

“Format” on page 1-99

“Output options” on page 1-101

“Refine factor” on page 1-103

“Output times” on page 1-105

Data Import/Export Overview

The Data Import/Export pane allows you to import input signal and initial state data from a workspace and export output signal and state data to the MATLAB® workspace during simulation. This capability allows you to use standard or custom MATLAB functions to generate a simulated system's input signals and to graph, analyze, or otherwise postprocess the system's outputs.

Configuration

- 1 Specify the data to load from a workspace before simulation begins.
- 2 Specify the data to save to the MATLAB workspace after simulation completes.

Tips

- For more information on using this pane, see [Importing and Exporting Simulation Data](#).
- See the documentation of the `sim` command for some capabilities that are available only for programmatic simulation.

See Also

- [Importing Data from a Workspace](#)
- [Exporting Data to the MATLAB Workspace](#)
- [Configuration Parameters Dialog Box](#)
- [Data Import/Export Pane](#)

Input

Loads input data from a workspace before the simulation begins.

Settings

Default: Off, [t,u]



On

Loads data from a workspace.

Specify a MATLAB expression for the data to be imported from a workspace. The Simulink® software resolves symbols used in this specification as described in “Resolving Symbols”. The input data can take any of the following forms:

- Time series
- Data array
- Time expression
- Data structure

See Importing Data from a Workspace for information on how to use this field.



Off

Does not load data from a workspace.

Tips

- You must select the **Input** check box before entering input data.
- Simulink software linearly interpolates or extrapolates input values as necessary if the Interpolate data option is selected for the corresponding Inport.
- The use of the **Input** box is independent of the setting of the **Format** list on the **Data Import/Export** pane.

Command-Line Information

Parameter: LoadExternalInput

Type: string

Value: 'on' | 'off'

Default: 'off'

Parameter: ExternalInput

Type: scalar or vector

Value: any valid value

Default: [t,u]

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact for simulation and during development Off for production code generation

See Also

- Importing Data from a Workspace
- Configuration Parameters Dialog Box
- Data Import/Export Pane

Initial State

Loads the model's initial states from a workspace before simulation begins.

Settings

Default: Off, `xInitial`



On

Simulink software loads initial states from a workspace.

Specify the name of a variable that contains the initial state values, for example, a variable containing states saved from a previous simulation.

Use the structure or structure-with-time option to specify initial states if you want to accomplish any of the following:

- Associate initial state values directly with the full path name to the states. This eliminates errors that could occur if Simulink software reorders the states, but the initial state array is not correspondingly reordered.
- Assign a different data type to each state's initial value.
- Initialize only a subset of the states.

See [Importing and Exporting States](#) for more information.



Off

Simulink software does not load initial states from a workspace.

Tips

- You must select the **Initial State** check box before entering initial state data.
- The initial values specified by the workspace variable override the initial values specified by the model itself (the values specified by the initial condition parameters of those blocks in the model that have states).
- You must use the structure or structure-with-time format to initialize the states of a top model and the models that it references.

Command-Line Information

Parameter: LoadInitialState

Type: string

Value: 'on' | 'off'

Default: 'off'

Parameter: InitialState

Type: variable (string) or vector

Value: any valid value

Default: 'xInitial'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact for simulation and during development Off for production code generation

See Also

- Importing Data from a Workspace
- Importing and Exporting States
- Configuration Parameters Dialog Box
- Data Import/Export Pane

Time

Saves simulation time data to the specified variable during simulation.

Settings

Default: On, tout



On

Simulink software exports time data to the MATLAB workspace during simulation.

Specify the name of the MATLAB variable used to store time data. See [Exporting Data to the MATLAB Workspace](#) for more information.



Off

Simulink software does not export time data to the MATLAB workspace during simulation.

Tips

- You must select the **Time** check box before entering the time variable.
- Simulink software saves the output to the MATLAB workspace at the base sample rate of the model. Use a To Workspace block if you want to save output at a different sample rate.
- The **Save options** area enables you to specify the format and restrict the amount of output saved.

Command-Line Information

Parameter: SaveTime

Type: string

Value: 'on' | 'off'

Default: 'on'

Parameter: TimeSaveName

Type: string

Value: any valid value

Default: 'tout'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact for simulation and during development Off for production code generation

See Also

- Exporting Data to the MATLAB Workspace
- Configuration Parameters Dialog Box
- Data Import/Export Pane

States

Saves state data to the specified MATLAB variable during a simulation.

Settings

Default: Off, xout



On

Simulink software exports state data to the MATLAB workspace during simulation.

Specify the name of the MATLAB variable used to store state data. See [Importing and Exporting States](#) for more information.



Off

Simulink software does not export state data during simulation.

Tips

- You must select the **States** check box before entering the states variable.
- Simulink software saves the states in a MATLAB workspace variable having the specified name.
- The saved data has the format that you specify in the **Save options** area.

Command-Line Information

Parameter: SaveState

Type: string

Value: 'on' | 'off'

Default: 'off'

Parameter: StateSaveName

Type: string

Value: any valid value

Default: 'xout'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact for simulation and during development Off for production code generation

See Also

- Importing and Exporting States
- Configuration Parameters Dialog Box
- Data Import/Export Pane

Output

Saves signal data to the specified MATLAB variable during simulation.

Settings

Default: On, yout



On

Simulink software exports signal data to the MATLAB workspace during simulation.

Specify the name of the MATLAB variable used to store signal data. See [Exporting Data to the MATLAB Workspace](#) for more information.



Off

Simulink software does not export signal data during simulation.

Tips

- You must select the **Output** check box before entering the output variable.
- Simulink software saves the output to the MATLAB workspace at the base sample rate of the model. Use a To Workspace block if you want to save output at a different sample rate.
- The **Save options** area enables you to specify the format and restrict the amount of output saved.

Command-Line Information

Parameter: SaveOutput

Type: string

Value: 'on' | 'off'

Default: 'on'

Parameter: OutputSaveName

Type: string

Value: any valid value

Default: 'yout'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact for simulation and during development Off for production code generation

See Also

- Exporting Data to the MATLAB Workspace
- Configuration Parameters Dialog Box
- Data Import/Export Pane

Final states

Saves the model's states at the end of a simulation to the specified MATLAB variable.

Settings

Default: Off, xFinal



On

Simulink software exports final state data to the MATLAB workspace during simulation.

Specify the name of the MATLAB variable used to store the values of the final states. See [Importing and Exporting States](#) for more information.



Off

Simulink software does not export final state data during simulation.

Tips

- You must select the **Final states** check box before entering the final states variable.
- Simulink software saves the final states in a MATLAB workspace variable having the specified name.
- The saved data has the format that you specify in the **Save options** area.

Command-Line Information

Parameter: SaveFinalState

Type: string

Value: 'on' | 'off'

Default: 'off'

Parameter: FinalStateName

Type: string

Value: any valid value

Default: 'xFinal'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact for simulation and during development Off for production code generation

See Also

- Importing and Exporting States
- Configuration Parameters Dialog Box
- Data Import/Export Pane

Signal logging

Globally enable or disable signal logging for this model.

Settings

Default: On, logout



On

Enables signal logging to the MATLAB workspace during simulation.

Specify the name of the signal logging object used to record logged signal data in the MATLAB workspace. See Logging Signals for more information.



Off

Disables signal logging to the MATLAB workspace during simulation.

Tips

- You must select the **Signal logging** check box before entering the signal logging variable.
- Simulink software saves the signal data in a MATLAB workspace variable having the specified name.
- The saved data has the format that you specify in the **Save options** area.
- Simulink software does not support signal logging for the following types of signals:
 - Output of a Function-Call Generator block
 - Signal connected to the input of a Merge block
 - Outputs of Trigger and Enable blocks

Dependencies

This parameter enables **Inspect signal logs when simulation is paused/stopped**.

Command-Line Information

Parameter: SignalLogging

Type: string

Value: 'on' | 'off'

Default: 'on'

Parameter: SignalLoggingName

Type: string

Value: any valid value

Default: 'logsOut'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact for simulation and during development Off for production code generation

See Also

- Logging Signals
- Configuration Parameters Dialog Box
- Data Import/Export Pane

Inspect signal logs when simulation is paused/stopped

Specify whether Simulink software displays logged signals in the MATLAB **Time Series Tools** viewer at the end of a simulation or whenever you pause the simulation.

Settings

Default: Off



On

Simulink software displays logged signals in the MATLAB **Time Series Tools** viewer at the end of a simulation or whenever you pause the simulation.



Off

Simulink software does not display logged signals at the end of a simulation.

Tips

- If this option is off, you must select **Tools > Inspect logged signals** in the model editor to display logged signals in the **Time Series Tools** viewer.
- You must run the simulation before selecting **Tools > Inspect logged signals**. Otherwise, the command has no effect.

Dependencies

This parameter is enabled by **Signal logging**.

Command-Line Information

Parameter: InspectSignalLogs

Type: string

Value: 'on' | 'off'

Default: 'off'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact for simulation and during development Off for production code generation

See Also

- Logging Signals
- Configuration Parameters Dialog Box
- Data Import/Export Pane

Limit data points to last

Specify the number of data points exported to the MATLAB workspace be limited to the number specified.

Settings

Default: On, 1000



On

Limits the number of data points exported to the MATLAB workspace to the number specified.

Specify the maximum number of data points exported to the MATLAB workspace. At the end of the simulation, the MATLAB workspace contains the last N points generated by the simulation.



Off

Does not limit the number of data points.

Tips

- You must select the **Limit data points to last** check box before specifying the number of data points.
- Saving data to the MATLAB workspace can slow down the simulation and consume memory. Use this parameter to limit the number of samples saved to help avoid this problem.
- You can also apply a **Decimation** factor to skip a selected number of samples.

Command-Line Information

Parameter: LimitDataPoints

Type: string

Value: 'on' | 'off'

Default: 'on'

Parameter: MaxDataPoints

Type: string

Value: any valid value

Default: '1000'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact for simulation and during development Off for production code generation

See Also

- Exporting Data to the MATLAB Workspace
- Configuration Parameters Dialog Box
- Data Import/Export Pane

Decimation

Specify that Simulink software output only every N points, where N is the specified decimation factor.

Settings

Default: 1

- The default value (1) specifies that all data points are saved.
- Simulink software outputs data only at the specified number of data points. For example, specifying 2 saves every other data point, while specifying 10 saves just one in ten data points.
- At the end of the simulation, the total number of data points is reduced by the factor specified.

Tips

- Saving data to the MATLAB workspace can slow down the simulation and consume memory. Use this parameter to limit the number of samples saved to help avoid this problem.
- You can also use the **Limit data points to last** parameter to help resolve this problem.

Command-Line Information

Parameter: Decimation

Type: string

Value: any valid value

Default: '1'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	No impact for simulation and during development Off for production code generation

See Also

- Exporting Data to the MATLAB Workspace
- Configuration Parameters Dialog Box
- Data Import/Export Pane

Format

Select the format of state and output data saved to the MATLAB workspace.

Settings

Default: Array

Array

The format of the data is a matrix each row of which corresponds to a simulation time step.

Structure

The format of the data is a structure that contains substructures for each port. Each port substructure contains signal data for the corresponding port.

Structure with time

The format of the data is a structure that has two fields: a time field and a signals field. The time field contains a vector of simulation times. The signals field contains a substructure for each model input port (for imported data) or output port (for exported data). Each port substructure contains signal data for the corresponding port.

Tips

- You can use array format to save your model's outputs and states only if the outputs are either all scalars or all vectors (or all matrices for states), are either all real or all complex, and are all of the same data type. Use the Structure or Structure with time output formats (see Structure with time) if your model's outputs and states do not meet these conditions.
- Simulink software can read back simulation data saved to the workspace in the Structure with time output format. See Importing Signal-and-Time Data Structures for more information.

Command-Line Information

Parameter: SaveFormat

Type: string

Value: 'Array' | 'Structure' | 'StructureWithTime'

Default: 'Array'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact for simulation and during development Off for production code generation

See Also

- Exporting Data to the MATLAB Workspace
- Configuration Parameters Dialog Box
- Data Import/Export Pane

Output options

Select options for generating additional output signal data for variable-step solvers.

Settings

Default: Refine output

Refine output

Generates data output between as well as at simulation times steps.

Use this field to specify the number of points to generate between simulation time steps. For more information, see Refining Output.

Produce additional output

Generates additional output at specified times. Use this field to specify the simulation times at which Simulink software should generate additional output.

Produce specified output only

Generates output only at specified times. Use this field to specify the simulation times at which Simulink software should generate output.

Tips

- These settings can help the solver locate zero crossings. In particular, it helps reduce the chance of missing a zero crossing. It does not help locate the missed zero crossings.
- For additional information on how Simulink software calculates outputs for these three options, see Specifying Output Options.

Dependencies

This parameter is enabled only if the model specifies a variable-step solver (see Solver Type).

Selecting Refine output enables the **Refine factor** parameter.

Selecting Produce additional output or Produce specified output only enables the **Output times** parameter.

Command-Line Information

Parameter: OutputOption

Type: string

Value: 'RefineOutputTimes' | 'AdditionalOutputTimes' | 'SpecifiedOutputTimes'

Default: 'RefineOutputTimes'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact for simulation and during development Off for production code generation

See Also

- Specifying Output Options
- Refine factor
- Refining Output
- Exporting Data to the MATLAB Workspace
- Configuration Parameters Dialog Box
- Data Import/Export Pane

Refine factor

Specify how many points to generate between time steps to refine the output.

Settings

Default: 1

- The default refine factor is 1, meaning that no extra data points are generated.
- A refine factor of 2 provides output midway between the time steps, as well as at the steps.

Tip

Simulink software ignores this option for discrete models. This is because the value of data between time steps is undefined for discrete models.

Dependency

This parameter is enabled only if you select `Refine` output as the value of **Output options**.

Command-Line Information

Parameter: `Refine`
Type: string
Value: any valid value
Default: '1'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	No impact for simulation and during development Off for production code generation

See Also

- Refining Output
- Configuration Parameters Dialog Box
- Data Import/Export Pane

Output times

Specify the times at which Simulink software should generate output in addition to or instead of at the simulation steps taken by the solver used to simulate the model.

Settings

Default: []

- Enter a matrix containing the times at which Simulink software should generate additional output.
- The default refine factor [], indicates that no extra data points are generated.

Tips

- The Produce additional output option generates output at the specified times as well as the regular simulation steps.
- The Produce specified output only option generates output only at the specified times.
- Discrete models define outputs only at major time steps. Therefore, Simulink software logs output for discrete models only at major time steps. If the **Output times** field specifies other times, Simulink software displays a warning at the MATLAB command line.

Dependency

This parameter is enabled only if you select Produce additional output or Produce specified output only as the value of **Output options**.

Command-Line Information

Parameter: OutputTimes

Type: string

Value: any valid value

Default: '[]'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact for simulation and during development Off for production code generation

See Also

- Refining Output
- Configuration Parameters Dialog Box
- Data Import/Export Pane

Optimization Pane

Simulation and code generation

Block reduction

Implement logic signals as boolean data (vs. double).

Inline parameters

Conditional input branch execution

Signal storage reuse

Application lifespan (days):

Accelerating simulations

Compiler optimization level:

Verbose accelerator builds

Code generation

Signals

Enable local block outputs

Reuse block outputs

Ignore integer downcasts in folded expressions

Inline invariant signals

Eliminate superfluous temporary variables (Expression folding)

Loop unrolling threshold:

Integer and fixed-point

Remove code from floating-point to integer conversions that wraps out-of-range values

Stateflow

Use bitsets for storing state configuration

Minimize array reads using temporary variables

Use bitsets for storing boolean data

In this section...

“Optimization Overview” on page 1-110

In this section...

- “Block reduction” on page 1-111
- “Conditional input branch execution” on page 1-115
- “Inline parameters” on page 1-117
- “Implement logic signals as boolean data (vs. double)” on page 1-120
- “Signal storage reuse” on page 1-122
- “Application lifespan (days)” on page 1-124
- “Parameter structure” on page 1-127
- “Enable local block outputs” on page 1-129
- “Ignore integer downcasts in folded expressions” on page 1-132
- “Eliminate superfluous temporary variables (Expression folding)” on page 1-135
- “Reuse block outputs” on page 1-137
- “Inline invariant signals” on page 1-140
- “Loop unrolling threshold” on page 1-142
- “Remove root level I/O zero initialization” on page 1-144
- “Use memset to initialize floats and doubles to 0.0” on page 1-146
- “Remove internal state zero initialization” on page 1-148
- “Optimize initialization code for model reference” on page 1-150
- “Remove code from floating-point to integer conversions that wraps out-of-range values” on page 1-152
- “Remove code that protects against division arithmetic exceptions” on page 1-155
- “Use bitsets for storing state configuration” on page 1-157
- “Use bitsets for storing boolean data” on page 1-159
- “Minimize array reads using temporary variables” on page 1-161
- “Model Parameter Configuration Dialog Box” on page 1-163

In this section...

“Compiler optimization level” on page 1-165

“Verbose accelerator builds” on page 1-167

Optimization Overview

Set up optimizations for a model's active configuration set. Optimizations are set for both simulation and code generation.

Configuration

Set the parameters displayed.

Tips

Stateflow® software optimizations appear only when Real-Time Workshop® and Stateflow products are both installed on your system and the model includes Stateflow charts or Embedded MATLAB™ Function blocks. The settings you make for the Stateflow software options also apply to all Embedded MATLAB Function blocks in the model. You do not need a Stateflow license to use Embedded MATLAB Function blocks.

See Also

- [Optimizing a Model for Code Generation](#)
- [Configuration Parameters Dialog Box](#)
- [Optimization Pane](#)

Block reduction

Reduce execution time by collapsing or removing groups of blocks.

Settings

Default: On



On

Simulink® software searches for and reduces the following block patterns:

- **Accumulators** — A group consisting of a Constant block, a Sum block, and feedback through a Unit Delay block.
- **Redundant type conversions** — Unnecessary type conversion blocks, such as an int type conversion block with an input and output of type int.
- **Dead code** — Blocks or signals in an unused code path.
- **Fast-to-slow Rate Transition block in a single-tasking system** — Rate Transition blocks with an input frequency faster than its output frequency.



Off

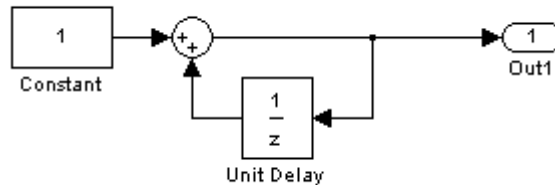
Simulink software does not search for block patterns that can be optimized. Simulation and generated code are not optimized.

Tips

- When you select **Block reduction**, Simulink software collapses certain groups of blocks into a single, more efficient block, or removes them entirely. This results in faster execution during model simulation and in generated code.
- Block reduction does not change the appearance of the source model.
- Tunable parameters do not prevent a block from being reduced by dead code elimination.
- Once block reduction takes place, Simulink software does not display the sorted order for blocks that have been removed.

- Block reduction is intended to remove only the generated code that represents execution of a block. Other supporting data, such as definitions for sample time and data types might remain in the generated code.

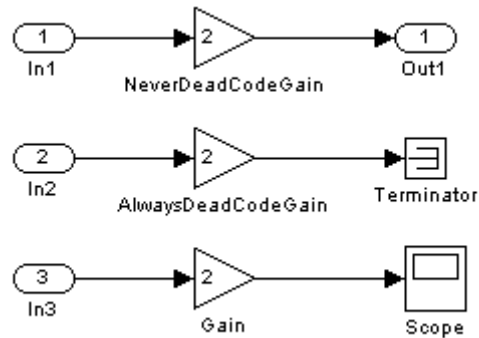
Accumulators. Simulink software recognizes the block diagram shown in the following figure as an accumulator:



- An accumulator construct is recognized anywhere across a block diagram, or within subsystems at lower levels.
- With the **Block reduction** option enabled, Simulink software creates a synthesized block, `Sum_synth_accum`. The synthesized block replaces the previous block diagram, resulting in a simple increment calculation.

Dead Code Elimination. Any blocks or signals in an *unused code path* are eliminated from generated code.

- The following conditions need to be met for a block to be considered part of an unused code path:
 - All signal paths for the block end with a block that does not execute. Examples of blocks that do not execute include Terminator blocks, disabled Assertion blocks, S-Function blocks configured for block reduction, and To Workspace blocks for which MAT-file logging is disabled for code generation.
 - No signal paths for the block include global signal storage downstream from the block.
- Tunable parameters do not prevent a block from being reduced by dead code elimination.
- Consider the signal paths in the following block diagram.



If you check **Block reduction**, Real-Time Workshop software responds to each signal path as follows:

For Signal Path...	Real-Time Workshop Software...
In1 to Out1	Always generates code because dead code elimination conditions are not met.
In2 to Terminator	Never generates code because dead code elimination conditions are met.
In3 to Scope	Generates code if MAT-file logging is enabled and eliminates code if MAT-file logging is disabled.

Command-Line Information

Parameter: BlockReduction

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended Settings

Application	Setting
Debugging	Off (for simulation and during development) No impact (for production code generation)
Traceability	Off
Efficiency	On
Safety precaution	Off

See Also

- Block Reduction
- Single-Tasking Execution
- Optimizing a Model for Code Generation
- Configuration Parameters Dialog Box
- Optimization Pane

Conditional input branch execution

Improve model execution when the model contains Switch and Multiport Switch blocks.

Settings

Default: On



On

Executes only the blocks required to compute the control input and the data input selected by the control input. This optimization speeds execution of code generated from the model. Limits to Switch block optimization:

- Only blocks with -1 (inherited) or inf (Constant) sample time can participate.
- Blocks with outputs flagged as test points cannot participate.
- No multirate block can participate.
- Blocks with states cannot participate.
- Only S-functions with option `SS_OPTION_CAN_BE_CALLED_CONDITIONALLY` set can participate.



Off

Executes all blocks driving the Switch block input ports at each time step.

Command-Line Information

Parameter: `ConditionallyExecuteInputs`

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended Settings

Application	Setting
Debugging	No impact

Application	Setting
Traceability	On
Efficiency	On
Safety precaution	Off

See Also

- Expression Folding
- Conditional Input Execution
- Optimizing a Model for Code Generation
- Configuration Parameters Dialog Box
- Optimization Pane

Inline parameters

Transform tunable parameters into constant values.

Settings

Default: Off



Enabling Inline parameters has the following effects:

- Real-Time Workshop software uses the numerical values of model parameters, instead of their symbolic names, in generated code.
- Reduces global RAM usage, because parameters are not declared in the global parameters structure.
- The **Configure** button becomes enabled. Clicking the **Configure** button opens the Model Parameter Configuration dialog box.



Uses symbolic names for model parameters in generated code.

Tips

- Simulink software allows you to override the **Inline parameters** option for parameters whose values are defined by variables in the MATLAB® workspace. To specify that such a parameter remain tunable, specify the parameter as global in the Model Configuration Parameters dialog box (see Model Parameter Configuration Dialog Box). To display the dialog box, click the adjacent **Configure** button.
- To tune a global parameter, change the value of the corresponding workspace variable and select **Update Diagram (Ctrl+D)** from the Simulink **Edit** menu.
- You cannot tune inlined parameters in code generated from a model. However, when simulating a model, you can tune an inlined parameter if its value derives from a workspace variable. For example, suppose that a model has a Gain block whose **Gain** parameter is inlined and equals a, where a is a variable defined in the model's workspace. When simulating the model, Simulink software disables the **Gain** parameter field, preventing you from using the block's dialog box to change the gain.

However, you can still tune the gain by changing the value of a at the MATLAB command line and updating the diagram.

- When a top-level model uses referenced models:
 - All referenced models must specify **Inline parameters** to be on.
 - The top-level model can specify **Inline parameters** to be on or off.

See Inline Parameter Requirements for more information.

Dependencies

This parameter enables:

- **Configure** button
- **Inline invariant signals**

Command-Line Information

Parameter: InlineParams

Type: string

Value: 'on' | 'off'

Default: 'off'

Recommended Settings

Application	Setting
Debugging	Off (for simulation and during development) On (for production code generation)
Traceability	On
Efficiency	On
Safety precaution	No impact

See Also

- Model Parameter Configuration Dialog Box

- Parameter Storage, Interfacing, and Tuning
- Model Referencing Inline Parameters
- Optimizing a Model for Code Generation
- Configuration Parameters Dialog Box
- Optimization Pane

Implement logic signals as boolean data (vs. double)

Controls the output data type of blocks that generate logic signals.

Settings

Default: On



On

Blocks that generate logic signals output a signal of boolean data type. This reduces the memory requirements of generated code.



Off

Blocks that generate logic signals output a signal of double data type. This ensures compatibility with models created by earlier versions of Simulink software.

Tips

- Setting this option **on** reduces the memory requirements of generated code, because a Boolean signal typically requires one byte of storage compared to eight bytes for a double signal.
- Setting this option **off** allows the current version of Simulink software to run models that were created by earlier versions of Simulink software that supported only signals of type double.
- This optimization affects the following blocks:
 - **Logical Operator block** – This parameter affects only those Logical Operator blocks whose **Output data type mode** parameter specifies Logical. If this parameter is selected, such blocks output a signal of boolean data type; otherwise, such blocks output a signal of double data type.
 - **Relational Operator block** – This parameter affects only those Relational Operator blocks whose **Output data type mode** parameter specifies Logical. If this parameter is selected, such blocks output a signal of boolean data type; otherwise, such blocks output a signal of double data type.
 - **Combinatorial Logic block** – If this parameter is selected, Combinatorial Logic blocks output a signal of boolean data type;

otherwise, they output a signal of double data type. See Combinatorial Logic in the *Simulink Reference* for an exception to this rule.

- **Hit Crossing block** – If this parameter is selected, Hit Crossing blocks output a signal of boolean data type; otherwise, they output a signal of double data type.

Dependencies

- This parameter is disabled for models created with a version of Simulink software that supports only signals of type double.

Command-Line Information

Parameter: BooleanDataType

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	On
Safety precaution	On

See Also

- Optimizing a Model for Code Generation
- Configuration Parameters Dialog Box
- Optimization Pane

Signal storage reuse

Reuse signal memory.

Settings

Default: On



Simulink software reuses memory buffers allocated to store block input and output signals, reducing the memory requirement of your real-time program.



Simulink software allocates a separate memory buffer for each block's outputs. This makes all block outputs global and unique, which in many cases significantly increases RAM and ROM usage.

Tips

- This option applies only to signals with storage class Auto.
- Turning this option off can substantially increase the amount of memory required to simulate large models.
- Disable this option if you need to:
 - Debug a C-MEX S-function
 - Use a Floating Scope or a Display block with the **Floating display** option selected to inspect signals in a model that you are debugging
- Simulink software opens an error dialog if **Signal storage reuse** is enabled and you attempt to use a Floating Scope or floating Display block to display a signal whose buffer has been reused.

Dependencies

This parameter enables:

- **Enable local block outputs**
- **Reuse block outputs**

- **Eliminate superfluous temporary variables (Expression folding)**

Command-Line Information

Parameter: OptimizeBlockIOStorage

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended Settings

Application	Setting
Debugging	Off
Traceability	Off
Efficiency	On
Safety precaution	No impact

See Also

- Signal Storage, Optimization, and Interfacing
- Optimizing a Model for Code Generation
- Configuration Parameters Dialog Box
- Optimization Pane

Application lifespan (days)

Specify how long (in days) an application that contains blocks depending on elapsed or absolute time should be able to execute before timer overflow.

Settings

Default: `inf`

Min: Must be greater than zero

Max: `inf`

Enter a positive (nonzero) scalar value (for example, 0.5) or `inf`.

If you are licensed for the Real-Time Workshop® Embedded Coder™ product and select an ERT target for your model, the default value for **Application lifespan (days)** is 1.

This parameter is ignored when you are operating your model in external mode, have **Mat-file logging** enabled, or have a continuous sample time because a 64 bit timer is required in these cases.

Tips

- Specifying a lifespan, along with the simulation step size, determines the data type used by blocks to store absolute time values.
- For simulation, setting this parameter to a value greater than the simulation time will ensure time does not overflow.
- Simulink software evaluates this parameter first against the model workspace. If this does not resolve the parameter, Simulink software then evaluates it against the base workspace.
- The Application lifespan also determines the word size used by timers in the generated code, which can lower RAM usage. For more information, see Timing Services in the Real-Time Workshop documentation.
- Application lifespan, when combined with the step size of each task, determines the data type used for integer absolute time for each task, as follows:
 - If your model does not require absolute time, this option affects neither simulation nor the generated code.

- If your model requires absolute time, this option optimizes the word size used for storing integer absolute time in generated code. This ensures that timers do not overflow within the lifespan you specify. If you set **Application lifespan** to `inf`, two `uint32` words are used.
- If your model contains fixed-point blocks that require absolute time, this option affects both simulation and generated code.

For example, using 64 bits to store timing data enables models with a step size of 0.001 microsecond (10E-09 seconds) to run for more than 500 years, which would rarely be required. To run a model with a step size of one millisecond (0.001 seconds) for one day would require a 32-bit timer (but it could continue running for 49 days).

- A timer will allocate 64 bits of memory if you specify a value of `inf`.
- To minimize the amount of RAM used by time counters, specify a lifespan no longer than necessary.
- Must be the same for top and referenced models.
- Optimize the size of counters used to compute absolute and elapsed time.

Command-Line Information

Parameter: LifeSpan

Type: string

Value: positive (nonzero) scalar value or `inf`

Default: `'inf'`

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	Finite value
Safety precaution	<code>inf</code>

See Also

- [Timing Services](#)
- [Using Timers in Asynchronous Tasks](#)
- [Optimizing a Model for Code Generation](#)
- [Configuration Parameters Dialog Box](#)
- [Optimization Pane](#)

Parameter structure

Control how parameter data is generated for reusable subsystems.

Settings

Default: NonHierarchical

Hierarchical

Generates a separate header file, defining an independent parameter structure, for each subsystem that meets the following conditions:

- The subsystem's **Real-Time Workshop system code** parameter is set to Reusable function.
- The subsystem does not violate any code reuse limitations.
- The subsystem does not access parameters other than its own (such as parameters of the root-level model).

Each subsystem parameter structure is referenced as a substructure of the root-level parameter data structure, creating a structure hierarchy.

NonHierarchical

Generates a single, flat parameter data structure. Subsystem parameters are defined as fields within the structure. A nonhierarchical data structure can reduce compiler padding between word boundaries, producing more efficient compiled code.

Dependency

- This parameter requires a Real-Time Workshop license.
- **Inline parameters** must be enabled.

Command-Line Information

Parameter: InlinedParameterPlacement

Type: string

Value: 'Hierarchical' | 'NonHierarchical'

Default: 'NonHierarchical'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	Hierarchical
Efficiency	NonHierarchical
Safety precaution	No impact

See Also

- Nonvirtual Subsystem Code Generation
- Optimizing a Model for Code Generation
- Configuration Parameters Dialog Box
- Optimization Pane

Enable local block outputs

Specify whether block signals are declared locally or globally.

Settings

Default: On

- On**
Block signals are declared locally in functions.
- Off**
Block signals are declared globally.

Tips

- If it is not possible to declare an output as a local variable, the generated code declares the output as a global variable.
- If you are constrained by limited stack space, you can turn **Enable local block outputs** off and still benefit from memory reuse.

Dependency

- This parameter requires a Real-Time Workshop license.
- This parameter is enabled by **Signal storage reuse**.

Command-Line Information

Parameter: LocalBlockOutputs
Type: string
Value: 'on' | 'off'
Default: 'on'

Recommended Settings

Application	Setting
Debugging	Off

Application	Setting
Traceability	No impact
Efficiency	On
Safety precaution	No impact

See Also

- Signal Storage, Optimization, and Interfacing
- Signals with Auto Storage Class
- Optimizing a Model for Code Generation
- Configuration Parameters Dialog Box
- Optimization Pane

Ignore integer downcasts in folded expressions

Specify how Real-Time Workshop software handles casting of intermediate variables in mixed-bit systems.

Settings

Default: Off (GUI), 'on' (command-line)



On

Real-Time Workshop software collapses block computations into a single expression, avoiding casts of intermediate variables, improving efficiency. Check this parameter if:

- You are concerned with generating the least amount of code possible.
- Code generation and simulation results do not need to match.



Off (default)

The results of 8- and 16-bit integer expressions are explicitly downcast.

Tips

- To ensure consistency between simulation and code generation, the results of 8 and 16-bit integer expressions must be explicitly downcast. Selecting this option improves code efficiency by avoiding casts of intermediate variables.
- Expressions involving 8- and 16-bit arithmetic are less likely to overflow in code than they are in simulation. Therefore, it is good practice to turn off **Ignore integer downcasts in folded expressions** for safety, to ensure that answers obtained from generated code are consistent with simulation results.

Dependency

This parameter requires a Real-Time Workshop license.

Command-Line Information

Parameter: EnforceIntegerDowncast

Type: string

Value: 'on' | 'off'

Default: 'on'

Note The command-line values are reverse of the settings values. Therefore, 'on' in the command line corresponds to the description of “Off” in the settings section, and 'off' in the command line corresponds to the description of “On” in the settings section.

Recommended Settings

Application	Setting
Debugging	Off
Traceability	No impact
Efficiency	On
Safety precaution	Off

See Also

- Optimizing a Model for Code Generation
- Expression Folding
- Configuration Parameters Dialog Box
- Optimization Pane

Eliminate superfluous temporary variables (Expression folding)

Collapse block computations into single expressions.

Settings

Default: On

On

- Enables expression folding.
- Eliminates temporary variables, incorporating the information into the main code statement.
- Improves code readability and efficiency.

Off

Disables expression folding.

Dependency

- This parameter requires a Real-Time Workshop license.
- This parameter is enabled by **Signal storage reuse**.

Command-Line Information

Parameter: ExpressionFolding

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended Settings

Application	Setting
Debugging	Off

Application	Setting
Traceability	No impact (for simulation and during development) Off (for production code generation)
Efficiency	On
Safety precaution	No impact

See Also

- Expression Folding
- Optimizing a Model for Code Generation
- Configuration Parameters Dialog Box
- Optimization Pane

Reuse block outputs

Specify whether Real-Time Workshop software reuses signal memory.

Settings

Default: On

On (default)

- Real-Time Workshop software reuses signal memory whenever possible, reducing stack size where signals are being buffered in local variables.
- Selecting this parameter trades code traceability for code efficiency.

Off

Signals are stored in unique locations.

Dependency

- This parameter requires a Real-Time Workshop license.
- This parameter is enabled by **Signal storage reuse**.

Command-Line Information

Parameter: BufferReuse

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended Settings

Application	Setting
Debugging	Off
Traceability	Off
Efficiency	On
Safety precaution	No impact

See Also

- [Signal Storage, Optimization, and Interfacing](#)
- [Signals with Auto Storage Class](#)
- [Optimizing a Model for Code Generation](#)
- [Configuration Parameters Dialog Box](#)
- [Optimization Pane](#)

Inline invariant signals

Transform symbolic names of invariant signals into constant values.

Settings

Default: Off

On

Real-Time Workshop software uses the numerical values of model parameters, instead of their symbolic names, in generated code. An invariant signal is not inlined if it is nonscalar, complex, or the block inport the signal is attached to takes the address of the signal.

Off

Uses symbolic names of model parameters in generated code.

Dependency

- This parameter requires a Real-Time Workshop license.
- This parameter is enabled by **Inline parameters**.

Command-Line Information

Parameter: InlineInvariantSignals

Type: string

Value: 'on' | 'off'

Default: 'off'

Recommended Settings

Application	Setting
Debugging	Off
Traceability	Off
Efficiency	On
Safety precaution	No impact

See Also

Inlining Invariant Signals

Loop unrolling threshold

Specify the minimum signal or parameter width for which a for loop is generated.

Settings

Default: 5

Specify the array size at which the code generator begins to use a for loop instead of separate assignment statements to assign values to the elements of a signal or parameter array.

When there are perfectly nested loops, the code generator uses a for loop if the product of the loop counts for all loops in the perfect loop nest is greater than or equal to the threshold.

Dependency

This parameter requires a Real-Time Workshop license.

Command-Line Information

Parameter: RollThreshold

Type: string

Value: any valid value

Default: '5'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	>0
Safety precaution	>1

See Also

- [Configuring a Loop Unrolling Threshold](#)
- [Real-Time Workshop Target Language Compiler](#)

Remove root level I/O zero initialization

Specify whether to generate initialization code for root-level inports and outports set to zero.

Settings

Default: Off (GUI), 'on' (command-line)



On

Does not generate initialization code for root-level inports and outports set to zero.



Off

Generates initialization code for all root-level inports and outports. You should use the default:

- To ensure that memory allocated for C MEX S-function wrappers is initialized to zero
- For safety-critical applications that require that all internal and external data be initialized to zero

Dependencies

- This parameter requires a Real-Time Workshop license.
- This parameter only appears for ERT-based targets.

Command-Line Information

Parameter: ZeroExternalMemoryAtStartup

Type: string

Value: 'off' | 'on'

Default: 'on'

Note The command-line values are reverse of the settings values. Therefore, 'on' in the command line corresponds to the description of “Off” in the settings section, and 'off' in the command line corresponds to the description of “On” in the settings section.

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	On
Safety precaution	Off

See Also

- [Optimizing a Model for Code Generation](#)
- [Configuration Parameters Dialog Box](#)
- [Optimization Pane](#)

Use memset to initialize floats and doubles to 0.0

Specify whether to generate code that explicitly initializes floating-point data to 0.0.

Settings

Default: Off

On

Uses memset to clear internal storage for floating-point data to integer bit pattern 0 (all bits 0), regardless of type. An example of a case for selecting this option is to gain compiler efficiency when the compiler and target CPU both represent floating-point zero with the integer bit pattern 0.

Off

Generates extra code to explicitly initialize storage for data of types float and double to 0.0. The resulting code is slightly less efficient than code generated when you select the option.

You should not select this option if you need to ensure that memory allocated for C MEX S-function wrappers is initialized to zero.

Dependencies

- This parameter requires a Real-Time Workshop Embedded Coder license.
- This parameter only appears for ERT-based targets.

Command-Line Information

Parameter: InitFltsAndDblsToZero

Type: string

Value: 'on' | 'off'

Default: 'off'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	On
Safety precaution	No impact

See Also

- [Optimizing a Model for Code Generation](#)
- [Configuration Parameters Dialog Box](#)
- [Optimization Pane](#)

Remove internal state zero initialization

Specify whether to generate initialization code for internal work structures, such as block states and block outputs, to zero.

Settings

Default: Off (GUI), 'on' (command-line)



On

Does not generate code that initializes internal work structures to zero. An example of when you might select this option is to test the behavior of a design during warm boot—a restart without full system reinitialization.

Selecting this option does not guarantee that memory is in a known state each time the generated code begins execution. When you run a model or generated S-function multiple times, each run can produce a different answer.

If want to get the same answer on every run from a generated S-function, enter the command `clear SFcnNam` or `clear mex` in the MATLAB Command Window before each run.



Off

Generates code that initializes internal work structures to zero. You should use the default:

- To ensure that memory allocated for C MEX S-function wrappers is initialized to zero
- For safety critical applications that require that all internal and external data be initialized to zero

Dependencies

- This parameter requires a Real-Time Workshop Embedded Coder license.
- This parameter only appears for ERT-based targets.

Command-Line Information

Parameter: ZeroInternalMemoryAtStartup

Type: string

Value: 'off' | 'on'

Default: 'on'

Note The command-line values are reverse of the settings values. Therefore, 'on' in the command line corresponds to the description of “Off” in the settings section, and 'off' in the command line corresponds to the description of “On” in the settings section.

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	On
Safety precaution	Off

See Also

- [Optimizing a Model for Code Generation](#)
- [Configuration Parameters Dialog Box](#)
- [Optimization Pane](#)

Optimize initialization code for model reference

Specify whether to generate initialization code for blocks that have states.

Settings

Default: on



Suppresses generation of initialization code for blocks that have states unless the blocks are in a system that can reset its states, such as an enabled subsystem. This results in more efficient code, but requires that you not refer to the model from a Model block that resides in a system that resets states. If you violate this constraint, Simulink software reports an error, in which case you can disable this optimization.



Generates initialization code for all blocks that have states. Disable this option if the current model includes a subsystem that resets states, such as an enabled subsystem, and the model is referred to from another model with a Model block.

Dependencies

- This parameter requires a Real-Time Workshop Embedded Coder license.
- This parameter only appears for ERT-based targets.
- You must disable this option if your model includes enabled subsystem and model is referred to from another model with Model block.

Command-Line Information

Parameter: OptimizeModelRefInitCode

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	On
Safety precaution	No impact

See Also

- [Optimizing a Model for Code Generation](#)
- [Configuration Parameters Dialog Box](#)
- [Optimization Pane](#)

Remove code from floating-point to integer conversions that wraps out-of-range values

Remove wrapping code that handles out-of-range floating-point to integer conversion results.

Settings

Default: Off

On

Removes code that ensures the execution of the generated code produces the same results as simulation when out-of-range conversions occur. Select this parameter if code efficiency is critical to your application and the following conditions are true for at least one block in the model:

- Computing the block's outputs or parameters involves converting floating-point data to integer or fixed-point data.
- The block's **Saturate on integer overflow** parameter is disabled.

Off

Out-of-range values simulation and generated code results match. The generated code is larger than when this parameter is checked.

Tips

- Enabling this parameter reduces the size and increases the speed of the generated code at the cost of potentially producing results that do not match simulation in the case of out-of-range values.
- Enabling this parameter affects code generation results only for out-of-range values and cannot cause code generation results to differ from simulation results for in-range values.
- The code generator uses the `fmod` function to handle out-of-range conversion results.
- The following code fragment shows the code generated for a conversion with this option disabled.

```
_fixptlowering0 = (rtb_Switch[i1] + 9.0) / 0.09375;  
_fixptlowering1 = fmod(_fixptlowering0 >= 0.0 ?
```

```

floor(_fixptlowering0) :
    ceil(_fixptlowering0), 4.2949672960000000E+009);
if(_fixptlowering1 < -2.1474836480000000E+009) {
    _fixptlowering1 += 4.2949672960000000E+009;
} else if(_fixptlowering1 >= 2.1474836480000000E+009) {
    _fixptlowering1 -= 4.2949672960000000E+009;
}
cg_in_0_20_0[i1] = (int32_T)_fixptlowering1;

```

- The code generated for the conversion when you select this optimization follows:

```
cg_in_0_20_0[i1] = (int32_T)((rtb_Switch[i1] + 9.0) / 0.09375);
```

Dependency

This parameter requires a Real-Time Workshop license.

Command-Line Information

Parameter: EfficientFloat2IntCast

Type: string

Value: 'on' | 'off'

Default: 'off'

Recommended Settings

Application	Setting
Debugging	Off
Traceability	Off
Efficiency	On
Safety precaution	Off (for simulation and during development) On (for production code generation)

See Also

- [Optimizing a Model for Code Generation](#)
- [Configuration Parameters Dialog Box](#)
- [Optimization Pane](#)

Remove code that protects against division arithmetic exceptions

Specify whether to generate code that guards against division by zero for fixed-point data.

Settings

Default: on



On

Does not generate code that guards against division by zero for fixed-point data. When you select this option, simulation results and results from generated code might not be in bit-for-bit agreement.



Off

Generates code that guards against division by zero for fixed-point data.

Dependencies

- This parameter requires a Real-Time Workshop Embedded Coder license.
- This parameter only appears for ERT-based targets.

Command-Line Information

Parameter: NoFixptDivByZeroProtection

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	On
Safety precaution	Off

See Also

- [Optimizing a Model for Code Generation](#)
- [Configuration Parameters Dialog Box](#)
- [Optimization Pane](#)

Use bitsets for storing state configuration

Use bitsets to reduce the amount of memory required to store state configuration variables.

Settings

Default: Off

On

Stores state configuration variables in bitsets. Potentially reduces the amount of memory required to store the variables. Potentially requires more instructions to access state configuration, which can result in less optimal code.

Off

Stores state configuration variables in unsigned bytes. Potentially increases the amount of memory required to store the variables. Potentially requires fewer instructions to access state configuration, which can result in more optimal code.

Tips

- Enabling this parameter can significantly reduce the amount of memory required to store the variables. However, it can increase the amount of memory required to store target code if the target processor does not include instructions for manipulating bitsets.
- Enable this setting for Stateflow charts that have a large number of sibling states at a given level of hierarchy.
- Disable this setting for Stateflow charts with a small number of sibling states at a given level of hierarchy.

Command-Line Information

Parameter: statebitsets

Type: Boolean

Value: 1 | 0

Default: 0

Recommended Settings

Application	Setting
Debugging	Off
Traceability	Off
Efficiency	Off
Safety precaution	No impact

See Also

- [Configuring Real-Time Workshop for Stateflow](#)
- [Configuration Parameters Dialog Box](#)
- [Optimization Pane](#)

Use bitsets for storing boolean data

Use bitsets to reduce the amount of memory required to store Boolean data.

Settings

Default: Off

On

Stores Boolean data in bitsets. Potentially reduces the amount of memory required to store the data. Potentially requires more instructions to access the data, which can result in less optimal code.

Off

Stores Boolean data in unsigned bytes. Potentially increases the amount of memory required to store the data. Potentially requires fewer instructions to access the data, which can result in more optimal code.

Tips

- Enable this setting for Stateflow charts that reference Boolean data infrequently.
- Disable this setting for Stateflow charts that reference Boolean data frequently.

Command-Line Information

Parameter: databitsets

Type: Boolean

Value: 1 | 0

Default: 0

Recommended Settings

Application	Setting
Debugging	Off
Traceability	Off

Application	Setting
Efficiency	Off
Safety precaution	No impact

See Also

- [Configuring Real-Time Workshop for Stateflow](#)
- [Configuration Parameters Dialog Box](#)
- [Optimization Pane](#)

Minimize array reads using temporary variables

Minimize global array read operations by using temporary variables. In certain microprocessors, global array read operations are more expensive than accessing a temporary variable on the stack.

Settings

Default: Off

On
Uses temporary variables to access array data.

Off
Uses global memory to access array data.

Tips

- Enable this setting if your target tolerates large stack frames.
- In certain microprocessors, global array read operations are more expensive than accessing a temporary variable on stack. Using this option minimizes array reads by using temporary variables when possible.

For example, the generated code

```
a[i] = foo();
if(a[i]<10 && a[i]>1) {
    y = a[i]+5;
}else{
    z = a[i];
}
```

now becomes

```
a[i] = foo();
temp = a[i];
if(temp<10 && temp>1) {
    y = temp+5;
}else{
    z = temp;
}
```

Command-Line Information

Parameter: redundantloadelimination

Type: Boolean

Value: 1 | 0

Default: 0

Recommended Settings

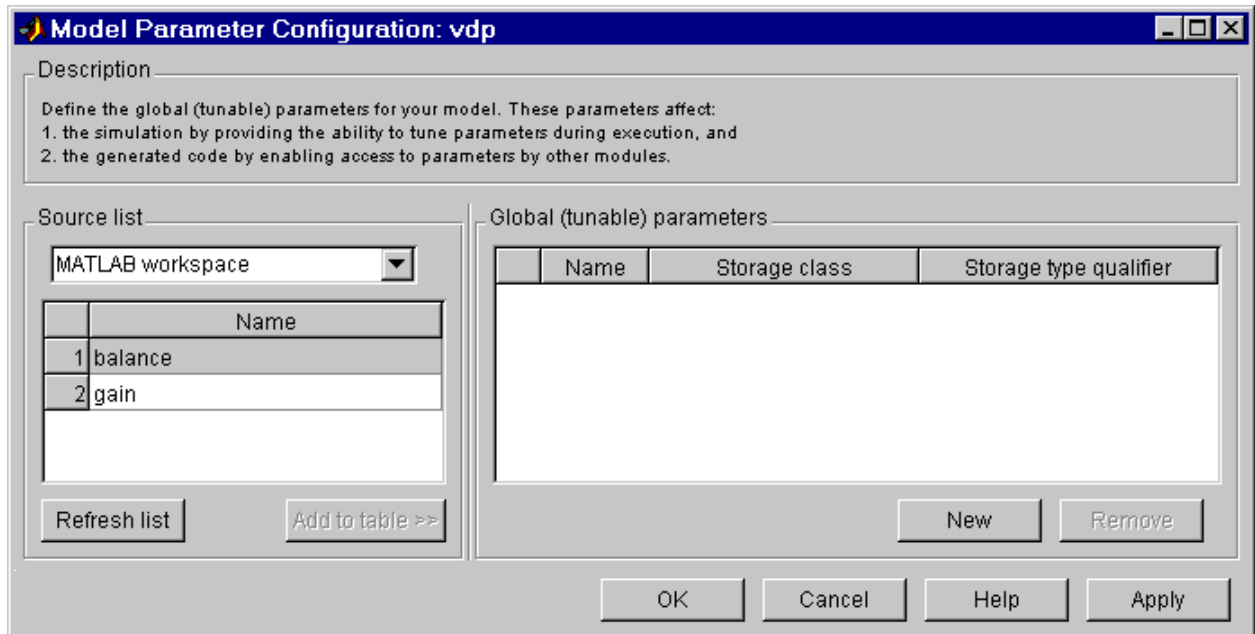
Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	On
Safety precaution	No impact

See Also

- [Configuring Real-Time Workshop for Stateflow](#)
- [Configuration Parameters Dialog Box](#)
- [Optimization Pane](#)

Model Parameter Configuration Dialog Box

The **Model Parameter Configuration** dialog box allows you to override the **Inline parameters** option (see Inline parameters) for selected parameters.



Note Simulink software ignores the settings of this dialog box if a model contains references to other models. However, you can still tune parameters of such models, using `Simulink.Parameter` objects (see “Inline Parameter Requirements” for more information).

The dialog box has the following controls.

Source list

Displays a list of workspace variables. The options are:

- MATLAB workspace

Lists all variables in the MATLAB workspace that have numeric values.

- Referenced workspace variables

Lists only those variables referenced by the model.

Refresh list

Updates the source list. Click this button if you have added a variable to the workspace since the last time the list was displayed.

Add to table

Adds the variables selected in the source list to the adjacent table of tunable parameters.

New

Defines a new parameter and adds it to the list of tunable parameters. Use this button to create tunable parameters that are not yet defined in the MATLAB workspace.

Note This option does not create the corresponding variable in the MATLAB workspace. You must create the variable yourself.

Storage class

Used for code generation. See the *Real-Time Workshop User's Guide* for more information.

Storage type qualifier

Used for code generation. See the *Real-Time Workshop User's Guide* for more information.

Compiler optimization level

Sets the degree of optimization used by the compiler when generating code for acceleration.

Settings

Default: Optimizations off (faster builds)

Optimizations off (faster builds)

Specifies the compiler not to optimize code. This results in faster build times.

Optimizations on (faster runs)

Specifies the compiler to generate optimized code. The generated code will run faster, but the model build will take longer than if optimizations are off.

Tips

- The default Optimizations off is a good choice for most models. This quickly produces code that can be used with acceleration.
- Set Optimizations on to optimize your code. The fast running code produced by optimization can be advantageous if you will repeatedly run your model with the accelerator.

Command-Line Information

Parameter: SimCompilerOptimization

Type: string

Value: 'on' | 'off'

Default: 'off'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	No impact

See Also

- “Accelerating Models”
- “Interacting with the Acceleration Modes Programmatically”
- “Customizing the Acceleration Build Process”

Verbose accelerator builds

Select the amount of information displayed during code generation.

Settings

Default: Off

- Off**
Display limited amount of information during the code generation process.
- On**
Display progress information during code generation, and show the compiler options in use.

Command-Line Information

Parameter: AccelVerboseBuild

Type: string

Value: 'on' | 'off'

Default: 'off'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

For more information about AccelVerboseBuild , see “Controlling Information During Code Generation”.

Diagnostics Pane: Solver

Solver	
Algebraic loop:	warning
Minimize algebraic loop:	warning
Block priority violation:	warning
Min step size violation:	warning
Sample hit time adjusting:	none
Consecutive zero crossings violation:	error
Unspecified inheritability of sample time:	warning
Solver data inconsistency:	none
Automatic solver parameter selection:	warning
Extraneous discrete derivative signals:	error
State name clash:	warning

In this section...

“Solver Diagnostics Overview” on page 1-169

“Algebraic loop” on page 1-170

“Minimize algebraic loop” on page 1-172

“Block priority violation” on page 1-174

“Min step size violation” on page 1-176

“Sample hit time adjusting” on page 1-178

“Consecutive zero crossings violation” on page 1-180

“Unspecified inheritability of sample time” on page 1-182

“Solver data inconsistency” on page 1-184

“Automatic solver parameter selection” on page 1-186

“Extraneous discrete derivative signals” on page 1-188

“State name clash” on page 1-190

Solver Diagnostics Overview

Specify what diagnostic actions Simulink® software should take, if any, when it detects an abnormal condition with the solver.

Configuration

Set the parameters displayed.

Tips

- The options are typically to do nothing or to display a warning or an error message.
- A warning does not terminate a simulation, but an error does.

See Also

- Diagnosing Simulation Errors
- Sample Time Diagnostics
- Data Validity Diagnostics
- Type Conversion Diagnostics
- Connectivity Diagnostics
- Compatibility Diagnostics
- Model Referencing Diagnostics
- Saving Diagnostics
- Configuration Parameters Dialog Box
- Diagnostics Pane: Solver

Algebraic loop

Select the diagnostic action to take if Simulink software detects an algebraic loop while compiling the model.

Settings

Default: warning

none

Simulink software does not check for algebraic loops.

warning

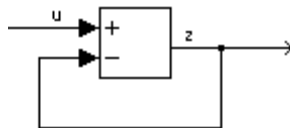
When Simulink software detects an algebraic loop, it displays a warning.

error

When Simulink software detects an algebraic loop, it terminates the simulation, displays an error message, and highlights the portion of the block diagram that comprises the loop.

Tips

- An *algebraic loop* generally occurs when an input port with direct feedthrough is driven by the output of the same block, either directly, or by a feedback path through other blocks with direct feedthrough. An example of an algebraic loop is this simple scalar loop.



- When a model contains an algebraic loop, Simulink software calls a loop solving routine at each time step. The loop solver performs iterations to determine the solution to the problem (if it can). As a result, models with algebraic loops run slower than models without them.
- Use the error option to highlight algebraic loops when you simulate a model. This causes Simulink software to display an error dialog (the Diagnostics Viewer) and recolor portions of the diagram that represent the algebraic loops that it detects. Simulink software uses red to color the

blocks and lines that constitute the loops. Closing the error dialog restores the diagram to its original colors.

- See Algebraic Loops for more information.

Command-Line Information

Parameter: AlgebraicLoopMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	error
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Algebraic Loops
- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Solver

Minimize algebraic loop

Select the diagnostic action to take if algebraic loop minimization cannot be performed for a subsystem because an input port of that subsystem has direct feedthrough.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- If the port is involved in an algebraic loop, Simulink software can remove the loop only if at least one other input port in the loop lacks direct feedthrough.
- Simulink software cannot minimize algebraic loops containing signals designated as test points (see Working with Test Points).

Command-Line Information

Parameter: ArtificialAlgebraicLoopMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact

Application	Setting
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Algebraic Loops
- Diagnosing Simulation Errors
- Working with Test Points
- Configuration Parameters Dialog Box
- Diagnostics Pane: Solver

Block priority violation

Select the diagnostic action to take if Simulink software detects a block priority specification error.

Settings

Default: warning

warning

When Simulink software detects a block priority specification error, it displays a warning.

error

When Simulink software detects a block priority specification error, it terminates the simulation and displays an error message.

Tips

- Simulink software allows you to assign update priorities to blocks. Simulink software executes the output methods of higher priority blocks before those of lower priority blocks.
- Simulink software honors the block priorities that you specify only if they are consistent with the Simulink block sorting algorithm. If Simulink software is unable to honor a user specified block priority, it generates a block priority specification error.

Command-Line Information

Parameter: BlockPriorityViolationMsg

Type: string

Value: 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact

Application	Setting
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Controlling and Displaying the Sorted Order
- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Solver

Min step size violation

Select the diagnostic action to take if Simulink software detects that the next simulation step is smaller than the minimum step size specified for the model.

Settings

Default: warning

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- A minimum step size violation can occur if the specified error tolerance for the model requires a step size smaller than the specified minimum step size. See [Min step size](#) and [Maximum order](#) for more information.
- Simulink software allows you to specify the maximum number of consecutive minimum step size violations permitted (see [Number of consecutive min step size violations allowed](#)).

Command-Line Information

Parameter: MinStepSizeMsg

Type: string

Value: 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	No impact

See Also

- Min step size
- Maximum order
- Number of consecutive min step size violations allowed
- Determining Step Size for Discrete Systems
- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Solver

Sample hit time adjusting

Select the diagnostic action to take if Simulink software makes a minor adjustment to a sample hit time while running the model.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

Tips

- Simulink software might change a sample hit time if that hit time is close to the hit time for another task. If Simulink software considers the difference to be due only to numerical errors (for example, precision issues or round-off errors), it changes the sample hits of the faster task or tasks to exactly match the time of the slowest task that has that hit.
- Over time, these sample hit changes might cause a discrepancy between the numerical simulation results and the actual theoretical results.
- When this option is set to warning, the MATLAB® Command Window displays a warning like the following when Simulink software detects a change in the sample hit time:

Warning: Timing engine warning: Changing the hit time for ...

Command-Line Information

Parameter: TimeAdjustmentMsg

Type: string

Value: 'none' | 'warning'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Solver

Consecutive zero crossings violation

Select the diagnostic action to take when Simulink software detects that the number of consecutive zero crossings exceeds the specified maximum.

Settings

Default: error

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- If you select warning or error, Simulink software reports the current simulation time, the number of consecutive zero crossings counted, and the type and name of the block in which Simulink software detected the zero crossings.
- For more information, see *Preventing Excessive Zero Crossings*.

Command-Line Information

Parameter: MaxConsecutiveZCMsg

Type: string

Value: 'none' | 'warning'

Default: 'error'

Recommended Settings

Application	Setting
Debugging	No impact

Application	Setting
Traceability	No impact
Efficiency	No impact
Safety precaution	warning or error

See Also

- Zero-Crossing Detection
- Zero-Crossing Control
- Number of consecutive zero crossings allowed
- Consecutive zero crossings relative tolerance
- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Solver

Unspecified inheritability of sample time

Select the diagnostic action to take if this model contains S-functions that do not specify whether they preclude this model from inheriting their sample times from a parent model.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- Not specifying an inheritance rule may lead to incorrect simulation results.
- Simulink software checks for this condition only if the solver used to simulate this model is a fixed-step discrete solver and the periodic sample time constraint for the solver is set to ensure sample time independence
- For more information, see Periodic sample time constraint.

Command-Line Information

Parameter: UnknownTsInhSupMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact

Application	Setting
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Periodic sample time constraint
- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Solver

Solver data inconsistency

Select the diagnostic action to take if Simulink software detects S-functions that have continuous sample times, but do not produce consistent results when executed multiple times.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- Consistency checking can cause a significant decrease in performance (up to 40%).
- Consistency checking is a debugging tool that validates certain assumptions made by Simulink ODE solvers. Use this option to:
 - Validate your S-functions and ensure that they adhere to the same rules as Simulink built-in blocks.
 - Determine the cause of unexpected simulation results.
 - Ensure that blocks produce constant output when called with a given value of t (time).
- Simulink software saves (caches) output, zero-crossing, derivative, and state values from one time step for use in the next time step. The value at the end of a time step can generally be reused at the start of the next time step. Solvers, particularly stiff solvers such as ode23s and ode15s, take advantage of this to avoid redundant calculations. While calculating the Jacobian matrix, a stiff solver can call a block's output functions many times at the same value of t .

- When consistency checking is enabled, Simulink software recomputes the appropriate values and compares them to the cached values. If the values are not the same, a consistency error occurs. Simulink software compares computed values for these quantities:
 - Outputs
 - Zero crossings
 - Derivatives
 - States

Command-Line Information

Parameter: ConsistencyChecking

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	warning
Traceability	No impact
Efficiency	none
Safety precaution	No impact

See Also

- Diagnosing Simulation Errors
- Choosing a Solver
- Configuration Parameters Dialog Box
- Diagnostics Pane: Solver

Automatic solver parameter selection

Select the diagnostic action to take if Simulink software changes a solver parameter setting.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

When enabled, this option notifies you if:

- Simulink software changes a user-modified parameter to make it consistent with other model settings.
- Simulink software automatically selects solver parameters for the model, such as `FixedStepSize`.

For example, if you simulate a discrete model that specifies a continuous solver, Simulink software changes the solver type to discrete and displays a warning about this change at the MATLAB command line.

Command-Line Information

Parameter: `SolverPrmCheckMsg`

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors
- Choosing a Solver
- Configuration Parameters Dialog Box
- Diagnostics Pane: Solver

Extraneous discrete derivative signals

Select the diagnostic action to take when a discrete signal appears to pass through a Model block to the input of a block with continuous states.

Settings

Default: error

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- This error can occur if a discrete signal passes through a Model block to the input of a block with continuous states, such as an Integrator block. In this case, Simulink software cannot determine with certainty the minimum rate at which it needs to reset the solver to solve this model accurately.
- If this diagnostic is set to none or warning, Simulink software resets the solver whenever the value of the discrete signal changes. This ensures accurate simulation of the model if the discrete signal is the source of the signal entering the block with continuous states. However, if the discrete signal is not the source of the signal entering the block with continuous states, resetting the solver at the rate the discrete signal changes can lead to the solver being reset more frequently than necessary, slowing down the simulation.
- If this diagnostic is set to error, Simulink software halts when compiling this model and displays an error.

Command-Line Information

Parameter: ModelReferenceExtrNoncontSigs

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'error'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Diagnosing Simulation Errors
- Choosing a Solver
- Configuration Parameters Dialog Box
- Diagnostics Pane: Solver

State name clash

Select the diagnostic action to take when a name is used for more than one state in the model.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

Tips

- This diagnostic applies for continuous and discrete states during simulation.
- This diagnostic applies only if you save states to the MATLAB workspace using the format **Structure** or **Structure with time**. If you do not save states in structure format, the state names are not used, and therefore the diagnostic will not warn you about a naming conflict.

Command-Line Information

Parameter: StateNameClashWarn

Type: string

Value: 'none' | 'warning'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	No impact

See Also

- Diagnosing Simulation Errors
- Data Import/Export Pane
- Save to workspace: States
- Save options: Format
- Exporting Data to the MATLAB Workspace
- Configuration Parameters Dialog Box
- Diagnostics Pane: Solver

Diagnostics Pane: Sample Time

Sample Time	
Source block specifies -1 sample time:	warning
Discrete used as continuous:	warning
Multitask rate transition:	error
Single task rate transition:	none
Multitask conditionally executed subsystem:	error
Tasks with equal priority:	warning
Enforce sample times specified by Signal Specification blocks:	warning

In this section...

“Sample Time Diagnostics Overview” on page 1-193

“Source block specifies -1 sample time” on page 1-194

“Discrete used as continuous” on page 1-196

“Multitask rate transition” on page 1-198

“Single task rate transition” on page 1-200

“Multitask conditionally executed subsystem” on page 1-202

“Tasks with equal priority” on page 1-204

“Enforce sample times specified by Signal Specification blocks” on page 1-206

Sample Time Diagnostics Overview

Specify what diagnostic actions Simulink® software should take, if any, when it detects a compilation error related to model sample times.

Configuration

Set the parameters displayed.

Tips

- The options are typically to do nothing or to display a warning or an error message.
- A warning does not terminate a simulation, but an error does.

See Also

- Diagnosing Simulation Errors
- Solver Diagnostics
- Data Validity Diagnostics
- Type Conversion Diagnostics
- Connectivity Diagnostics
- Compatibility Diagnostics
- Model Referencing Diagnostics
- Saving Diagnostics
- Configuration Parameters Dialog Box
- Diagnostics Pane: Sample Time

Source block specifies -1 sample time

Select the diagnostic action to take if a source block (such as a Sine Wave block) specifies a sample time of -1.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- The Random Source block does not obey this parameter. If its **Sample time** parameter is set to -1, the Random Source block inherits its sample time from its output port and never produces warnings or errors.
- Some Communications Blockset™ blocks internally inherit sample times, which can be a useful and valid modeling technique. Set this parameter to none for these types of models.

Command-Line Information

Parameter: InheritedTsInSrcMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact

Application	Setting
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Sample Time

Discrete used as continuous

Select the diagnostic action to take if a discrete block (such as the Unit Delay block), inherits a continuous sample time from the block connected to its input.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Command-Line Information

Parameter: DiscreteInheritContinuousMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors

- Configuration Parameters Dialog Box
- Diagnostics Pane: Sample Time

Multitask rate transition

Select the diagnostic action to take if an invalid rate transition occurred between two blocks operating in multitasking mode.

Settings

Default: error

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- This parameter allows you to adjust error checking for sample rate transitions between blocks that operate at different sample rates.
- Use this option for models of real-time multitasking systems to ensure detection of illegal rate transitions between tasks that can result in a task's output not being available when needed by another task. You can then use Rate Transition blocks to eliminate such illegal rate transitions from the model.

Command-Line Information

Parameter: MultiTaskRateTransMsg

Type: string

Value: 'warning' | 'error'

Default: 'error'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	error

See Also

- Rate Transition block
- Model Execution and Rate Transitions
- Single-Tasking and Multitasking Execution Modes
- Sample Rate Transitions
- Tasking mode for periodic sample times
- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Sample Time

Single task rate transition

Select the diagnostic action to take if a rate transition occurred between two blocks operating in single-tasking mode.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- This parameter allows you to adjust error checking for sample rate transitions between blocks that operate at different sample rates.
- Use this parameter when you are modeling a single-tasking system. In such systems, task synchronization is not an issue.

Command-Line Information

Parameter: SingleTaskRateTransMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	none or error

See Also

- Rate Transition block
- Model Execution and Rate Transitions
- Single-Tasking and Multitasking Execution Modes
- Sample Rate Transitions
- Tasking mode for periodic sample times
- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Sample Time

Multitask conditionally executed subsystem

Select the diagnostic action to take if Simulink software detects a subsystem that may cause data corruption or non-deterministic behavior.

Settings

Default: error

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- These types of subsystems can be caused by either of the following conditions:
 - Your model uses multitasking solver mode and it contains an enabled subsystem that operates at multiple rates.
 - Your model contains a conditionally executed subsystem that can reset its states and that itself contains an asynchronous subsystem.These types of subsystems can cause corrupted data or non-deterministic behavior in a real-time system using code generated from the model.
- For models that use multitasking solver mode and contain an enabled subsystem that operates at multiple rates, consider using single-tasking solver mode or using a single-rate enabled subsystem instead.
- For models that contain a conditionally executed subsystem that can reset its states and that itself contains an asynchronous subsystem. consider moving the asynchronous subsystem outside the conditionally executed subsystem.

Command-Line Information

Parameter: MultiTaskCondExecSysMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'error'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Tasking mode for periodic sample times
- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Sample Time

Tasks with equal priority

Select the diagnostic action to take if Simulink software detects two tasks with equal priority that can preempt each other in the target system.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- This condition can occur when one asynchronous task of the target represented by this model has the same priority as another of the target's asynchronous tasks.
- This option must be set to Error if the target allows tasks having the same priority to preempt each other.

Command-Line Information

Parameter: TasksWithSamePriorityMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact

Application	Setting
Traceability	No impact
Efficiency	No impact
Safety precaution	none or error

See Also

- Diagnosing Simulation Errors
- Rate Transitions and Asynchronous Blocks
- Configuration Parameters Dialog Box
- Diagnostics Pane: Sample Time

Enforce sample times specified by Signal Specification blocks

Select the diagnostic action to take if the sample time of the source port of a signal specified by a Signal Specification block differs from the signal's destination port.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- The Signal Specification block allows you to specify the attributes of the signal connected to its input and output ports. If the specified attributes conflict with the attributes specified by the blocks connected to its ports, Simulink software displays an error when it compiles the model, for example, at the beginning of a simulation. If no conflict exists, Simulink software eliminates the Signal Specification block from the compiled model.
- You can use the Signal Specification block to ensure that the actual attributes of a signal meet desired attributes, or to ensure correct propagation of signal attributes throughout a model.

Command-Line Information

Parameter: SigSpecEnsureSampleTimeMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors
- Signal Specification block
- Configuration Parameters Dialog Box
- Diagnostics Pane: Sample Time

Diagnostics Pane: Data Validity

Data Validity

Signals

Signal resolution: Detect overflow:

Division by singular matrix: Inf or NaN block output:

Underspecified data types: "it" prefix for identifiers:

Simulation range checking:

Parameters

Detect downcast: Detect overflow:

Detect underflow: Detect precision loss:

Detect loss of tunability:

Data Store Memory Block

Detect read before write: Multitask data store:

Detect write after read: Duplicate data store names:

Detect write after write:

Debugging

Array bounds exceeded:

Model Verification block enabling:

In this section...

“Data Validity Diagnostics Overview” on page 1-210

“Signal resolution” on page 1-211

“Division by singular matrix” on page 1-213

“Underspecified data types” on page 1-215

“Simulation range checking” on page 1-217

In this section...

- “Detect overflow” on page 1-219
- “Inf or NaN block output” on page 1-221
- “rt" prefix for identifiers” on page 1-223
- “Detect downcast” on page 1-225
- “Detect overflow” on page 1-227
- “Detect underflow” on page 1-229
- “Detect precision loss” on page 1-231
- “Detect loss of tunability” on page 1-233
- “Detect read before write” on page 1-235
- “Detect write after read” on page 1-237
- “Detect write after write” on page 1-239
- “Multitask data store” on page 1-241
- “Duplicate data store names” on page 1-243
- “Array bounds exceeded” on page 1-245
- “Model Verification block enabling” on page 1-247

Data Validity Diagnostics Overview

Specify what diagnostic action Simulink® software should take, if any, when it detects a condition that could compromise the integrity of data defined by the model, as well as the Data Validity parameters that pertain to code generation, and are used to debug a model.

Configuration

Set the parameters displayed.

Tips

- The options are typically to do nothing or to display a warning or an error message.
- A warning does not terminate a simulation, but an error does.

See Also

- Diagnosing Simulation Errors
- Solver Diagnostics
- Sample Time Diagnostics
- Type Conversion Diagnostics
- Connectivity Diagnostics
- Compatibility Diagnostics
- Model Referencing Diagnostics
- Saving Diagnostics
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Signal resolution

Select the how Simulink software resolves signals to Simulink.Signal objects in the MATLAB® workspace.

Settings

Default: Explicit only

Explicit only

Do not perform implicit signal resolution. Only explicitly (locally) specified signal resolutions occur.

Explicit and implicit

Perform implicit signal resolution wherever possible without posting any warnings.

Explicit and warn implicit

Perform implicit signal resolution wherever possible, posting a warning of each implicit resolution.

Tips

- Use the Signal Properties dialog box (see Signal Properties Dialog Box) to specify explicit resolution for signals.
- Use the **State Attributes** pane on dialog boxes of blocks that have discrete states, e.g., the Discrete-Time Integrator block, to specify explicit resolution for discrete states.
- Multiple signals can resolve to the same signal object and have the properties that the object specifies.
- The MathWorks discourages using implicit signal resolution except for fast prototyping, because implicit resolution slows performance, complicates model validation, and can have nondeterministic effects. Simulink software provides the `disableimplicitsignalresolution` function, which you can use to change settings throughout a model so that it does not use implicit signal resolution.

Command-Line Information

Parameter: SignalResolutionControl

Type: string

Value: 'UseLocalSettings' | 'TryResolveAll' |
'TryResolveAllWithWarning'

Default: 'UseLocalSettings'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	Explicit only

See Also

- Diagnosing Simulation Errors
- Simulink.Signal
- Signal Properties Dialog Box
- Discrete-Time Integrator block
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Division by singular matrix

Select the diagnostic action to take if the Product block detects a singular matrix while inverting one of its inputs in matrix multiplication mode.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Command-Line Information

Parameter: CheckMatrixSingularityMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors

- Product block
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Underspecified data types

Select the diagnostic action to take if Simulink software could not infer the data type of a signal during data type propagation.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Command-Line Information

Parameter: UnderSpecifiedDataTypeMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors

- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Simulation range checking

Select the diagnostic action to take when signals exceed specified minimum or maximum values.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- Use a block's **Output minimum** or **Minimum** parameter to specify the minimum value that the block should output.
- Use a block's **Output maximum** or **Maximum** parameter to specify the maximum value that the block should output.
- Enable this diagnostic to check whether block outputs exceed the minimum or maximum values that you specified.
- When **Simulation range checking** is enabled, Simulink software performs signal range checking at every time step during a simulation. Setting this diagnostic to warning or error can cause a decrease in simulation performance.

Command-Line Information

Parameter: SignalRangeChecking

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	warning or error
Traceability	warning or error
Efficiency	none
Safety precaution	error

See Also

- Checking Signal Ranges
- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Detect overflow

Select the diagnostic action to take if the value of a signal or parameter is too large to be represented by the signal or parameter's data type.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Command-Line Information

Parameter: IntegerOverflowMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors

- Working with Data Types
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Inf or NaN block output

Select the diagnostic action to take if the value of a block output is Inf or NaN at the current time step.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Command-Line Information

Parameter: SignalInfNanChecking

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors

- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

"rt" prefix for identifiers

Select the diagnostic action to take during code generation if a Simulink object name (the name of a parameter, block, or signal) begins with `rt`.

Settings

Default: error

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- The default setting (error) causes code generation to terminate with an error if it encounters a Simulink object name (parameter, block, or signal), that begins with `rt`.
- This is intended to prevent inadvertent clashes with generated identifiers whose names begins with `rt`.

Command-Line Information

Parameter: RTPrefix

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'error'

Recommended Settings

Application	Setting
Debugging	No impact

Application	Setting
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Detect downcast

Select the diagnostic action to take when a parameter downcast occurs during simulation.

Settings

Default: error

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- A parameter downcast occurs if the computation of block output required converting the parameter's specified type to a type having a smaller range of values (for example, from `uint32` to `uint8`).
- This diagnostic applies only to named tunable parameters.

Command-Line Information

Parameter: `ParameterDowncastMsg`

Type: `string`

Value: `'none' | 'warning' | 'error'`

Default: `'error'`

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Detect overflow

Select the diagnostic action to take if a parameter overflow occurs during simulation.

Settings

Default: error

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- A parameter overflow occurs if Simulink software encounters a parameter whose data type's range is not large enough to accommodate the parameter's ideal value (the ideal value is either too large or too small to be represented by the data type). For example, suppose that the parameter's ideal value is 200 and its data type is `uint8`. Overflow occurs in this case because the maximum value that `uint8` can represent is 127.
- Parameter overflow differs from parameter precision loss, which occurs when the ideal parameter value is within the range of the data type and scaling being used, but cannot be represented exactly.
- Both parameter overflow and precision loss are quantization errors, and the distinction between them can be a fine one. The **Detect overflow** diagnostic reports all quantization errors greater than one bit. For very small parameter quantization errors, precision loss will be reported rather than an overflow when

$$(Max + Slope) \geq V_{ideal} > (Min - Slope)$$

where

- *Max* is the maximum value representable by the parameter data type

- *Min* is the minimum value representable by the parameter data type
- *Slope* is the slope of the parameter data type (slope = 1 for integers)
- V_{ideal} is the ideal value of the parameter

Command-Line Information

Parameter: ParameterOverflowMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'error'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Detect underflow

Select the diagnostic action to take when a parameter underflow occurs during simulation.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- Parameter underflow occurs when Simulink software encounters a parameter whose data type does not have enough precision to represent the parameter's ideal value because the ideal value is too small.
- When parameter underflow occurs, casting the ideal value to the data type causes the parameter's modeled value to become zero, and therefore to differ from its ideal value.

Command-Line Information

Parameter: ParameterUnderflowMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact

Application	Setting
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Detect precision loss

Select the diagnostic action to take when parameter precision loss occurs during simulation.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- Precision loss occurs when Simulink software encounters a parameter whose data type does not have enough precision to represent the parameter's value exactly. As a result, the modeled value differs from the ideal value.
- Parameter precision loss differs from parameter overflow, which occurs when the range of the parameter's data type, i.e., that maximum value that it can represent, is smaller than the ideal value of the parameter.
- Both parameter overflow and precision loss are quantization errors, and the distinction between them can be a fine one. The **Detect Parameter overflow** diagnostic reports all parameter quantization errors greater than one bit. For very small parameter quantization errors, precision loss will be reported rather than an overflow when

$$(Max + Slope) \geq V_{ideal} > (Min - Slope)$$

where

- *Max* is the maximum value representable by the parameter data type.
- *Min* is the minimum value representable by the parameter data type.

- *Slope* is the slope of the parameter data type (slope = 1 for integers).
- V_{ideal} is the full-precision, ideal value of the parameter.

Command-Line Information

Parameter: ParameterPrecisionLossMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Detect loss of tunability

Select the diagnostic action to take when an expression with tunable variables is reduced to its numerical equivalent.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tip

If a tunable workspace variable is modified by Mask Initialization code, or is used in an arithmetic expression with unsupported operators or functions, the expression is reduced to a numeric expression and therefore cannot be tuned.

Command-Line Information

Parameter: ParameterTunabilityLossMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors
- Tunable Expressions
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Detect read before write

Select the diagnostic action to take if the model attempts to read data from a data store in which it has not stored data in this time step.

Settings

Default: Use local settings

Use local settings

For each data store defined by a Data Store Memory block, use the setting specified by the block. This option disables the diagnostic for global data stores (i.e., data stores defined by Simulink.Signal objects).

Disable All

Disables this diagnostic for all data stores accessed by the model.

Enable All As Warnings

Displays diagnostic as a warning at the MATLAB command line.

Enable All As Errors

Halts the simulation and displays the diagnostic in an error dialog box.

Command-Line Information

Parameter: ReadBeforeWriteMsg

Type: string

Value: 'UseLocalSettings' | 'DisableAll' | 'EnableAllAsWarning' | 'EnableAllAsError'

Default: 'UseLocalSettings'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	Enable all as errors

See Also

- [Diagnosing Simulation Errors](#)
- [Simulink.Signal](#)
- [Working with Data Stores](#)
- [Configuration Parameters Dialog Box](#)
- [Diagnostics Pane: Data Validity](#)

Detect write after read

Select the diagnostic action to take if the model attempts to store data in a data store after previously reading data from it in the current time step.

Settings

Default: Use local settings

Use local settings

For each data store defined by a Data Store Memory block, use the setting specified by the block. This option disables the diagnostic for global data stores (i.e., data stores defined by Simulink.Signal objects).

Disable All

Disables this diagnostic for all data stores accessed by the model.

Enable All As Warnings

Displays diagnostic as a warning at the MATLAB command line.

Enable All As Errors

Halts the simulation and displays the diagnostic in an error dialog box.

Command-Line Information

Parameter: WriteAfterReadMsg

Type: string

Value: 'UseLocalSettings' | 'DisableAll' | 'EnableAllAsWarning' | 'EnableAllAsError'

Default: 'UseLocalSettings'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	Enable all as errors

See Also

- [Diagnosing Simulation Errors](#)
- [Simulink.Signal](#)
- [Working with Data Stores](#)
- [Configuration Parameters Dialog Box](#)
- [Diagnostics Pane: Data Validity](#)

Detect write after write

Select the diagnostic action to take if the model attempts to store data in a data store twice in succession in the current time step.

Settings

Default: Use local settings

Use local settings

For each data store defined by a Data Store Memory block, use the setting specified by the block. This option disables the diagnostic for global data stores (i.e., data stores defined by Simulink.Signal objects).

Disable All

Disables this diagnostic for all data stores accessed by the model.

Enable All As Warnings

Displays diagnostic as a warning at the MATLAB command line.

Enable All As Errors

Halts the simulation and displays the diagnostic in an error dialog box.

Command-Line Information

Parameter: WriteAfterWriteMsg

Type: string

Value: 'UseLocalSettings' | 'DisableAll' | 'EnableAllAsWarning' | 'EnableAllAsError'

Default: 'UseLocalSettings'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	Enable all as errors

See Also

- [Diagnosing Simulation Errors](#)
- [Simulink.Signal](#)
- [Working with Data Stores](#)
- [Configuration Parameters Dialog Box](#)
- [Diagnostics Pane: Data Validity](#)

Multitask data store

Select the diagnostic action to take when one task reads data from a Data Store Memory block to which another task writes data.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- Such a situation is safe only if one of the tasks cannot interrupt the other, such as when the data store is a scalar and the writing task uses an atomic copy operation to update the store or the target does not allow the tasks to preempt each other.
- You should disable this diagnostic (set it to none) only if the application warrants it, such as if the application uses a cyclic scheduler that prevents tasks from preempting each other.

Command-Line Information

Parameter: MultiTaskDSMMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors
- Data Store Memory block
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Duplicate data store names

Select the diagnostic action to take when the model contains multiple Data Store Memory blocks that specify the same data store name.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Command-Line Information

Parameter: UniqueDataStoreMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Diagnosing Simulation Errors

- Data Store Memory block
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Array bounds exceeded

Select the diagnostic action to take when blocks write data to locations outside the memory allocated to them.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- Use this option to check whether execution of each instance of a block during model simulation writes data to memory locations not allocated to the block. This can happen only if your model includes a user-written S-function that has a bug.
- Enabling this option slows down model execution considerably. Thus, you should enable it only if you suspect that your model contains a user-written S-function that has a bug.
- This option causes Simulink software to check whether a block writes outside the memory allocated to it during simulation. Typically this can happen only if your model includes a user-written S-function that has a bug.
- See Writing S-Functions for more information on using this option.

Command-Line Information

Parameter: ConsistencyChecking

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	warning
Traceability	No impact
Efficiency	none
Safety precaution	No impact

See Also

- Diagnosing Simulation Errors
- Writing S-Functions
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Model Verification block enabling

Enable model verification blocks in the current model either globally or locally.

Settings

Default: Use local settings

Use local settings

Enables or disables blocks based on the value of the `Enable` assertion parameter of each block. If a block's `Enable` assertion parameter is on, the block is enabled; otherwise, the block is disabled.

Enable All

Enables all model verification blocks in the model regardless of the settings of their `Enable` assertion parameters.

Disable All

Disables all model verification blocks in the model regardless of the settings of their `Enable` assertion parameters.

Command-Line Information

Parameter: `AssertControl`

Type: `string`

Value: `'UseLocalSettings'` | `'EnableAll'` | `'DisableAll'`

Default: `'UseLocalSettings'`

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	Disable all

See Also

- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Data Validity

Diagnostics Pane: Type Conversion

Type Conversion	
Unnecessary type conversions:	<input type="text" value="none"/>
Vector/matrix block input conversion:	<input type="text" value="none"/>
32-bit integer to single precision float conversion:	<input type="text" value="warning"/>

In this section...

“Type Conversion Diagnostics Overview” on page 1-250

“Unnecessary type conversions” on page 1-251

“Vector/matrix block input conversion” on page 1-252

“32-bit integer to single precision float conversion” on page 1-254

Type Conversion Diagnostics Overview

Specify the diagnostic actions that Simulink® software should take when it detects a data type conversion problem while compiling the model.

Configuration

Set the parameters displayed.

Tips

- The options are typically to do nothing or to display a warning or an error message.
- A warning does not terminate a simulation, but an error does.

See Also

- Diagnosing Simulation Errors
- Solver Diagnostics
- Sample Time Diagnostics
- Data Validity Diagnostics
- Connectivity Diagnostics
- Compatibility Diagnostics
- Model Referencing Diagnostics
- Saving Diagnostics
- Configuration Parameters Dialog Box
- Diagnostics Pane: Type Conversion

Unnecessary type conversions

Select the diagnostic action to take when Simulink software detects a Data Type Conversion block used where no type conversion is necessary.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

Command-Line Information

Parameter: UnnecessaryDatatypeConvMsg

Type: string

Value: 'none' | 'warning'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	warning

See Also

- Diagnosing Simulation Errors
- Data Type Conversion block
- Configuration Parameters Dialog Box
- Diagnostics Pane: Type Conversion

Vector/matrix block input conversion

Select the diagnostic action to take when Simulink software detects a vector-to-matrix or matrix-to-vector conversion at a block input.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

Simulink software converts vectors to row or column matrices and row or column matrices to vectors under the following circumstances:

- If a vector signal is connected to an input that requires a matrix, Simulink software converts the vector to a one-row or one-column matrix.
- If a one-column or one-row matrix is connected to an input that requires a vector, Simulink software converts the matrix to a vector.
- If the inputs to a block consist of a mixture of vectors and matrices and the matrix inputs all have one column or one row, Simulink software converts the vectors to matrices having one column or one row, respectively.

Command-Line Information

Parameter: VectorMatrixConversionMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors
- Determining Output Signal Dimensions
- Configuration Parameters Dialog Box
- Diagnostics Pane: Type Conversion

32-bit integer to single precision float conversion

Select the diagnostic action to take if Simulink software detects a 32-bit integer value was converted to a floating-point value.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

Tip

Converting a 32-bit integer value to a floating-point value can result in a loss of precision. See *Working with Data Types* for more information.

Command-Line Information

Parameter: Int32ToFloatConvMsg

Type: string

Value: 'none' | 'warning'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	warning

See Also

- Diagnosing Simulation Errors

- Working with Data Types
- Configuration Parameters Dialog Box
- Diagnostics Pane: Type Conversion

Diagnostics Pane: Connectivity

Connectivity

Signals

Signal label mismatch: none

Unconnected block input ports: warning

Unconnected block output ports: warning

Unconnected line: warning

Buses

Unspecified bus object at root Output block: warning

Element name mismatch: warning

Mux blocks used to create bus signals: warning

Bus signal treated as vector: none

Function calls

Invalid function-call connection: error

Context-dependent inputs: Use local settings

In this section...

- “Connectivity Diagnostics Overview” on page 1-258
- “Signal label mismatch” on page 1-259
- “Unconnected block input ports” on page 1-261
- “Unconnected block output ports” on page 1-263
- “Unconnected line” on page 1-265
- “Unspecified bus object at root Output block” on page 1-267
- “Element name mismatch” on page 1-269
- “Mux blocks used to create bus signals” on page 1-271
- “Bus signal treated as vector” on page 1-274

In this section...

“Invalid function-call connection” on page 1-276

“Context-dependent inputs” on page 1-278

Connectivity Diagnostics Overview

Specify the diagnostic actions that Simulink® software should take when it detects a problem with block connections while compiling the model.

Configuration

Set the parameters displayed.

Tips

- The options are typically to do nothing or to display a warning or an error message.
- A warning does not terminate a simulation, but an error does.

See Also

- Diagnosing Simulation Errors
- Solver Diagnostics
- Sample Time Diagnostics
- Data Validity Diagnostics
- Type Conversion Diagnostics
- Compatibility Diagnostics
- Model Referencing Diagnostics
- Saving Diagnostics
- Configuration Parameters Dialog Box
- Diagnostics Pane: Connectivity

Signal label mismatch

Select the diagnostic action to take when Simulink software detects virtual signals that have a common source signal but different labels.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Command-Line Information

Parameter: SignalLabelMismatchMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors

- Virtual Signals
- Configuration Parameters Dialog Box
- Diagnostics Pane: Connectivity

Unconnected block input ports

Select the diagnostic action to take when the model contains a block with an unconnected input.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Command-Line Information

Parameter: UnconnectedInputMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors

- Configuration Parameters Dialog Box
- Diagnostics Pane: Connectivity

Unconnected block output ports

Select the diagnostic action to take when the model contains a block with an unconnected output.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Command-Line Information

Parameter: UnconnectedOutputMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors

- Configuration Parameters Dialog Box
- Diagnostics Pane: Connectivity

Unconnected line

Select the diagnostic action to take when the Model contains an unconnected line or an unmatched Goto or From block.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Command-Line Information

Parameter: UnconnectedLineMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors

- Goto block
- From block
- Configuration Parameters Dialog Box
- Diagnostics Pane: Connectivity

Unspecified bus object at root Output block

Select the diagnostic action to take while generating a simulation target for a referenced model if any of the model's root Output blocks is connected to a bus but does not specify a bus object (see `Simulink.Bus`).

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Command-Line Information

Parameter: `RootOutputRequireBusObject`

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors

- Outport block
- Simulink.Bus
- Configuration Parameters Dialog Box
- Diagnostics Pane: Connectivity

Element name mismatch

Select the diagnostic action to take if the name of a bus element does not match the name specified by the corresponding bus object.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tip

You can use this diagnostic along with bus objects to ensure that your model meets bus element naming requirements imposed by some blocks, such as the Switch block.

Command-Line Information

Parameter: BusObjectLabelMismatch

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Connectivity

Mux blocks used to create bus signals

Select the diagnostic action to take if Simulink software detects a signal that some blocks treat as a mux/vector, while other blocks treat the signal as a bus.

Settings

Default: warning

none

Simulink software takes no action.

This option disables checking for Mux blocks used to create buses.

warning

Simulink software displays a warning.

This option does not enforce strict bus behavior. However, if it detects a Mux block that creates a bus during model update or simulation, it displays a message in the MATLAB® Command Window that identifies the offending block. It does this for the first ten Mux blocks that it encounters that violate strict bus behavior.

error

Simulink software terminates the simulation and displays an error message identifying the offending Mux block.

This option enforces the following “strict bus” behavior during model editing, updating, and simulation:

- A Mux block with more than one input is allowed to output only a vector signal. A Mux block with only one input is allowed to output only a scalar, vector, or matrix signal. Simulink software displays all nonscalar Mux outputs as wide signals.
- The dialog boxes for Bus Creator and Bus Selector blocks allow you to select input signals created by Mux blocks but not the individual elements of those signals. For example, suppose that the bus connected to a Bus Selector includes a vector signal created by a Mux block. The Bus Selector allows you to select the vector signal but not any of its elements.

Tips

- This diagnostic detects use of Mux blocks to create buses. The diagnostic considers a signal created by a Mux block to be a bus if the signal meets either or both of the following conditions:
 - A Bus Selector block individually selects one or more of the signal's elements (as opposed to the entire signal).
 - The signal's components have differing data types, numeric types (complex or real), dimensionality, storage classes (see the *Real-Time Workshop*® *User's Guide* for information on storage classes), or sampling modes (see the Signal Processing Blockset™ documentation for information on frame-based sampling).
- You can avoid strict bus behavior errors and warnings by using `slreplace_mux` to remove Mux blocks that violate strict bus behavior from your model. Before executing the command, you should set this diagnostic to warning or none.
- See Intermixing Composite Signal Types for more information.

Dependency

Selecting error enables the following parameter:

- **Bus signal treated as vector**

Command-Line Information

Parameter: StrictBusMsg

Type: string

Value: 'none' | 'warning' | 'ErrorLevel1' |
'WarnOnBusInputToNonBusBlock' | 'ErrorOnBusInputToNonBusBlock'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact

Application	Setting
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Intermixing Composite Signal Types
- Diagnosing Simulation Errors
- Mux block
- Bus Creator block
- Bus Selector block
- Configuration Parameters Dialog Box
- Diagnostics Pane: Connectivity

Bus signal treated as vector

Select the diagnostic action to take when Simulink software detects that virtual bus signals are used to specify muxes/vectors.

Settings

Default: warning

none

Disables checking for buses used as muxes/vectors.

warning

Simulink software displays a warning if it detects a bus used as a mux/vector. This option does not enforce strict bus behavior.

error

Simulink software terminates the simulation and displays an error message when it builds a model that uses any virtual bus as a mux/vector.

Tips

- This diagnostic detects the use of virtual bus signals used to specify muxes/vectors. The diagnostic considers a virtual bus signal to be used as a mux/vector if it is input to a Demux block or to any block that can input a mux or a vector but is not formally defined as bus-capable. See Bus-Capable Blocks for details.
- Virtual buses can be used as muxes/vectors only when they contain no nested buses and all constituent signals have the same attributes. This practice is deprecated as of R2007a (V6.6) and may cease to be supported at some future time. The MathWorks therefore discourages mixing vectors, muxes, and virtual buses in new applications, and encourages upgrading existing applications to avoid such mixtures.
- You can eliminate warnings and errors about buses used as muxes/vectors by using `Simulink.BlockDiagram.addBusToVector` to insert a Bus to Vector block into any bus signal that is used as a mux/vector. Before executing the command, you should set this diagnostic to warning or none.
- See Intermixing Composite Signal Types for more information.

Dependency

This parameter is enabled only when **Mux blocks used to create bus signals** is set to error.

Command-Line Information

Parameter: StrictBusMsg

Type: string

Value: 'none' | 'warning' | 'ErrorLevel1' |
'WarnOnBusTreatedAsVector' | 'ErrorOnBusTreatedAsVector'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Intermixing Composite Signal Types
- Diagnosing Simulation Errors
- Bus-Capable Blocks
- Demux block
- Bus to Vector block
- Simulink.BlockDiagram.addBusToVector
- Configuration Parameters Dialog Box
- Diagnostics Pane: Connectivity

Invalid function-call connection

Select the diagnostic action to take if Simulink software detects incorrect use of a function-call subsystem.

Settings

Default: error

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- See the "Function-call systems" examples in the Simulink "Subsystem Semantics" library for examples of invalid uses of function-call subsystems.
- Setting this parameter to none or warning can lead to invalid simulation results.
- Setting this parameter to none or warning may cause Simulink software to insert extra delay operations.

Command-Line Information

Parameter: InvalidFcnCallConnMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'error'

Recommended Settings

Application	Setting
Debugging	No impact

Application	Setting
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors
- Subsystem Semantics library
- Configuration Parameters Dialog Box
- Diagnostics Pane: Connectivity

Context-dependent inputs

Select the diagnostic action to take when Simulink software has to compute any of a function-call subsystem's inputs directly or indirectly during execution of a call to a function-call subsystem.

Settings

Default: Use local settings

Use local settings

Issues a warning only if the corresponding diagnostic is selected on the function-call subsystem's parameters dialog box (see the documentation for the Subsystem block's parameter dialog box for more information).

Enable all

Enables this diagnostic for all function-call subsystems in this model.

Disable all

Disables this diagnostic for all function-call subsystems in this model.

Tips

- This situation occurs when executing a function-call subsystem can change its inputs.
- See the "Function-call systems" examples in the Simulink "Subsystem Semantics" library for examples of such function-call subsystems).

Command-Line Information

Parameter: FcnCallInpInsideContextMsg

Type: string

Value: 'Use local settings' | 'Enable All' | 'Disable All'

Default: 'Use local settings'

Recommended Settings

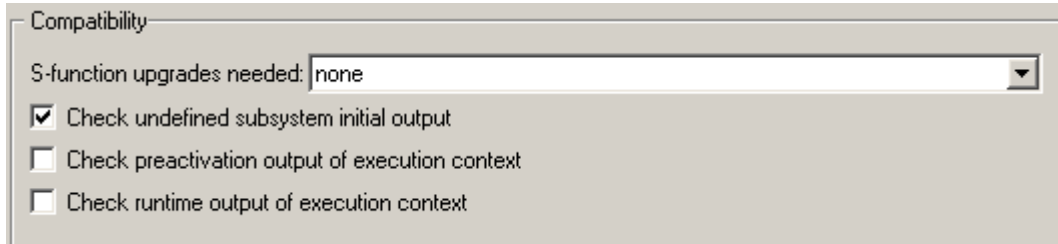
Application	Setting
Debugging	No impact

Application	Setting
Traceability	No impact
Efficiency	No impact
Safety precaution	Enable all

See Also

- Subsystem Semantics library
- Subsystem block
- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Connectivity

Diagnostics Pane: Compatibility



In this section...

“Compatibility Diagnostics Overview” on page 1-281

“S-function upgrades needed” on page 1-282

“Check undefined subsystem initial output” on page 1-284

“Check preactivation output of execution context” on page 1-287

“Check runtime output of execution context” on page 1-290

Compatibility Diagnostics Overview

Specify the diagnostic actions that Simulink® software should take when it detects an incompatibility between the current version of Simulink software and the model.

Configuration

Set the parameters displayed.

Tips

- The options are typically to do nothing or to display a warning or an error message.
- A warning does not terminate a simulation, but an error does.

See Also

- Diagnosing Simulation Errors
- Solver Diagnostics
- Sample Time Diagnostics
- Data Validity Diagnostics
- Type Conversion Diagnostics
- Connectivity Diagnostics
- Compatibility Diagnostics
- Model Referencing Diagnostics
- Saving Diagnostics
- Configuration Parameters Dialog Box
- Diagnostics Pane: Compatibility

S-function upgrades needed

Select the diagnostic action to take if Simulink software encounters a block that has not been upgraded to use features of the current release.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Command-Line Information

Parameter: SfunCompatibilityCheckMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Diagnosing Simulation Errors

- Configuration Parameters Dialog Box
- Diagnostics Pane: Compatibility

Check undefined subsystem initial output

Specify whether to display a warning if the model contains a conditionally executed subsystem in which a block with a specified initial condition drives an Output block with an undefined initial condition

Settings

Default: On

On

Displays a warning if the model contains a conditionally executed subsystem in which a block with a specified initial condition drives an Output block with an undefined initial condition.

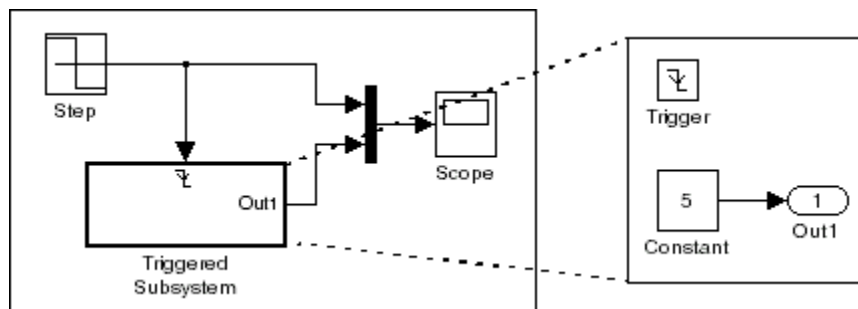
Off

Does not display a warning.

Tips

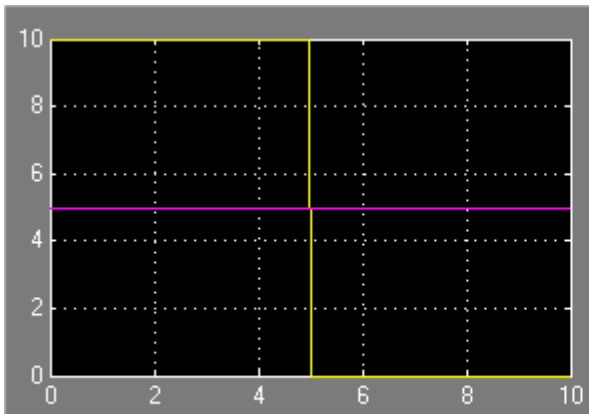
- This situation occurs when a block with a specified initial condition, such as a Constant, Initial Condition, or Delay block, drives an Output block with an undefined initial condition (**Initial output** parameter is set to []).
- Models with such subsystems can produce initial results (i.e., before initial activation of the conditionally executed subsystem) in the current release that differ from initial results produced in Release 13 or earlier releases.

Consider for example the following model.

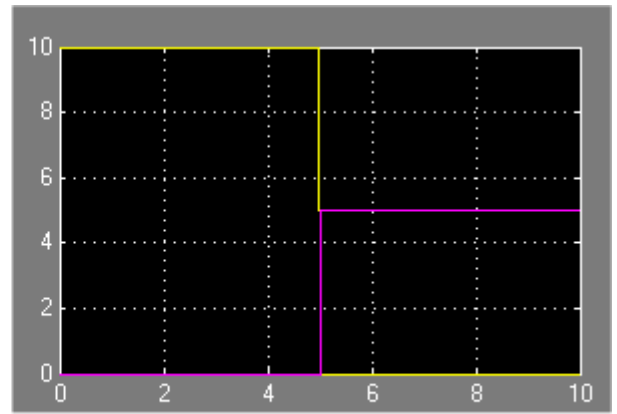


This model does not define the initial condition of the triggered subsystem's output port.

The following figure compares the superimposed output of this model's Step block and the triggered subsystem in Release 13 and the current release.



Release 13



Current Release

Notice that the initial output of the triggered subsystem differs between the two releases. This is because Release 13 and earlier releases use the initial output of the block connected to the output port (i.e., the Constant block) as the triggered subsystem's initial output. By contrast, this release outputs 0 as the initial output of the triggered subsystem because the model does not specify the port's initial output.

Command-Line Information

Parameter: CheckSSInitialOutputMsg

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	On

See Also

- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Compatibility

Check preactivation output of execution context

Specify whether to display a warning if Simulink software detects potential initial output differences from previous releases.

Settings

Default: Off



On

Displays a warning if Simulink software detects potential initial output differences from previous releases.



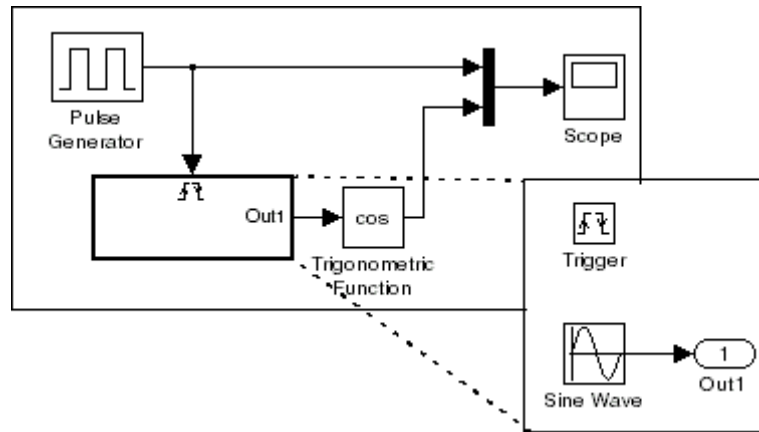
Off

Does not display a warning.

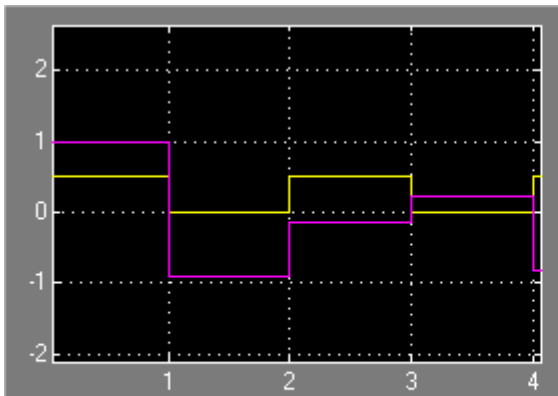
Tips

- This diagnostic is triggered if the model contains a block that meets the following conditions:
 - The block produces nonzero output for zero input (e.g., a Cosine block).
 - The block is connected to an output of a conditionally executed subsystem.
 - The block inherits its execution context from that subsystem.
 - The Output to which it is connected has an undefined initial condition, i.e., the Output block's **Initial output** parameter is set to [].
- Models with blocks that meet these criteria can produce initial results (i.e., before the conditionally executed subsystem is first activated in the current release that differ from initial results produced in Release 13 or earlier releases.

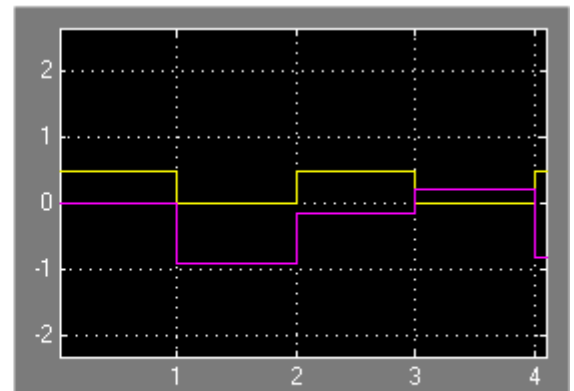
Consider for example the following model.



The following figure compares the superimposed output of the Pulse Generator and cos block in Release 13 and the current release.



Release 13



Current Release

Note that the initial output of the cos block differs between the two releases. This is because in Release 13, the cos block belongs to the execution context of the root system and hence executes at every time step whereas in the current release, the cos block belongs to the execution context of the triggered subsystem and hence executes only when the triggered subsystem executes.

Command-Line Information

Parameter: CheckExecutionContextPreStartOutputMsg

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	On

See Also

- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Compatibility

Check runtime output of execution context

Specify whether to display a warning if Simulink software detects potential output differences from previous releases.

Settings

Default: Off

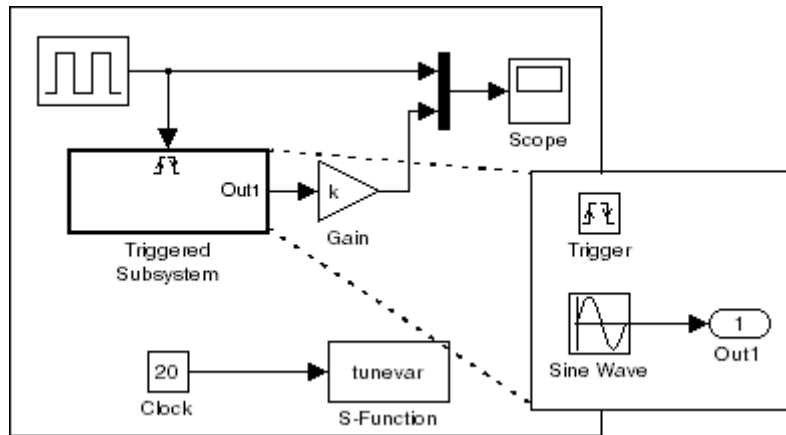
On
Displays a warning if Simulink software detects potential output differences from previous releases.

Off
Does not display a warning.

Tips

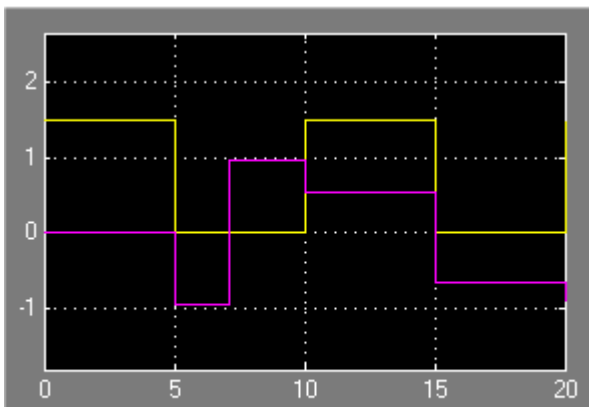
- This diagnostic is triggered if the model contains a block that meets the following conditions:
 - The block has a tunable parameter.
 - The block is connected to an output of a conditionally executed subsystem.
 - The block inherits its execution context from that subsystem.
 - The Output to which it is connected has an undefined initial condition, i.e., the Output block's **Initial output** parameter is set to [].
- Models with blocks that meet these criteria can produce results when the parameter is tuned in the current release that differ from results produced in Release 13 or earlier releases.

Consider for example the following model.

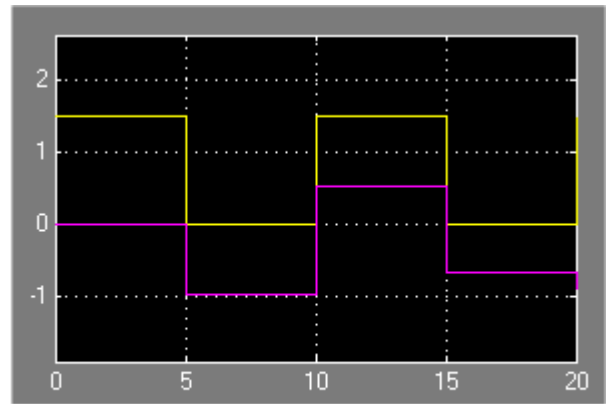


In this model, the tunevar S-function changes the value of the Gain block's k parameter and updates the diagram at simulation time 7 (i.e., it simulates tuning the parameter).

The following figure compares the superimposed output of the model's Pulse Generator block and its Gain block in Release 13 and the current release.



Release 13



Current Release

Note that the output of the Gain block changes at time 7 in Release 13 but does not change in the current release. This is because in Release 13, the Gain block belongs to the execution context of the root system and hence executes at every time step whereas in the current release, the Gain

block belongs to the execution context of the triggered subsystem and hence executes only when the triggered subsystem executes, i.e., at times 5, 10, 15, and 20.

Command-Line Information

Parameter: CheckExecutionContextRuntimeOutputMsg

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	On

See Also

- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Compatibility

Diagnostics Pane: Model Referencing

Model Referencing	
Model block version mismatch:	<input type="text" value="none"/>
Port and parameter mismatch:	<input type="text" value="none"/>
Model configuration mismatch:	<input type="text" value="none"/>
Invalid root Inport/Outport block connection:	<input type="text" value="none"/>
Unsupported data logging:	<input type="text" value="warning"/>

In this section...

“Model Referencing Diagnostics Overview” on page 1-294

“Model block version mismatch” on page 1-295

“Port and parameter mismatch” on page 1-297

“Model configuration mismatch” on page 1-299

“Invalid root Inport/Outport block connection” on page 1-301

“Unsupported data logging” on page 1-305

Model Referencing Diagnostics Overview

Specify the diagnostic actions that Simulink® software should take when it detects an incompatibility relating to a model reference hierarchy.

Configuration

Set the parameters displayed.

Tips

- The options are typically to do nothing or to display a warning or an error message.
- A warning does not terminate a simulation, but an error does.

See Also

- Referencing Models
- Diagnosing Simulation Errors
- Solver Diagnostics
- Sample Time Diagnostics
- Data Validity Diagnostics
- Type Conversion Diagnostics
- Connectivity Diagnostics
- Compatibility Diagnostics
- Saving Diagnostics
- Configuration Parameters Dialog Box
- Diagnostics Pane: Model Referencing

Model block version mismatch

Select the diagnostic action to take when loading or updating this model if Simulink software detects a mismatch between the version of the model used to create or refresh a Model block in this model and the referenced model's current version.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning and refreshes the Model block.

error

Simulink software displays an error message and does not refresh Model block.

Tip

If you have enabled display of referenced model version numbers on Model blocks for this model (see [Displaying Referenced Model Version Numbers](#)), Simulink software displays a version mismatch on the Model block icon, for example: Rev:1.0 != 1.2.

Command-Line Information

Parameter: ModelReferenceVersionMismatchMessage

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	none

See Also

- Referencing Models
- Diagnosing Simulation Errors
- Displaying Referenced Model Version Numbers
- Configuration Parameters Dialog Box
- Diagnostics Pane: Model Referencing

Port and parameter mismatch

Select the diagnostic action to take if Simulink software detects a port or parameter mismatch during model loading or updating.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning and refreshes the Model block.

error

Simulink software displays an error message and does not refresh the Model block.

Tips

- Port mismatches occur when there is a mismatch between the I/O ports of a Model block and the root-level I/O ports of the model it references.
- Parameter mismatches occur when there is a mismatch between the parameter arguments recognized by the Model block and the parameter arguments declared by the referenced model.
- Model block icons can display a message indicating port or parameter mismatches. To enable this feature, select **Block displays > Model Block I/O Mismatch** from the parent model's **Format** menu.

Command-Line Information

Parameter: ModelReferenceIOMismatchMessage

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Referencing Models
- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Model Referencing

Model configuration mismatch

Select the diagnostic action to take if the configuration parameters of a model referenced by this model do not match this model's configuration parameters or are inappropriate for a referenced model.

Settings

Default: none

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tip

Set this diagnostic to warning or error if you suspect that an inappropriate or mismatched configuration parameter may be causing your model to give the wrong result.

Command-Line Information

Parameter: ModelReferenceCSMismatchMessage

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	warning

See Also

- Referencing Models
- Configuration Parameter Requirements
- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Model Referencing

Invalid root Inport/Output block connection

Select the diagnostic action to take if Simulink software detects invalid internal connections to this model's root-level Output port blocks.

Settings

Default: none

none

Simulink software silently inserts hidden blocks to satisfy the constraints wherever possible.

warning

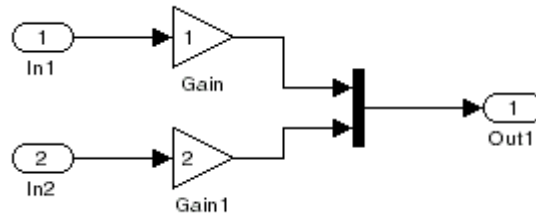
Simulink software warns you that a connection constraint has been violated and attempts to satisfy the constraint by inserting hidden blocks.

error

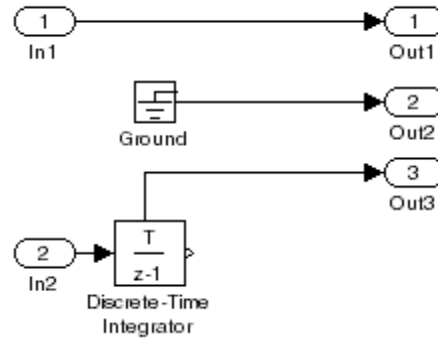
Simulink software terminates the simulation or code generation and displays an error message.

Tips

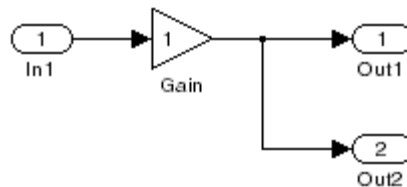
- In some cases (such as function-call feedback loops), automatically inserted hidden blocks may introduce delays and thus may change simulation results.
- Auto-inserting hidden blocks to eliminate root I/O problems stops at subsystem boundaries. Therefore, you may need to manually modify models with subsystems that violate any of the constraints below.
- The types of invalid internal connections are:
 - A root Output port is connected directly or indirectly to more than one nonvirtual block port:



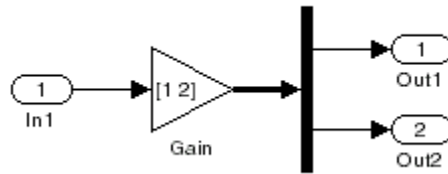
- A root Output port is connected to a root Input block, a Ground block, or a nondata port:



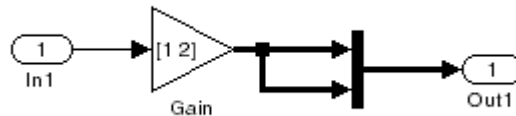
- Two root Output blocks are connected to the same block port:



- An Outputport block is connected to some elements of a block output and not others:



- An Outport block is connected more than once to the same element:



Command-Line Information

Parameter: ModelReferenceIOMsg
Type: string
Value: 'none' | 'warning' | 'error'
Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	error

See Also

- Referencing Models

- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Diagnostics Pane: Model Referencing

Unsupported data logging

Select the diagnostic action to take if this model contains To Workspace blocks or Scope blocks with data logging enabled.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning.

error

Simulink software terminates the simulation and displays an error message.

Tips

- The default action warns you that Simulink software does not support use of these blocks to log data from referenced models.
- See Logging Referenced Model Signals for information on how to log signals from a reference to this model.

Command-Line Information

Parameter: ModelReferenceDataLoggingMessage

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

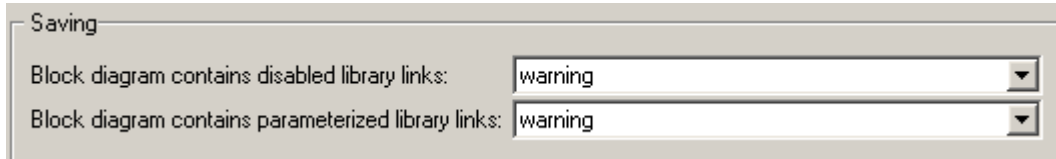
Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	error

See Also

- Referencing Models
- Diagnosing Simulation Errors
- Logging Referenced Model Signals
- To Workspace block
- Scope block
- Configuration Parameters Dialog Box
- Diagnostics Pane: Model Referencing

Diagnostics Pane: Saving



Saving

Block diagram contains disabled library links: warning

Block diagram contains parameterized library links: warning

In this section...

“Saving Tab Overview” on page 1-308

“Block diagram contains disabled library links” on page 1-309

“Block diagram contains parameterized library links” on page 1-311

Saving Tab Overview

Specify the diagnostic actions that Simulink® software takes when saving a block diagram containing disabled library links or parameterized library links.

Configuration

Set the parameters displayed.

Tips

- The options are typically to do nothing or to display a warning or an error message.
- A warning does not terminate a simulation, but an error does.

See Also

- Saving a Model
- Model Parameters
- Diagnosing Simulation Errors
- Solver Diagnostics
- Sample Time Diagnostics
- Data Validity Diagnostics
- Type Conversion Diagnostics
- Connectivity Diagnostics
- Compatibility Diagnostics
- Model Referencing Diagnostics
- Configuration Parameters Dialog Box
- Diagnostics Pane: Saving

Block diagram contains disabled library links

Select the diagnostic action to take when saving a model containing disabled library links.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning and saves the block diagram. The diagram may not contain the information you had intended.

error

Simulink software displays an error message. The model is not saved.

Tip

Use the Model Advisor Identify disabled library links check to find disabled library links.

Command-Line Information

Parameter: SaveWithDisabledLinksMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'warning'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- [Disabling Library Links](#)
- [Identify disabled library links](#)
- [Saving a Model](#)
- [Model Parameters](#)
- [Configuration Parameters Dialog Box](#)
- [Diagnostics Pane: Saving](#)

Block diagram contains parameterized library links

Select the diagnostic action to take when saving a model containing parameterized library links.

Settings

Default: warning

none

Simulink software takes no action.

warning

Simulink software displays a warning and saves the block diagram. The diagram may not contain the information you had intended.

error

Simulink software displays an error message. The model is not saved.

Tips

- Use the Model Advisor Identify parameterized library links check to find parameterized library links.

Command-Line Information

Parameter: SaveWithParameterizedLinksMsg

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'none'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Identify parameterized library links
- Configuration Parameters Dialog Box
- Diagnostics Pane: Saving

Hardware Implementation Pane

The screenshot shows a software configuration window titled "Hardware Implementation Pane". It is divided into two main sections: "Embedded hardware (simulation and code generation)" and "Emulation hardware (code generation only)".

Embedded hardware (simulation and code generation)

- Device vendor:
- Device type:
- Number of bits:
 - char:
 - short:
 - int:
 - long:
 - native word size:
- Byte ordering:
- Signed integer division rounds to:
- Shift right on a signed integer as arithmetic shift

Emulation hardware (code generation only)

- None

In this section...

- “Hardware Implementation Overview” on page 1-315
- “Device vendor” on page 1-316
- “Device type” on page 1-319
- “Number of bits: char” on page 1-328
- “Number of bits: short” on page 1-330
- “Number of bits: int” on page 1-332
- “Number of bits: long” on page 1-334
- “Number of bits: native word size” on page 1-336
- “Byte ordering” on page 1-338
- “Signed integer division rounds to” on page 1-340
- “Shift right on a signed integer as arithmetic shift” on page 1-343
- “None” on page 1-345
- “Device vendor” on page 1-347

In this section...

“Device type” on page 1-349

“Number of bits: char” on page 1-357

“Number of bits: short” on page 1-359

“Number of bits: int” on page 1-361

“Number of bits: long” on page 1-363

“Number of bits: native word size” on page 1-365

“Byte ordering” on page 1-367

“Signed integer division rounds to” on page 1-369

“Shift right on a signed integer as arithmetic shift” on page 1-372

Hardware Implementation Overview

Describe the hardware characteristics for the modelled system, including how to set up embedded and emulation hardware settings for both simulation and code generation.

Note Hardware Implementation pane options do not control hardware or compiler behavior: their purpose is to solely describe hardware and compiler properties to MATLAB® software, which uses the information to generate code that is correct for the platform, runs as efficiently as possible, and gives bit-true agreement among simulation results, production code, and test code.

Configuration

- 1 Choose the **Device type** in the Embedded hardware subpane.
- 2 Set the parameters displayed for the selected device type.
- 3 Apply the changes.
- 4 Repeat as needed for Emulation hardware.

Tips

- This pane applies to models of computer-based systems, such as embedded controllers.
- Specifying hardware characteristics enables simulation of the model to detect error conditions that could arise when executing code, such as hardware overflow.

See Also

- Configuring Hardware Properties
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Device vendor

Select the manufacturer of the hardware you will use to implement the production version of the system represented by this model.

Settings

Default: Generic

- AMD
- ARM Compatible
- ASIC/FPGA
- Analog Devices
- Atmel
- Freescale
- Infineon
- Intel
- Microchip
- NEC
- Renesas
- SGI
- STMicroelectronics
- Texas Instruments
- Generic

Tips

- Select the device vendor before you specify the hardware device used to define your system's constraints.
- If your test hardware does not match any of the listed vendors, select Generic.

- The **Device vendor** and **Device type** fields both share the same command line parameter: `ProdHWDeviceType`. When specifying this parameter from the command line, separate the device vendor and device type values using the characters `>`. For example: `'Intel->8051 Compatible'`.
- To add **Device vendor** and **Device type** values to the default set that is displayed on the **Hardware Implementation** pane, see “Registering Additional Device Vendor and Device Type Values” in the Real-Time Workshop® documentation.

Dependencies

This parameter determines the options available in the **Device type** drop-down menu.

Command-Line Information

Parameter: `ProdHWDeviceType`

Type: `string`

Value: any valid value (see tips)

Default: `'Generic->Unspecified (assume 32-bit Generic)'`

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Device type (production hardware)
- Device vendor (test hardware)
- Hardware Implementation Options

- Specifying Embedded Hardware Characteristics
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Device type

Select the type of hardware you will use to implement the production version of the system represented by this model.

Settings

Default: Unspecified (assume 32 bit Generic)

Generic options:

- 16-bit Embedded Processor
- 32-bit Embedded Processor
- 32-bit Real-Time Simulator
- 32-bit x-86 compatible
- 8-bit Embedded Processor
- Custom
- Unspecified (assume 32-bit Generic)

AMD® options:

- K5/K6/Athlon

ARM® Compatible options:

- ARM 7
- ARM 8
- ARM 9

ASIC/FPGA options:

- ASIC/FPGA

Analog Devices™ options:

- Blackfin

- SHARC
- TigerSHARC

Atmel® options:

- AVR

Freescale™ options:

- 32-bit PowerPC
- 68332
- 68HC08
- 68HC11
- ColdFire
- DSP563xx (16-bit mode)
- HC(S)12
- MPC5500

Infineon® options:

- C16x, XC16x
- TriCore

Intel® options:

- 8051 Compatible
- x86/Pentium

Microchip:

- PIC18
- dsPIC

NEC® options:

- V850

Renesas® options:

- M16C
- M32C
- R8C/Tiny
- SH-2/3/4

SGI:

- UltraSPARC IIi

STMicroelectronics:

- ST10/Super10

Texas Instruments™ options:

- C2000
- C5000
- C6000
- MSP430

Tips

- Select the device vendor before you specify the hardware device type.
- Selecting a device type specifies the hardware device to define your system's constraints:
 - Default hardware properties appear as the initial values.
 - Parameters with only one possible value cannot be changed.
 - Parameters with more than one possible value provide a pulldown list of legal values.
 - Static values for each device type are displayed in the following table.

- Parameters that you can modify are identified with an x.

Key:	Word size = native word size							
	Rounds to = Signed integer division rounds to							
	Shift right = Shift right on a signed integer as arithmetic shift							
Device vendor / Device type	Number of bits					Byte ordering	Rounds to	Shift right
	char	short	int	long	Word size			
Generic								
Unspecified (assume 32-bit Generic) (default)	8	16	32	32	32	Un-specified	x	Set
Custom	x	x	x	x	x	x	x	x
16-bit Embedded Processor	8	16	16	32	16	x	x	Set
32-bit Embedded Processor	8	16	32	32	32	x	x	Set
32-bit Real Time Simulator	8	16	32	32	32	x	x	Set
32-bit x86 compatible	8	16	32	32	32	Little Endian	Zero	Set
8-bit Embedded Processor	8	16	16	32	8	x	x	Set
AMD								
K5/K6/Athlon	8	16	32	32	32	Little Endian	x	Set
ARM Compatible								
ARM 7/8/9	8	16	32	32	x	x	x	x
ASIC/FPGA								
ASIC/FPGA	NA	NA	NA	NA	NA	NA	NA	NA
Analog Devices								

Key:	Word size = native word size							
	Rounds to = Signed integer division rounds to							
	Shift right = Shift right on a signed integer as arithmetic shift							
Device vendor / Device type	Number of bits					Byte ordering	Rounds to	Shift right
	char	short	int	long	Word size			
Blackfin	8	16	32	32	32	Little Endian	Zero	Set
SHARC	32	32	32	32	32	Big Endian	Zero	Set
TigerSHARC	32	32	32	32	32	Little Endian	Zero	Set
Atmel								
AVR	8	16	16	32	8	Little Endian	Zero	Set
Freescale								
32-bit PowerPC	8	16	32	32	32	Big Endian	Zero	Set
68332	8	16	32	32	32	Big Endian	x	Set
68HC08	8	16	16	32	8	Big Endian	x	Set
68HC11	8	16	16	32	8	Big Endian	x	Set
ColdFire	8	16	32	32	32	Big Endian	Zero	Set
DSP563xx (16-bit mode)	8	16	16	32	16	x	x	Set
HC(S)12	8	16	16	32	16	Big Endian	x	Set
MPC5500	8	16	32	32	32	x	Zero	Set

Key:	Word size = native word size							
	Rounds to = Signed integer division rounds to							
	Shift right = Shift right on a signed integer as arithmetic shift							
Device vendor / Device type	Number of bits					Byte ordering	Rounds to	Shift right
	char	short	int	long	Word size			
Infineon								
C16x, XC16x	8	16	16	32	16	Little Endian	Zero	Set
TriCore	8	16	32	32	32	Little Endian	x	Set
Intel								
8051 Compatible	8	16	16	32	8	x	x	Clear
xPC/Pentium	8	16	32	32	32	Little Endian	x	Set
Microchip								
PIC18	8	16	16	32	8	Little Endian	Zero	Set
dsPIC	8	16	16	32	16	Little Endian	Zero	Set
NEC								
V850	8	16	32	32	32	x	x	x
Renesas								
M16C	8	16	16	32	16	Little Endian	x	x
M32C	8	16	x	32	x	Little Endian	x	x
R8C/Tiny	8	16	16	32	16	Little Endian	x	x
SH-2/3/4	8	16	32	32	32	x	x	x

Key:	Word size = native word size							
	Rounds to = Signed integer division rounds to							
	Shift right = Shift right on a signed integer as arithmetic shift							
Device vendor / Device type	Number of bits					Byte ordering	Rounds to	Shift right
	char	short	int	long	Word size			
SGI								
UltraSPARC Iii	8	16	32	32	32	Big Endian	x	Set
STMicroelectronics								
ST10/Super10	8	16	16	32	16	Little Endian	Zero	Set
Texas Instruments								
C2000	16	16	16	32	16	x	Zero	Set
C5000	16	16	16	32	16	Big Endian	Zero	Set
C6000	8	16	32	40	32	x	Zero	Set

- If your production hardware does not match any of the listed types, select Unspecified (assume 32-bit Generic) if it has the characteristics of a generic 32-bit microprocessor; otherwise select Custom.
- The **Device vendor** and **Device type** fields both share the same command line parameter: TargetHWDeviceType. When specifying this parameter from the command line, separate the device vendor and device type values using the characters >. For example: 'Intel->8051 Compatible'.
- To add **Device vendor** and **Device type** values to the default set that is displayed on the **Hardware Implementation** pane, see “Registering Additional Device Vendor and Device Type Values” in the Real-Time Workshop documentation.

Dependencies

The options available in the drop-down menu are determined by the **Device vendor** parameter.

Selecting **ASIC/FPGA** enables the **Emulation hardware (code generation only)** subpane.

For all other device types, this parameter sets:

- **char**
- **short**
- **int**
- **long**
- **native word size**
- **Byte ordering**
- **Signed integer division rounds to**
- **Shift right on a signed integer as arithmetic shift**

Command-Line Information

Parameter: ProdHWDeviceType

Type: string

Value: any valid value (see tips)

Default: 'Generic->Unspecified (assume 32-bit Generic)'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Device vendor (production hardware)
- Device type (test hardware)
- Hardware Implementation Options
- Specifying Embedded Hardware Characteristics
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Number of bits: char

Describe the character bit length for the production hardware.

Settings

Default: 8

Minimum: 8

Maximum: 32

Enter a value between 8 and 32.

Tip

All values must be a multiple of 8.

Dependencies

- This parameter is enabled by **Device type**.
- Simulink® software disables this control if it knows the data type lengths for the selected device type.

Command-Line Information

Parameter: ProdBitPerChar

Type: integer

Value: any valid value

Default: 8

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	Target specific
Safety precaution	No impact for simulation and during development. Match operation of compiler and hardware for code generation.

See Also

- Hardware Implementation Options
- Specifying Embedded Hardware Characteristics
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Number of bits: short

Describe the data bit length for the production hardware.

Settings

Default: 16

Minimum: 8

Maximum: 32

Enter a value between 8 and 32.

Tip

All values must be a multiple of 8.

Dependencies

- This parameter is enabled by **Device type**.
- Simulink software disables this control if it knows the data type lengths for the selected device type.

Command-Line Information

Parameter: ProdBitPerShort

Type: integer

Value: any valid value

Default: 16

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	Target specific
Safety precaution	No impact for simulation and during development. Match operation of compiler and hardware for code generation.

See Also

- Hardware Implementation Options
- Specifying Embedded Hardware Characteristics
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Number of bits: int

Describe the data integer bit length for the production hardware.

Settings

Default: 32

Minimum: 8

Maximum: 32

Enter a number between 8 and 32.

Tip

All values must be a multiple of 8.

Dependencies

- This parameter is enabled by **Device type**.
- Simulink software disables this control if it knows the data type lengths for the selected device type.

Command-Line Information

Parameter: ProdBitPerInt

Type: integer

Value: any valid value

Default: 32

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	Target specific
Safety precaution	No impact for simulation and during development. Match operation of compiler and hardware for code generation.

See Also

- Hardware Implementation Options
- Specifying Embedded Hardware Characteristics
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Number of bits: long

Describe the data bit lengths for the production hardware.

Settings

Default: 32

Minimum: 8

Maximum: 32

Enter a value between 8 and 32.

Tip

All values must be a multiple of 8.

Dependencies

- This parameter is enabled by **Device type**.
- Simulink software disables this control if it knows the data type lengths for the selected device type.

Command-Line Information

Parameter: ProdBitPerLong

Type: integer

Value: any valid value

Default: 32

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	Target specific
Safety precaution	No impact for simulation and during development. Match operation of compiler and hardware for code generation.

See Also

- Hardware Implementation Options
- Specifying Embedded Hardware Characteristics
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Number of bits: native word size

Describe the microprocessor native word size for the production hardware.

Settings

Default: 32

Minimum: 8

Maximum: 32

Enter a value between 8 and 32.

Tip

All values must be a multiple of 8.

Dependencies

- This parameter is enabled by **Device type**.
- Simulink software disables this control if it knows the data type lengths for the selected device type.

Command-Line Information

Parameter: ProdWordSize

Type: integer

Value: any valid value

Default: 32

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	Target specific
Safety precaution	No impact for simulation and during development. Match operation of compiler and hardware for code generation.

See Also

- Hardware Implementation Options
- Specifying Embedded Hardware Characteristics
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Byte ordering

Describe the byte ordering for the production hardware.

Settings

Default: Unspecified

Unspecified

Specifies that the code determines the endianness of the hardware.
This is the least efficient choice.

Big Endian

The most significant byte appears first.

Little Endian

The least significant byte appears first.

Dependencies

- This parameter is enabled by **Device type**.
- Simulink software disables this control if it knows the data type lengths for the selected device type.

Command-Line Information

Parameter: ProdEndianness

Type: string

Value: 'Unspecified' | 'LittleEndian' | 'BigEndian'

Default: 'Unspecified'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	No impact

See Also

- Hardware Implementation Options
- Specifying Embedded Hardware Characteristics
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Signed integer division rounds to

Describe how to produce a signed integer quotient for the production hardware.

Settings

Default: Undefined

Undefined

Choose this option if neither Zero nor Floor describes the compiler's behavior, or if that behavior is unknown.

Zero

If the quotient is between two integers, the compiler chooses the integer that is closer to zero as the result.

Floor

If the quotient is between two integers, the compiler chooses the integer that is closer to negative infinity.

Tips

- Use the **Round integer calculations toward** parameter on your model's blocks to simulate the rounding behavior of the C compiler that you intend to use to compile code generated from the model. This setting appears on the Signal Data Type pane of the parameter dialog boxes of blocks that can perform signed integer arithmetic.
- For most blocks, the value of **Round integer calculations toward** completely defines rounding behavior. For blocks that support fixed-point data and the Simplest rounding mode, the value of **Signed integer division rounds to** also affects rounding. For details, see "Rounding" in the *Simulink® Fixed Point™ User's Guide*.
- See Hardware Implementation Options in the Real-Time Workshop documentation for information on how this option affects code generation.
- The following table illustrates the compiler behavior described by the options for this parameter.

N	D	Ideal N/D	Zero	Floor	Undefined
33	4	8.25	8	8	8
-33	4	-8.25	-8	-9	-8 or -9
33	-4	-8.25	-8	-9	-8 or -9
-33	-4	8.25	8	8	8

Dependency

This parameter is enabled by **Device type**.

Command-Line Information

Parameter: ProdIntDivRoundTo

Type: string

Value: 'Floor' | 'Zero' | 'Undefined'

Default: 'Undefined'

Recommended settings

Application	Setting
Debugging	No impact for simulation and during development. Undefined for production code generation.
Traceability	No impact for simulation and during development. Zero or Floor for production code generation.
Efficiency	No impact for simulation and during development. Zero for production code generation.
Safety precaution	No impact for simulation and during development. Floor for production code generation.

See Also

- [Hardware Implementation Options](#)
- [Specifying Embedded Hardware Characteristics](#)
- [Configuration Parameters Dialog Box](#)
- [Hardware Implementation Pane](#)

Shift right on a signed integer as arithmetic shift

Describe how your compiler rounds the result of two signed integers for the production hardware.

Settings

Default: On



On

Generates simple efficient code whenever the Simulink model performs arithmetic shifts on signed integers.



Off

Generates fully portable but less efficient code to implement right arithmetic shifts.

Tips

- Select this parameter if the C compiler implements a signed integer right shift as an arithmetic right shift.
- An arithmetic right shift fills bits vacated by the right shift with the value of the most significant bit, which indicates the sign of the number in twos complement notation.

Dependency

This parameter is enabled by **Device type**.

Command-Line Information

Parameter: ProdShiftRightIntArith

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	On
Safety precaution	No impact

See Also

- Hardware Implementation Options
- Specifying Embedded Hardware Characteristics
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

None

Specify whether the test hardware differs from the deployment hardware.

Settings

Default: On

On
Specifies that the hardware used to test the code generated from the model is the same as the production hardware, or has the same characteristics.

Off
Specifies that the hardware used to test the code generated from the model has different characteristics than the production hardware.

Tips

- You can generate code that runs on the test hardware but behaves as if it had been generated for and executed on the deployment hardware.
- The **Embedded hardware (simulation and code generation)** subpane specifies the deployment hardware properties. The **Emulation hardware (code generation only)** subpane is used to specify the test hardware properties.

Dependency

Enables the Emulation hardware subpane.

Command-Line Information

Parameter: ProdEqTarget

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

More information

- [Specifying Emulation Hardware Characteristics](#)
- [Hardware Implementation Options](#)
- [Configuration Parameters Dialog Box](#)
- [Hardware Implementation Pane](#)

Device vendor

Select the manufacturer of the hardware that will be used to test the code generated from the model.

Settings

Default: Generic

- AMD
- ARM Compatible
- ASIC/FPGA
- Analog Devices
- Atmel
- Freescale
- Infineon
- Intel
- Microchip
- NEC
- Renesas
- SGI
- STMicroelectronics
- Texas Instruments
- Generic

Tips

- Select the device vendor before you specify the hardware device used to define your system's constraints.
- If your test hardware does not match any of the listed vendors, select Generic.

- The **Device vendor** and **Device type** fields both share the same command line parameter: `TargetHWDeviceType`. When specifying this parameter from the command line, separate the device vendor and device type values using the characters `>`. For example: `'Intel->8051 Compatible'`.

Dependencies

This parameter determines the options available in the **Device type** drop-down menu.

Command-Line Information

Parameter: `TargetHWDeviceType`

Type: `string`

Value: any valid value (see tips)

Default: `'Generic->Unspecified (assume 32-bit Generic)'`

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Device type (test hardware)
- Specifying Emulation Hardware Characteristics
- Hardware Implementation Options
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Device type

Select the type of hardware that will be used to test the code generated from the model.

Settings

Default: Unspecified (assume 32 bit Generic)

Generic options:

- 16-bit Embedded Processor
- 32-bit Embedded Processor
- 32-bit Real-Time Simulator
- 32-bit x-86 compatible
- 8-bit Embedded Processor
- Custom
- MATLAB Host Computer
- Unspecified (assume 32-bit Generic)

AMD options:

- K5/K6/Athlon

ARM Compatible options:

- ARM 7
- ARM 8
- ARM 9

Analog Devices options:

- Blackfin
- SHARC
- TigerSHARC

Atmel options:

- AVR

Freescale options:

- 32-bit PowerPC
- 68332
- 68HC08
- 68HC11
- ColdFire
- DSP563xx (16-bit mode)
- HC(S)12
- MPC5500

Infineon options:

- C16x, XC16x
- TriCore

Intel options:

- 8051 Compatible
- x86/Pentium

Microchip:

- PIC18
- dsPIC

NEC options:

- V850

Renesas options:

- M16C
- R8C/Tiny
- SH-2/3/4

SGI:

- UltraSPARC IIi

STMicroelectronics:

- ST10/Super10

Texas Instruments options:

- C2000
- C5000
- C6000
- MSP430

Tips

- Select the device vendor before you specify the hardware device type.
- Selecting a device type specifies the hardware device to define your system's constraints:
 - Default hardware properties appear as the initial values.
 - Parameters with only one possible value cannot be changed.
 - Parameters with more than one possible value provide a pulldown list of legal values.
 - Static values for each device type are displayed in the following table. Parameters that you can modify are identified with an x.

Key:	Word size = native word size							
	Rounds to = Signed integer division rounds to							
	Shift right = Shift right on a signed integer as arithmetic shift							
Device vendor / Device type	Number of bits					Byte ordering	Rounds to	Shift right
	char	short	int	long	Word size			
Generic								
Unspecified (assume 32-bit Generic) (default)	8	16	32	32	32	Un-specified	x	Set
Custom	x	x	x	x	x	x	x	x
16-bit Embedded Processor	8	16	16	32	16	x	x	Set
32-bit Embedded Processor	8	16	32	32	32	x	x	Set
32-bit Real Time Simulator	8	16	32	32	32	x	x	Set
32-bit x86 compatible	8	16	32	32	32	Little Endian	Zero	Set
8-bit Embedded Processor	8	16	16	32	8	x	x	Set
MATLAB Host Computer	8	16	32	32	32	Little Endian	x	Set
AMD								
K5/K6/Athlon	8	16	32	32	32	Little Endian	x	Set
ARM Compatible								
ARM 7/8/9	8	16	32	32	x	x	x	x
ASIC/FPGA								
ASIC/FPGA	NA	NA	NA	NA	NA	NA	NA	NA

Key:	Word size = native word size							
	Rounds to = Signed integer division rounds to							
	Shift right = Shift right on a signed integer as arithmetic shift							
Device vendor / Device type	Number of bits					Byte ordering	Rounds to	Shift right
	char	short	int	long	Word size			
Analog Devices								
Blackfin	8	16	32	32	32	Little Endian	Zero	Set
SHARC	32	32	32	32	32	Big Endian	Zero	Set
TigerSHARC	32	32	32	32	32	Little Endian	Zero	Set
Atmel								
AVR	8	16	16	32	8	Little Endian	Zero	Set
Freescale								
32-bit PowerPC	8	16	32	32	32	Big Endian	Zero	Set
68332	8	16	32	32	32	Big Endian	x	Set
68HC08	8	16	16	32	8	Big Endian	x	Set
68HC11	8	16	16	32	8	Big Endian	x	Set
ColdFire	8	16	32	32	32	Big Endian	Zero	Set
DSP563xx (16-bit mode)	8	16	16	32	16	x	x	Set
HC(S)12	8	16	16	32	16	Big Endian	x	Set

Key:	Word size = native word size							
	Rounds to = Signed integer division rounds to							
	Shift right = Shift right on a signed integer as arithmetic shift							
Device vendor / Device type	Number of bits					Byte ordering	Rounds to	Shift right
	char	short	int	long	Word size			
MPC5500	8	16	32	32	32	x	Zero	Set
Infineon								
C16x, XC16x	8	16	16	32	16	Little Endian	Zero	Set
TriCore	8	16	32	32	32	Little Endian	x	Set
Intel								
8051 Compatible	8	16	16	32	8	x	x	Clear
xPC/Pentium	8	16	32	32	32	Little Endian	x	Set
Microchip								
PIC18	8	16	16	32	8	Little Endian	Zero	Set
dsPIC	8	16	16	32	16	Little Endian	Zero	Set
NEC								
V850	8	16	32	32	32	x	x	x
Renesas								
M16C	8	16	16	32	16	Little Endian	x	x
M32C	8	16	x	32	x	Little Endian	x	x
R8C/Tiny	8	16	16	32	16	Little Endian	x	x

Key:	Word size = native word size							
	Rounds to = Signed integer division rounds to							
	Shift right = Shift right on a signed integer as arithmetic shift							
Device vendor / Device type	Number of bits					Byte ordering	Rounds to	Shift right
	char	short	int	long	Word size			
SH-2/3/4	8	16	32	32	32	x	x	x
SGI								
UltraSPARC Iii	8	16	32	32	32	Big Endian	x	Set
STMicroelectronics								
ST10/Super10	8	16	16	32	16	Little Endian	Zero	Set
Texas Instruments								
C2000	16	16	16	32	16	x	Zero	Set
C5000	16	16	16	32	16	Big Endian	Zero	Set
C6000	8	16	32	40	32	x	Zero	Set

- If your production hardware does not match any of the listed types, select Unspecified (assume 32-bit Generic) if it has the characteristics of a generic 32-bit microprocessor; otherwise select Custom.
- The **Device vendor** and **Device type** fields both share the same command line parameter: TargetHWDeviceType. When specifying this parameter from the command line, separate the device vendor and device type values using the characters >. For example: 'Intel->8051 Compatible'.

Dependencies

The options available in the drop-down menu are determined by the **Device vendor** parameter.

This parameter sets:

- **char**
- **short**
- **int**
- **long**
- **native word size**
- **Byte ordering**
- **Signed integer division rounds to**
- **Shift right on a signed integer as arithmetic shift**

Command-Line Information

Parameter: TargetHWDeviceType

Type: string

Value: any valid value (see tips)

Default: 'Generic->Unspecified (assume 32-bit Generic)'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Device vendor (test hardware)
- Specifying Emulation Hardware Characteristics
- Hardware Implementation Options
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Number of bits: char

Describe the character bit length for the hardware used to test code.

Settings

Default: 8

Minimum: 8

Maximum: 32

Enter a value between 8 and 32.

Tip

All values must be a multiple of 8.

Dependencies

- This parameter is enabled by **Device type**.
- Simulink software disables this control if it knows the data type lengths for the selected device type.

Command-Line Information

Parameter: TargetBitPerChar

Type: integer

Value: any valid value

Default: 8

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	Target specific
Safety precaution	No impact for simulation and during development. Match operation of compiler and hardware for code generation.

See Also

- Specifying Emulation Hardware Characteristics
- Hardware Implementation Options
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Number of bits: short

Describe the data bit length for the hardware used to test code.

Settings

Default: 16

Minimum: 8

Maximum: 32

Enter a value between 8 and 32.

Tip

All values must be a multiple of 8.

Dependencies

- This parameter is enabled by **Device type**.
- Simulink software disables this control if it knows the data type lengths for the selected device type.

Command-Line Information

Parameter: TargetBitPerShort

Type: integer

Value: any valid value

Default: 16

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	Target specific
Safety precaution	No impact for simulation and during development. Match operation of compiler and hardware for code generation.

See Also

- Specifying Emulation Hardware Characteristics
- Hardware Implementation Options
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Number of bits: int

Describe the data integer bit length of the hardware used to test code.

Settings

Default: 32

Minimum: 8

Maximum: 32

Enter a number between 8 and 32.

Tip

All values must be a multiple of 8.

Dependencies

- This parameter is enabled by **Device type**.
- Simulink software disables this control if it knows the data type lengths for the selected device type.

Command-Line Information

Parameter: TargetBitPerInt

Type: integer

Value: any valid value

Default: 32

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	Target specific
Safety precaution	No impact for simulation and during development. Match operation of compiler and hardware for code generation.

See Also

- Specifying Emulation Hardware Characteristics
- Hardware Implementation Options
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Number of bits: long

Describe the data bit lengths for the hardware used to test code.

Settings

Default: 32

Minimum: 8

Maximum: 32

Enter a value between 8 and 32.

Tip

All values must be a multiple of 8.

Dependencies

- This parameter is enabled by **Device type**.
- Simulink software disables this control if it knows the data type lengths for the selected device type.

Command-Line Information

Parameter: TargetBitPerLong

Type: integer

Value: any valid value

Default: 32

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	Target specific
Safety precaution	No impact for simulation and during development. Match operation of compiler and hardware for code generation.

See Also

- Specifying Emulation Hardware Characteristics
- Hardware Implementation Options
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Number of bits: native word size

Describe the microprocessor native word size for the hardware used to test code.

Settings

Default: 32

Minimum: 8

Maximum: 32

Enter a value between 8 and 32.

Tip

All values must be a multiple of 8.

Dependencies

- This parameter is enabled by **Device type**.
- Simulink software disables this control if it knows the data type lengths for the selected device type.

Command-Line Information

Parameter: TargetWordSize

Type: integer

Value: any valid value

Default: 32

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	Target specific
Safety precaution	No impact for simulation and during development. Match operation of compiler and hardware for code generation.

See Also

- Specifying Emulation Hardware Characteristics
- Hardware Implementation Options
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Byte ordering

Describe the byte ordering for the hardware used to test code.

Settings

Default: Unspecified

Unspecified

Specifies that the code determines the endianness of the hardware.
This is the least efficient choice.

Big Endian

The most significant byte comes first.

Little Endian

The least significant byte comes first.

Note For guidelines to observe when configuring **Embedded hardware** controls for code generation, see Hardware Implementation Options in the Real-Time Workshop documentation.

Dependencies

- This parameter is enabled by **Device type**.
- Simulink software disables this control if it knows the data type lengths for the selected device type.

Command-Line Information

Parameter: TargetEndianness

Type: string

Value: 'Unspecified' | 'LittleEndian' | 'BigEndian'

Default: 'Unspecified'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact for simulation and during development. Match operation of compiler and hardware for code generation.

See Also

- Specifying Emulation Hardware Characteristics
- Hardware Implementation Options
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Signed integer division rounds to

Describe how to produce a signed integer quotient for the hardware used to test code.

Settings

Default: Undefined

Undefined

Choose this option if neither Zero nor Floor describes the compiler's behavior, or if that behavior is unknown.

Zero

If the quotient is between two integers, the compiler chooses the integer that is closer to zero as the result.

Floor

If the quotient is between two integers, the compiler chooses the integer that is closer to negative infinity.

Tips

- Use the **Round integer calculations toward** parameter on your model's blocks to simulate the rounding behavior of the C compiler that you intend to use to compile code generated from the model. This setting appears on the Signal Data Type pane of the parameter dialog boxes of blocks that can perform signed integer arithmetic.
- For most blocks, the value of **Round integer calculations toward** completely defines rounding behavior. For blocks that support fixed-point data and the Simplest rounding mode, the value of **Signed integer division rounds to** also affects rounding. For details, see "Rounding" in the *Simulink Fixed Point User's Guide*.
- See Hardware Implementation Options in the Real-Time Workshop documentation for information on how this option affects code generation.
- The following table illustrates the compiler behavior described by the options for this parameter.

N	D	Ideal N/D	Zero	Floor	Undefined
33	4	8.25	8	8	8
-33	4	-8.25	-8	-9	-8 or -9
33	-4	-8.25	-8	-9	-8 or -9
-33	-4	8.25	8	8	8

Dependency

This parameter is enabled by **Device type**.

Command-Line Information

Parameter: TargetIntDivRoundTo

Type: string

Value: 'Floor' | 'Zero' | 'Undefined'

Default: 'Undefined'

Recommended settings

Application	Setting
Debugging	No impact for simulation and during development. Undefined for production code generation.
Traceability	No impact for simulation and during development. Zero or Floor for production code generation.
Efficiency	No impact for simulation and during development. Zero for production code generation.
Safety precaution	No impact for simulation and during development. Floor for production code generation.

See Also

- [Specifying Emulation Hardware Characteristics](#)
- [Hardware Implementation Options](#)
- [Configuration Parameters Dialog Box](#)
- [Hardware Implementation Pane](#)

Shift right on a signed integer as arithmetic shift

Describe how your compiler rounds the result of two signed integers for the hardware used to test code.

Settings

Default: On



Generates simple efficient code whenever the Simulink model performs arithmetic shifts on signed integers.



Generates fully portable but less efficient code to implement right arithmetic shifts.

Tips

- Select this parameter if your C compiler implements a signed integer right shift as an arithmetic right shift.
- An arithmetic right shift fills bits vacated by the right shift with the value of the most significant bit, which indicates the sign of the number in twos complement notation. It is equivalent to dividing the number by 2.
- This setting affects only code generation

Dependency

This parameter is enabled by **Device type**.

Command-Line Information

Parameter: TargetShiftRightIntArith

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	On
Safety precaution	No impact

See Also

- Specifying Emulation Hardware Characteristics
- Hardware Implementation Options
- Configuration Parameters Dialog Box
- Hardware Implementation Pane

Model Referencing Pane

Rebuild options for all referenced models

Rebuild options:

Options for referencing this model

Total number of instances allowed per top model:

Model dependencies:

```
% Specify the model dependencies as a cell array of file names. The dependencies
% automatically include the model.mdl and linked library .mdl files. For files
% not on the MATLAB path, use absolute paths; prefix $MDL to a file path if the
% path is relative to the location of the .mdl file; wildcards are allowed; use a '%'
% to comment out a line; use '...' to continue lines. For example,
%
% {'D:\Work\parameters.mat', '$MDL\mdlvars.mat', ...
% 'D:\Work\masks\*.m'}
```

Pass scalar root inputs by value for Real-Time Workshop

Minimize algebraic loop occurrences

In this section...

“Model Referencing Pane Overview” on page 1-376

“Rebuild options” on page 1-377

“Never rebuild targets diagnostic” on page 1-380

“Total number of instances allowed per top model” on page 1-382

In this section...

“Model dependencies” on page 1-384

“Pass scalar root inputs by value for Real-Time Workshop” on page 1-386

“Minimize algebraic loop occurrences” on page 1-388

Model Referencing Pane Overview

Specify the options for including other models in this model, this model in other models, and for building simulation and code generation targets.

Configuration

Set the parameters displayed.

Tips

- The Model Referencing pane allows you to specify options for:
 - Including other models in this model.
 - Including the current model in other models.
- The option descriptions use the term *this model* to refer to the model that you are configuring and the term *referenced model* to designate models referenced by *this model*.

See Also

- Model Dependencies
- Configuration Parameters Dialog Box
- Model Referencing Pane

Rebuild options

Select whether to rebuild simulation and Real-Time Workshop® targets for referenced models before updating, simulating, or generating code from this model.

Settings

Default: If any changes detected

If any changes detected

Rebuilds the target for a referenced model if Simulink® software detects any changes of any kind in the target's dependencies. This option also checks for changes in the compiled form of the referenced model. Checking the compiled model can detect some changes that occur even in dependencies that you do not specify.

Always

Rebuilds all targets referenced by this model before simulating, updating, or generating code from it.

If any changes in known dependencies detected

Rebuilds a target if Simulink software detects any changes in known target dependencies (see below) since the target was last built. This option ignores cosmetic changes, such as annotation changes, in the referenced model and in any block library dependencies, thus preventing unnecessary rebuilds.

Never

Never rebuild targets before simulating or generating code from this model. If you are certain that your targets are up-to-date, you can use this option to avoid time-consuming target dependency checking when simulating, updating, or generating code from a model.

Tips

- Simulink software detects target dependencies in:
 - The referenced model's model file
 - Block library files used by the referenced model
 - Targets of models referenced by the referenced model

- S-functions and associated TLC files used by the referenced model
- User-specified dependencies (see Model Dependencies)
- Workspace variables used by the referenced model
- It is a good idea to use the Always option before deployment of a model to assure that all the model reference targets are up-to-date.
- Before selecting If any changes in known dependencies detected, you should be certain that you have specified every user-created dependency (e.g., M-files or MAT-files) for this model to ensure that all targets that need to be rebuilt are rebuilt. Otherwise, invalid simulation results may occur.
- The If any changes in known dependencies detected option cannot detect changes in unspecified dependencies, such as M-files used to initialize block masks. If you suspect that a model has such unknown dependencies, you can still guarantee valid simulation by selecting the Always or the If any changes detected option.
- Use the Never option with caution because it may lead to invalid results if referenced model targets are not in fact up-to-date.

Dependency

Selecting Never enables the **Never rebuild targets diagnostic** parameter.

Command-Line Information

Parameter: UpdateModelReferenceTargets

Type: string

Value: 'IfOutOfDate' | 'Force' | 'AssumeUpToDate' | 'IfOutOfDateOr Structural Change'

Default: 'IfOutOfDateOr Structural Change'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	Never or If any changes detected

See Also

- Model Dependencies
- Configuration Parameters Dialog Box
- Model Referencing Pane

Never rebuild targets diagnostic

Select the diagnostic action that Simulink software should take if it detects a target that needs to be rebuilt.

Settings

Default: Error if targets require rebuild

none

Simulink software takes no action.

Warn if targets require rebuild

Simulink software displays a warning.

Error if targets require rebuild

Simulink software terminates the simulation and displays an error message.

Tip

Selecting None bypasses dependency checking, and thus enables faster updating, simulation, and code generation, but can cause models that are not up-to-date to malfunction or generate incorrect results.

Dependency

This parameter is enabled only if you select Never in the **Rebuild options** field.

Command-Line Information

Parameter: CheckModelReferenceTargetMessage

Type: string

Value: 'none' | 'warning' | 'error'

Default: 'error'

Recommended Settings

Application	Setting
Debugging	No impact

Application	Setting
Traceability	No impact
Efficiency	No impact
Safety precaution	error if targets require rebuild

See Also

- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Model Referencing Pane

Total number of instances allowed per top model

Specify how many references to this model can occur in another model.

Settings

Default: Multiple

Zero

The model cannot be referenced. An error occurs if a reference to the model occurs in another model.

One

The model can be referenced at most once in a model reference hierarchy. An error occurs if more than one reference exists.

Multiple

The model can be referenced more than once in a hierarchy, provided that it contains no constructs that preclude multiple reference. An error occurs if the model cannot be multiply referenced, even if only one reference exists.

Command-Line Information

Parameter: ModelReferenceNumInstancesAllowed

Type: string

Value: 'Zero' | 'Single' | 'Multi'

Default: 'Multi'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- [Diagnosing Simulation Errors](#)
- [Configuration Parameters Dialog Box](#)
- [Model Referencing Pane](#)

Model dependencies

Specify the files on which this model relies. They are typically MAT-files and M-files used to initialize parameters and to provide data.

Settings

No Default

- Specify the dependencies as a cell array of strings, where each cell array entry is the filename or path of a dependent file. These filenames may include spaces and must include file extensions (e.g., .m, .mat, etc.).
- Prefix the token \$MDL to a dependency to indicate that the path to the dependency is relative to the location of this model file.
- Wildcards are allowed. Use a '%' to comment out a line; use '...' to continue lines.

Tips

- If Simulink software cannot find a specified dependent file when you update or simulate a model that references this model, Simulink software displays a warning.
- The dependencies automatically include the model.mdl and linked library .mdl files.
- For files not on the MATLAB® path, use absolute paths.

Command-Line Information

Parameter: ModelDependencies

Type: string

Value: any valid value

Default: ''

Recommended Settings

Application	Setting
Debugging	No impact

Application	Setting
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Model Dependencies
- Configuration Parameters Dialog Box
- Model Referencing Pane

Pass scalar root inputs by value for Real-Time Workshop

Specify whether a model that calls (references) this model passes this model's scalar inputs by value.

Settings

Default: Off



On

A model that calls (i.e., references) this model passes this model's scalar inputs by value.



Off

The calling model passes the inputs by reference (it passes the addresses of the inputs rather than the input values).

Tips

- Passing roots by value allows this model to read its scalar inputs from register or local memory which is faster than reading the inputs from their original locations.
- However, passing roots by value can lead to incorrect results if the model's root scalar inputs can change within a time step. This can happen, for instance, if this model's inputs and outputs share memory locations (as a result of a feedback loop) and the model is invoked multiple times in a time step (i.e., by a Function-Call Subsystem). In such cases, this model sees scalar input changes that occur in the same time step only if the inputs are passed by reference. That is why this option is off by default.
- If you are certain that this model is not referenced in contexts where its inputs can change within a time step, select this option to generate more efficient code for this model.
- Selecting this option can affect reuse of code generated for subsystems. See Reusable Code and Referenced Models for more information.

Command-Line Information

Parameter: ModelrefPassRootInputsByReference

Type: string

Value: 'on' | 'off'

Default: 'off'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	Off

See Also

- Function-Call Subsystem
- Reusable Code and Referenced Models
- Configuration Parameters Dialog Box
- Model Referencing Pane

Minimize algebraic loop occurrences

Specify whether Simulink software tries to eliminate algebraic loops involving this model from models that reference it.

Settings

Default: Off

On
Simulink software tries to eliminate algebraic loops involving this model from models that reference it.

Off
Simulink software does not try to eliminate algebraic loops from the models that reference this model.

Tips

- Enabling this option disables:
 - Conditional input branch optimization for simulation
 - The Real-Time Workshop software **Single update/output function** optimization for code generation
- See Algebraic Loops for more information.

Command-Line Information

Parameter: ModelrefMinAlgLoopOccurrences

Type: string

Value: 'on' | 'off'

Default: 'off'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	Off

See Also

- Model block
- Algebraic Loops
- Model Blocks and Direct Feedthrough
- Diagnosing Simulation Errors
- Configuration Parameters Dialog Box
- Model Referencing Pane

Library Browser

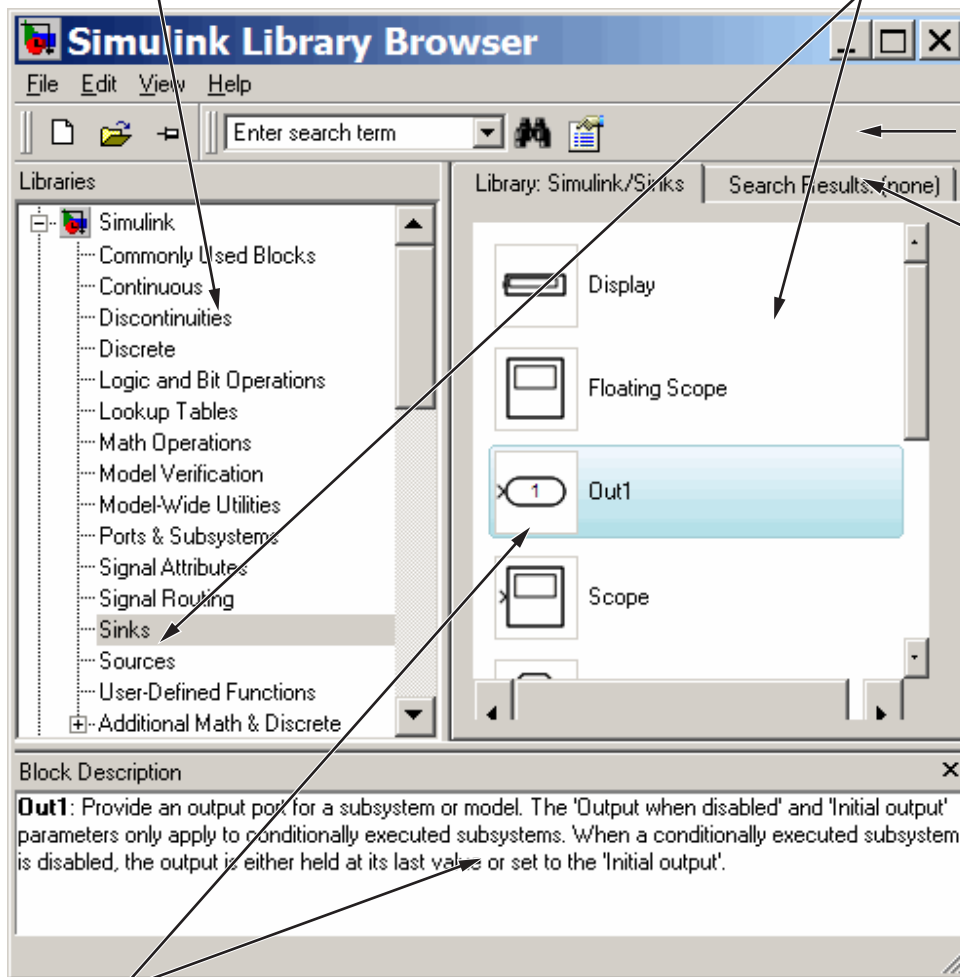
About the Library Browser (p. 2-2)	Overview of the Library Browser
Libraries Pane (p. 2-5)	Using the Library Browser's Libraries pane
Library Pane (p. 2-8)	Using the Library Browser's Library pane.
Block Description Pane (p. 2-11)	Using the Library Browser's Block Description pane.
Search Toolbar (p. 2-12)	Using the Library Browser's Search toolbar.
Found Pane (p. 2-15)	Using the Library Browser's Found pane
Library Data Repository (p. 2-18)	Purpose of the Library Data Repository
Library Browser Keyboard Shortcuts (p. 2-19)	Keyboard shortcuts for navigating the Library Browser and executing Library Browser commands

About the Library Browser

The Library Browser lets you browse and search Simulink® block libraries for blocks to use in your models. The following figure points out the Library Browser's principal features and their usage.

Libraries pane:
use to select libraries for browsing

Library pane:
use to browse selected library



Search toolbar:
use to find blocks by name

Search Results pane: displays blocks found by the Search tool

Block Description pane:
describes selected block

For detailed descriptions of the Library Browser's features and their usage, see the following topics:

- “Libraries Pane” on page 2-5
- “Library Pane” on page 2-8
- “Block Description Pane” on page 2-11
- “Search Toolbar” on page 2-12
- “Found Pane” on page 2-15

For a task-oriented guide to using the Library Browser to browse and search block libraries, see “Browsing and Searching Block Libraries”.

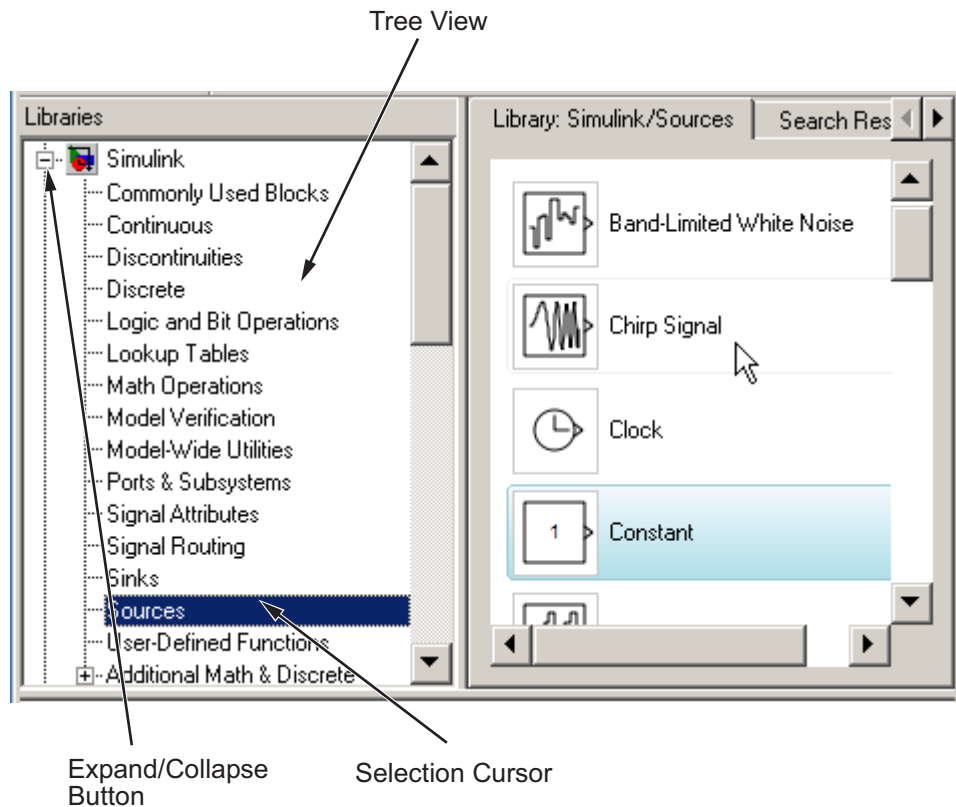
For information on adding your own block libraries to the Library Browser, see “Adding Libraries to the Library Browser”.

Libraries Pane

In this section...
“About the Libraries Pane” on page 2-5
“Selecting Nodes” on page 2-6
“Expanding and Collapsing Nodes” on page 2-7
“Refreshing the Tree View” on page 2-7

About the Libraries Pane

The **Libraries** pane allows you to select block libraries for browsing. The pane displays a tree-structured directory (tree view) of libraries installed on your system, each of whose nodes you can select with your mouse or keyboard. Selecting a node displays the contents of the corresponding library in the Library Browser’s **Library** pane. The **Libraries** pane indicates the selected library via a selection cursor that highlights the corresponding node.



Selecting Nodes

You can use your keyboard or your mouse to select a node in the tree view. To select a node with the mouse, first use the **Libraries** pane's scroll bars, if necessary, to move the node into view. Then click the node. To select a node with the keyboard, use the keyboard's **Up** or **Down** arrow key to move the tree's node selection cursor to the desired node.

Note Selecting a node can cause the Library Browser to update its library data repository (see “Library Data Repository” on page 2-18). In this case, it displays an advisory dialog box while the update is in process.

Expanding and Collapsing Nodes

To facilitate navigation of large directory trees, the **Libraries** pane displays libraries containing sublibraries as expandable nodes, each containing a toggle button labeled + or -, depending on whether the node is collapsed or expanded. Clicking the button expands or collapses its node to show or hide the corresponding parent library's children. You can also expand or collapse the currently selected node by pressing the **Right** or **Left** arrow keys on your keyboard, respectively.

Refreshing the Tree View

To refresh the tree view displayed in the **Libraries** pane, select **Refresh Tree View** from the Library Browser's **View** menu. The Library Browser adds or removes any libraries from the tree view that have been added or deleted, respectively, from the MATLAB® path since the view was last updated.

Library Pane

In this section...

“About the Library Pane” on page 2-8

“Choosing the Library Pane’s Layout” on page 2-8

“Selecting Blocks” on page 2-9

“Creating an Instance of a Library Block in a Model” on page 2-9

“Displaying a Library Block’s Parameters” on page 2-10

“Displaying Help for a Library Block” on page 2-10

“Displaying the Contents of a Sublibrary” on page 2-10

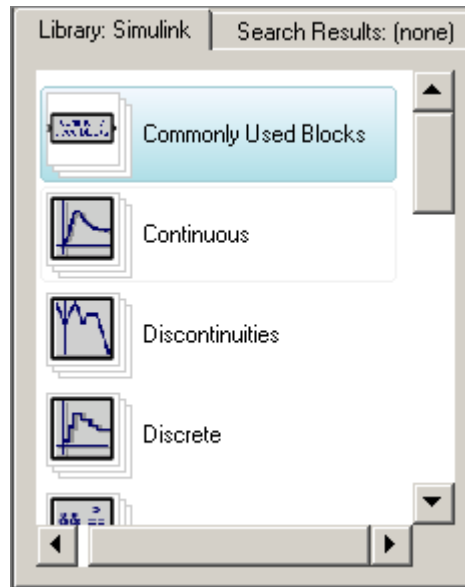
“Displaying a Block or Sublibrary’s Parent” on page 2-10

About the Library Pane

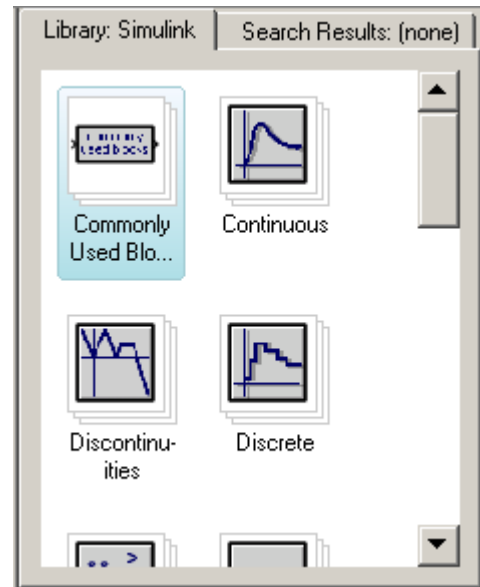
The Library Browser’s **Library** pane allows you to browse the contents of the library selected in the Library Browser’s **Libraries** pane (see “Libraries Pane” on page 2-5). You can use this pane to display the contents of sublibraries, to view a library block’s parameters or help, and to create instances of library blocks in models.

Choosing the Library Pane’s Layout

The Library Browser can display blocks in either a list (single-column) layout or a screen-saving grid layout. To choose a layout, select **List** or **Grid** from the Library Browser’s **View > Layout** menu.



List Layout



Grid Layout

Selecting Blocks

Many **Library** pane procedures require you to select a block displayed in the pane. To select a block, click it with the mouse. The Library Browser moves the block selection cursor to the selected block.

Creating an Instance of a Library Block in a Model

To create an instance of a library block in an existing model, select the block in the **Library** pane and drag it into the model's window.

To create an instance of a library block in a new model, select the block in the **Library** pane and then select **Add to a new model** from the library block's context menu or **Add Selected Block to a New Model** from the Library Browser's **Edit** menu.

Displaying a Library Block's Parameters

To display a library block's parameter dialog box, double-click the block in the **Library** pane or select **Block parameter** from the block's context menu.

Note The dialog box is disabled to prevent you from using it to change a library block's parameters.

Displaying Help for a Library Block

To display help for a library block, select the block and then select **Help** from the block's context menu or from the Library Browser's **Help** menu.

Displaying the Contents of a Sublibrary

To display the contents of a sublibrary appearing in the **Library** pane, double-click the library. The library's contents replaces the parent's contents in the **Library** pane.

Displaying a Block or Sublibrary's Parent

To display the parent of an item (block or library) displayed in the **Library** pane, select **Go to parent** from the item's context menu. The contents of the parent replaces the contents of the child in the **Library** pane.

Block Description Pane

This pane displays a description of the block selected in the **Library** pane. The text of the description is the same as the text of the description that appears on the block's parameter dialog box.

To hide this pane, deselect **Show block descriptions** on the Library Browser's **View** menu.

Search Toolbar

In this section...

“About the Search Toolbar” on page 2-12

“Using the Search Toolbar to Find Blocks” on page 2-12

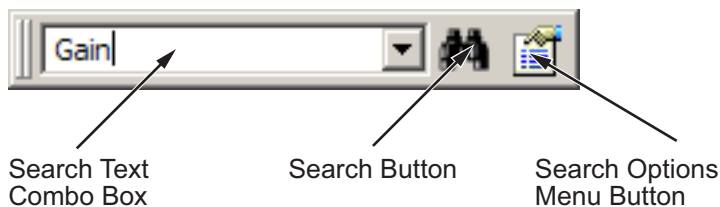
“Search Text Combo Box” on page 2-13

“Search Options Menu” on page 2-13

“Search Button” on page 2-14

About the Search Toolbar

The Library Browser’s **Search** toolbar allows you to search the block libraries installed on your system for blocks whose names contain or match a text string that you specify. The **Search** toolbar displays the blocks it finds in the Library Browser’s **Found** pane. You can use the **Found** pane to view a found block’s help or parameters or create an instance of the block in a model (see “Found Pane” on page 2-15 for more information).



Using the Search Toolbar to Find Blocks

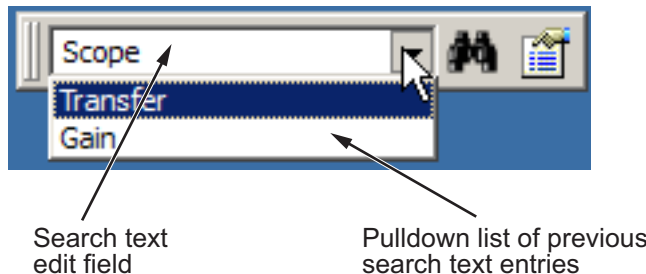
To find blocks whose names contain or match a specified text string:

- 1 Enter the string in the toolbar’s search text combo box (see “Search Text Combo Box” on page 2-13).
- 2 Use the toolbar’s search options menu to specify the search options you want to use, e.g., match whole words (see “Search Options Menu” on page 2-13).
- 3 Select the toolbar’s **Search** button to start the search.

The Library Browser searches the libraries installed on your system whose names contain or match the search string you specified, depending on the options you specified. It displays the results of the search in its **Found** pane (see “Found Pane” on page 2-15).

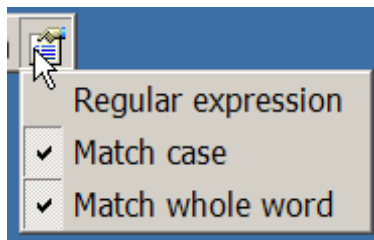
Search Text Combo Box

The search text combo box allows you to specify the text string to be used to search the block libraries on your system. The combo box consists of an edit field and a drop-down list of search strings that you previously entered in the current or previous sessions. To specify a string, enter the string in the edit field or choose a string from the drop-down list.



Search Options Menu

The **Search Options** menu allows you to specify various search options. To display the menu, select the search options menu button on the **Search** toolbar.



Regular Expression

Treat search text as a MATLAB regular expression (see “Regular Expressions”).

Match Case

Consider case when matching search text against block names.

Match Whole Word

Allow a match only if the search string matches a word, i.e., text set off by white space, in a block’s name.

Search Button

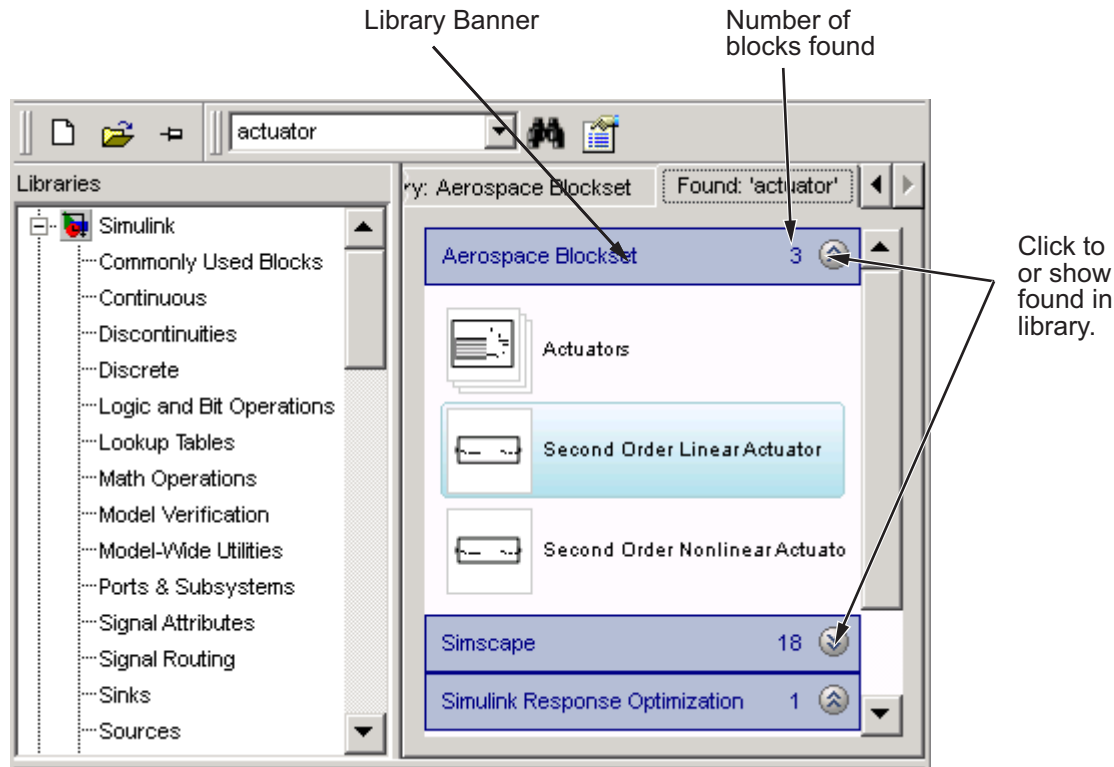
Selecting this button initiates a search of the block libraries on your system, using the text string and search options specified in the toolbar’s search text combo box (see “Search Text Combo Box” on page 2-13) and search options menu (see “Search Options Menu” on page 2-13), respectively.

Found Pane

In this section...
“About the Found Pane” on page 2-15
“Selecting a Found Block in Library View” on page 2-16
“Displaying a Found Block’s Path” on page 2-17

About the Found Pane

The **Found** pane allows you to view and select blocks found by the Library Browser’s **Search** tool. The pane displays the blocks found by the tool grouped by library. A banner at the top of each group displays the name of the top-level library containing the found blocks, the number of blocks found in the library, and a button that allows you to hide or display the search results for the library.



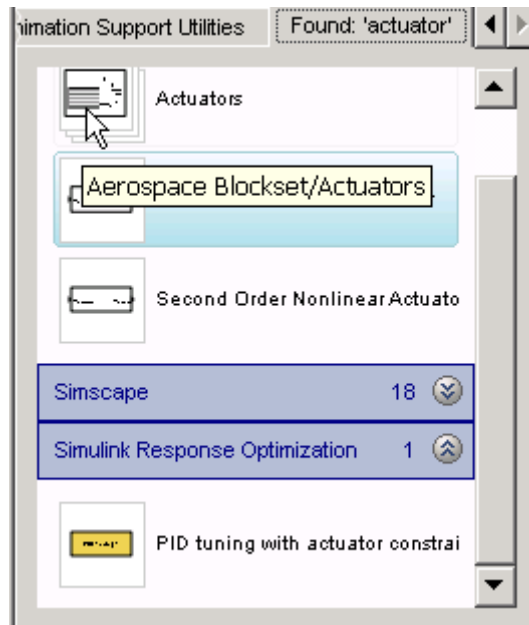
You can use the **Found** pane to perform many of the same tasks you can perform with the **Library** pane, including creating instances of found blocks in models and other libraries, displaying a found block's parameters, and displaying help for a found block. The procedures for these tasks are the same as for the **Library** pane (see “Library Pane” on page 2-8 for more information).

Selecting a Found Block in Library View

To select a found block in the **Library** pane, select the block in the **Found** pane and then select **Select in library view** from the block's context menu or enter **Ctrl+R**. This brings the **Library** pane forward in the Library Browser with the found block in view and selected.

Displaying a Found Block's Path

To display the path of a found block, move the cursor over the block. The block's path appears in a tooltip.



Library Data Repository

To allow you to browse and search libraries without first loading them, the Library Browser maintains a repository of data about each library installed on your system. The data includes block paths and icons. The Library Browser updates the repository when you select a library whose contents have changed since the last time you selected the library.

Library Browser Keyboard Shortcuts

Task	Shortcut
Select Libraries or Library panes or tools on the Search toolbar.	Tab
Switch from the Library pane to the Found pane, or vice versa.	Ctrl+P
Open a model.	Ctrl+O
Move node selection down in the Libraries pane tree view.	Down Arrow
Move node selection up in the Libraries pane tree view.	Up Arrow
Expand a node in the Libraries pane tree view.	Right Arrow
Collapse a node in the Libraries pane tree view.	Left Arrow
Select a block found with the search tool in the Library pane.	Ctrl+R
Insert the selected block in a new model.	Ctrl+I
Increase the Library Browser's font size.	Ctrl++
Decrease the Library Browser's font size	Ctrl+-
Use small block icons.	Ctrl+1
Use medium block icons.	Ctrl+2
Use large block icons.	Ctrl+3
Find a block.	Ctrl+F

Signal Properties Dialog Box

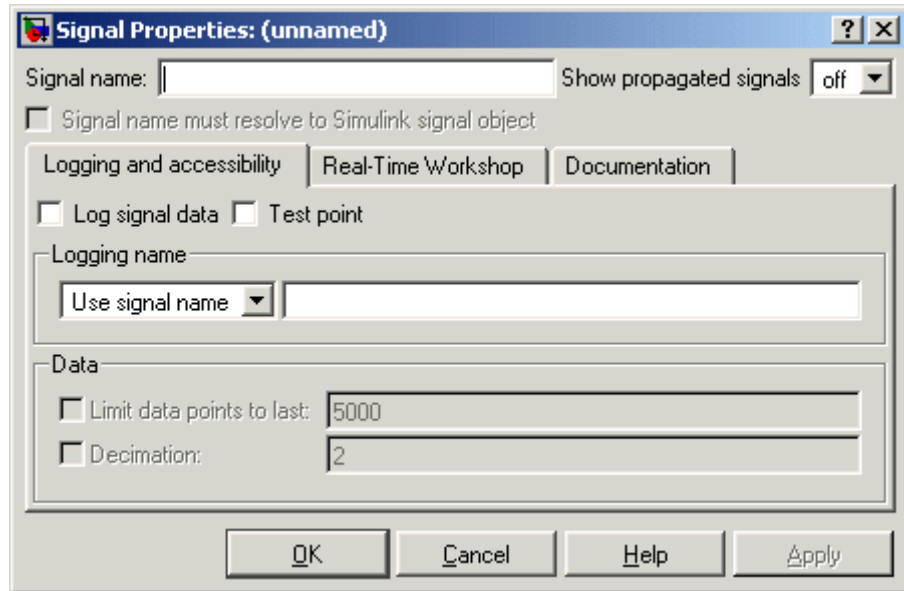
The **Signal Properties** dialog box lets you display and edit signal properties. To display the dialog box, either

- Select the line that represents the signal whose properties you want to set and then choose **Signal Properties** from the signal's context menu or from the Simulink® **Edit** menu

or

- Select a block that outputs or inputs the signal and select **Port Signal Properties** from the block's context menu, then select the port to which the signal is connected from the resulting menu

The **Signal Properties** dialog box appears.



The dialog box includes the following controls.

- “Signal Properties Controls” on page 3-2
- “Logging and Accessibility Options” on page 3-4
- “Real-Time Workshop® Options” on page 3-6
- “Documentation Options” on page 3-7

Signal Properties Controls

Signal name

Name of signal.

Signal name must resolve to Simulink® signal object

Specifies that either the base MATLAB® workspace or the model workspace must contain a `Simulink.Signal` object with the same name as this signal.

Simulink® software displays an error message if it cannot find such an object when you update or simulate the model containing this signal.

Note Simulink.Signal objects in the model workspace must have their storage class set to Auto. See “Using Model Workspaces” for more information.

When **Signal name must resolve to Simulink signal object** is enabled, a signal resolution icon appears by default to the left of any label on the signal. The icon looks like this:



See “Signal Resolution Indicators” for more information.

Show propagated signals

Note This option appears only for signals that originate from a virtual block other than a Bus Selector block.

Show propagated signal names. You can select one of the following options:

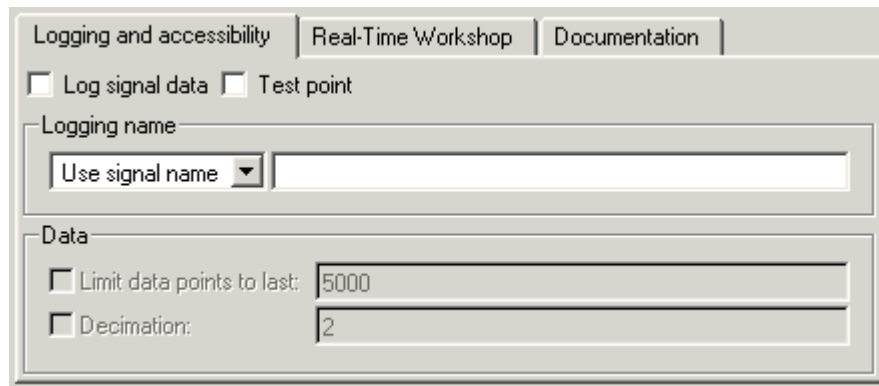
Option	Description
off	Do not display signals represented by a virtual signal in the signal's label.

Option	Description
on	Display the virtual and nonvirtual signals represented by a virtual signal in the signal's label. For example, suppose that virtual signal s1 represents a nonvirtual signal s2 and a virtual signal s3. If this option is selected, the label for s1 is s1<s2, s3>.
all	Display all the nonvirtual signals that a virtual signal represents either directly or indirectly. For example, suppose that virtual signal s1 represents a nonvirtual signal s2 and a virtual signal s3 and virtual signal s3 represents nonvirtual signals s4 and s5. If this option is selected, the label for s1 is s1<s2, s4, s5>.

See “Displaying the Nonvirtual Components of Virtual Signals” for more information.

Logging and Accessibility Options

Select the **Logging and accessibility** tab on the **Signal Properties** dialog box to display controls that enable you to specify signal logging and accessibility options for this signal.



Log signal data

Select this option to cause Simulink® software to save this signal’s values to the MATLAB® workspace during simulation (see “Logging Signals”).

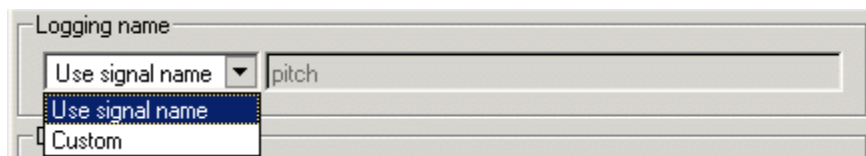
Test point

Select this option to designate this signal as a test point (see “Working with Test Points”).

Note If you select the **Log signal data** option for this signal, Simulink software selects and disables the **Test point** option so that you cannot deselect it. This is because a signal must be a test point to be logged.

Logging name

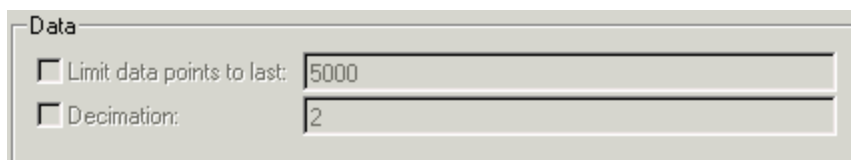
This pair of controls, consisting of a list box and an edit field, specifies the name associated with logged signal data.



Simulink software uses the signal’s signal name as its logging name by default. To specify a custom logging name, select Custom from the list box and enter the custom name in the adjacent edit field.

Data

This group of controls enables you to limit the amount of data that Simulink software logs for this signal.



The options are as follows.

Limit data points to last

Discard all but the last N data points where N is the number entered in the adjacent edit field.

Decimation

Log every Nth data point where N is the number entered in the adjacent edit field. For example, suppose that your model uses a fixed-step solver with a step size of 0.1 s. If you select this option and accept the default decimation value (2), Simulink software records data points for this signal at times 0.0, 0.2, 0.4, etc.

Real-Time Workshop® Options

The following controls set properties used by Real-Time Workshop® to generate code from the model. You can ignore them if you are not going to generate code from the model.

RTW storage class

Select the storage class of this signal from the list. See “Interfacing Signals to External Code” for an explanation of the listed options.

RTW storage type qualifier

Enter a storage type qualifier for this signal. See “Interfacing Signals to External Code” for more information.

Documentation Options

Description

Enter a description of the signal in this field.

Document link

Enter a MATLAB® expression in the field that displays documentation for the signal. To display the documentation, click “Document Link.” For example, entering the expression

```
web(['file:/// ' which('foo_signal.html')])
```

in the field causes MATLAB software’s default Web browser to display `foo_signal.html` when you click the field’s label.

3 Signal Properties Dialog Box

Simulink[®] Preferences Window

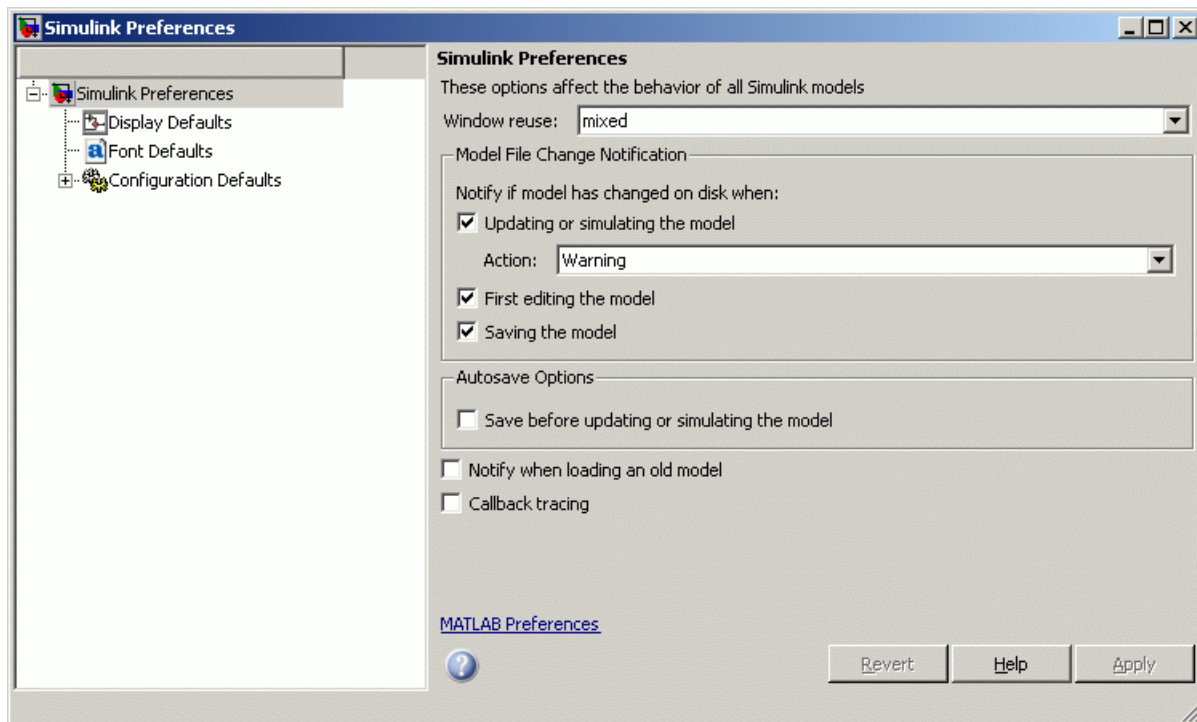
Simulink[®] Preferences Window:
Main Pane (p. 4-2)

Parameters for controlling window reuse, file change and old version notifications, autosave, and callback tracing.

Simulink[®] Preferences Window:
Display Defaults Pane (p. 4-16)

Parameters for controlling display options for the Model Browser, block connection lines and port data types.

Simulink® Preferences Window: Main Pane



In this section...

“Simulink® Preferences Window Overview” on page 4-3

“Window Reuse” on page 4-4

“Updating or simulating the model” on page 4-6

“Action” on page 4-7

“First editing the model” on page 4-9

“Saving the model” on page 4-10

“Save before updating or simulating the model” on page 4-11

“Notify when loading an old model” on page 4-14

“Callback tracing” on page 4-15

Simulink® Preferences Window Overview

Set preferences for configuring window reuse, file change and old version notifications, autosave, and callback tracing.

Configuration

To open the Simulink® Preferences window do one of the following:

- From a Simulink model, select **File > Preferences**.
- From MATLAB®, the Simulink library browser, or the Help Browser, select **File > Preferences**, then select **Simulink** in the tree, and click the button **Launch Simulink Preferences**.

- 1 Select the check boxes to configure preferences.
- 2 Close the window to apply your changes.

Click **Apply** to apply your changes and keep the window open.

Your settings affect the behavior of all Simulink models, including those currently open, and all subsequent models. Your preference settings are preserved for the next time you use the software.

See Also

Model File Change Notification

Window Reuse

Specify whether to use the current window or open new windows to display a subsystem or its parent.

Settings

Default: Mixed

Mixed

Mixed window reuse. Subsystem opens in its own window. When you press **Escape** (go to parent), the parent window rises to the front, and the subsystem window disappears.

None

Never reuse windows. Subsystem opens in a new window. Press **Escape**: the parent window moves to the front.

Replace

Replace parent window. Subsystem opens in a new window. Parent window disappears. Press **Escape**: the parent window appears, and the subsystem window disappears.

Reuse

Reuse parent window. Subsystem replaces the parent in the current window. Press **Escape**: the parent window replaces subsystem in current window.

Tips

- Reusing windows avoids cluttering your screen.
- Creating a window for each subsystem allows you to view subsystems side by side with their parents or siblings.
- The **WindowReuse** preference does not affect the behavior of library windows.

Command-Line Information

Parameter: WindowReuse

Type: string

Value: 'none' | 'reuse' | 'replace' | 'mixed'

Default: 'mixed'

Updating or simulating the model

Specify whether to notify if the model has changed on disk when updating or simulating the model.

Settings

Default: On



On

Notify if the model has changed on disk when updating or simulating the model. Select the action to take in the **Action** list.



Off

Do not notify if the model has changed on disk when updating or simulating the model.

Tip

To programmatically check whether the model has changed on disk since it was loaded, use the function `slIsFileChangedOnDisk`.

Dependency

This parameter enables **Action**.

Command-Line Information

Parameter: `MDLFileChangedOnDiskChecks`

Type: struct, field name: `CheckWhenUpdating`

Value: `true | false | 1 | 0`

Default: `true`

See Also

Model File Change Notification

Action

Select what action to take if the file has changed on disk since it was loaded.

Settings

Default: Warning

Warning

Displays a warning in MATLAB command window

Error

Displays an error, at the MATLAB command window if simulating from the command line, or if simulating from a menu item, in the Simulation Diagnostics window.

Reload model (if unmodified)

Reloads if the model is unmodified. If the model is modified, you see the prompt dialog.

Show prompt dialog

Shows prompt dialog. In the dialog, you can choose to close and reload, or ignore the changes.

Tip

To programmatically check whether the model has changed on disk since it was loaded, use the function `slIsFileChangedOnDisk`.

Dependencies

This parameter is enabled by the parameter **Updating or simulating the model**.

Command-Line Information

Parameter: `MdlFileChangedOnDiskHandling`

Type: string

Value: 'Warning' | 'Error' | 'Reload model (if unmodified)' | 'Show prompt dialog'

Default: 'Warning'

See Also

Model File Change Notification

First editing the model

Specify whether to notify if the file has changed on disk when editing the model.

Settings

Default: On



On

Displays a warning if the file has changed on disk when you modify the block diagram. Any graphical operation that modifies the block diagram (e.g., adding a block) causes a warning dialog to appear. Any command-line operation that causes the block diagram to be modified (e.g., a call to `set_param`) will result in a warning like this at the command line:

```
Warning: Block diagram 'mymodel' is being edited but file has
changed on disk since it was loaded. You should close and
reload the block diagram.
```



Off

Do not check for changes on disk when first editing the model.

Tip

To programmatically check whether the model has changed on disk since it was loaded, use the function `slIsFileChangedOnDisk`.

Command-Line Information

Parameter: `MDLFileChangedOnDiskChecks`

Type: struct, field name: `CheckWhenEditing`

Value: `true` | `false` | `1` | `0`

Default: `true`

See Also

Model File Change Notification

Saving the model

Specify whether to notify if the file has changed on disk when saving the model.

Settings

Default: On



On

Notify if the file has changed on disk when you save the model.

- The `save_system` function displays an error, unless the `OverwriteIfChangedOnDisk` option is used.
- Saving the model by using the menu (**File > Save**) or a keyboard shortcut causes a dialog to be shown. In the dialog, you can choose to overwrite, save with a new name, or cancel the operation.



Off

Do not check for changes on disk when saving the model.

Tip

To programmatically check whether the model has changed on disk since it was loaded, use the function `slIsFileChangedOnDisk`.

Command-Line Information

Parameter: `MDLFileChangedOnDiskChecks`

Type: struct, field name: `CheckWhenSaving`

Value: `true` | `false` | `1` | `0`

Default: `true`

See Also

Model File Change Notification

Save before updating or simulating the model

Specify whether to automatically save a backup copy of the model before updating or simulating.

Settings

Default: On



On

If the model has unsaved changes, automatically save a backup copy of the model before updating or simulating. This autosave copy can be useful for crash recovery.

The copy is saved in the same directory as the model, with the name *MyModel.mdl.autosave*.



Off

Do not automatically save a copy before updating or simulating.

Tips

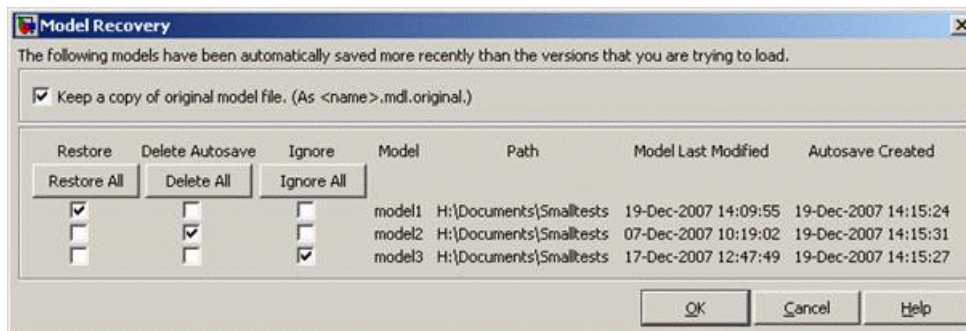
If you open or load a model with a more recent autosave copy available, the model loads, and the non-modal dialog Model Recovery appears.

For each model in the list, you can choose to:

- **Restore** — Overwrite the original model file with the autosave copy, and delete the autosave copy.

If you select the check box to **Keep a copy of original model file**, you can save copies of the original model files named *MyModel.mdl.original*.

- **Delete Autosave** — Delete the autosave copy, and do nothing else.
- **Ignore** — Do nothing. This leaves the model and the autosave copy untouched. The Model Recovery dialog will reappear the next time you open the model, so you can choose to restore or delete autosave files later.



Select a check box for each model in the list to specify whether to restore, delete autosave, or ignore. You can click the **Restore All**, **Delete All** or **Ignore All** buttons to select that option for all models.

The default option is **Ignore All**. If you click **OK** without making any changes, your models and autosave copies are left untouched. The next time you open the model, the Model Recovery dialog will reappear and you can choose again whether to restore or delete any autosave copies.

Caution If you restore, any changes made since your last save will be lost.

If you ignore any model recovery options and continue to work with the model as loaded, any update or simulate will cause the model recovery information to be lost. If you then subsequently attempt to restore the model, the restore will fail, and any changes made since your last save will be lost.

If a segmentation violation occurred, note that the last autosave file for the model reflects the state of the autosave data prior to the segmentation violation. Because Simulink models might be corrupted by a segmentation violation, a model is not autosaved after a segmentation violation occurs.

Click **OK** to apply your choices.

Autosave has the following rules:

- If you deliberately close a modified model, any autosave copy is deleted.

- Autosave does not occur for models that are part of the MATLAB installation, so you will not create autosave copies of the demo models.
- Autosave does not occur if the autosave file or location is read only.

Command-Line Information

Parameter: AutoSaveOptions

Type: struct, field name: SaveOnModelUpdate

Value: true | false | 1 | 0

Default: true

Notify when loading an old model

Specify whether to notify when loading a model last saved in a older version of Simulink software.

Settings

Default: Off

On

Print a message in the command window when loading a model last saved in an old version of Simulink software.

Off

No notification when loading old models.

Tips

- Run `slupdate('modelName')` to convert the block diagram to the format of the current version of Simulink software.
- For advice on upgrading a model to the current version of Simulink software, see “Consulting the Model Advisor”.

Command-Line Information

Parameter: `NotifyIfLoadOldModel`

Type: string

Value: 'on' | 'off'

Default: off

Callback tracing

Specify whether to display the model callbacks that Simulink software invokes when simulating a model.

Settings

Default: Off



On

Display the model callbacks in the MATLAB command window as they are invoked.

Callback tracing allows you to determine the callbacks the software invokes, and in what order, when you open or simulate a model.



Off

Do not display model callbacks.

Command-Line Information

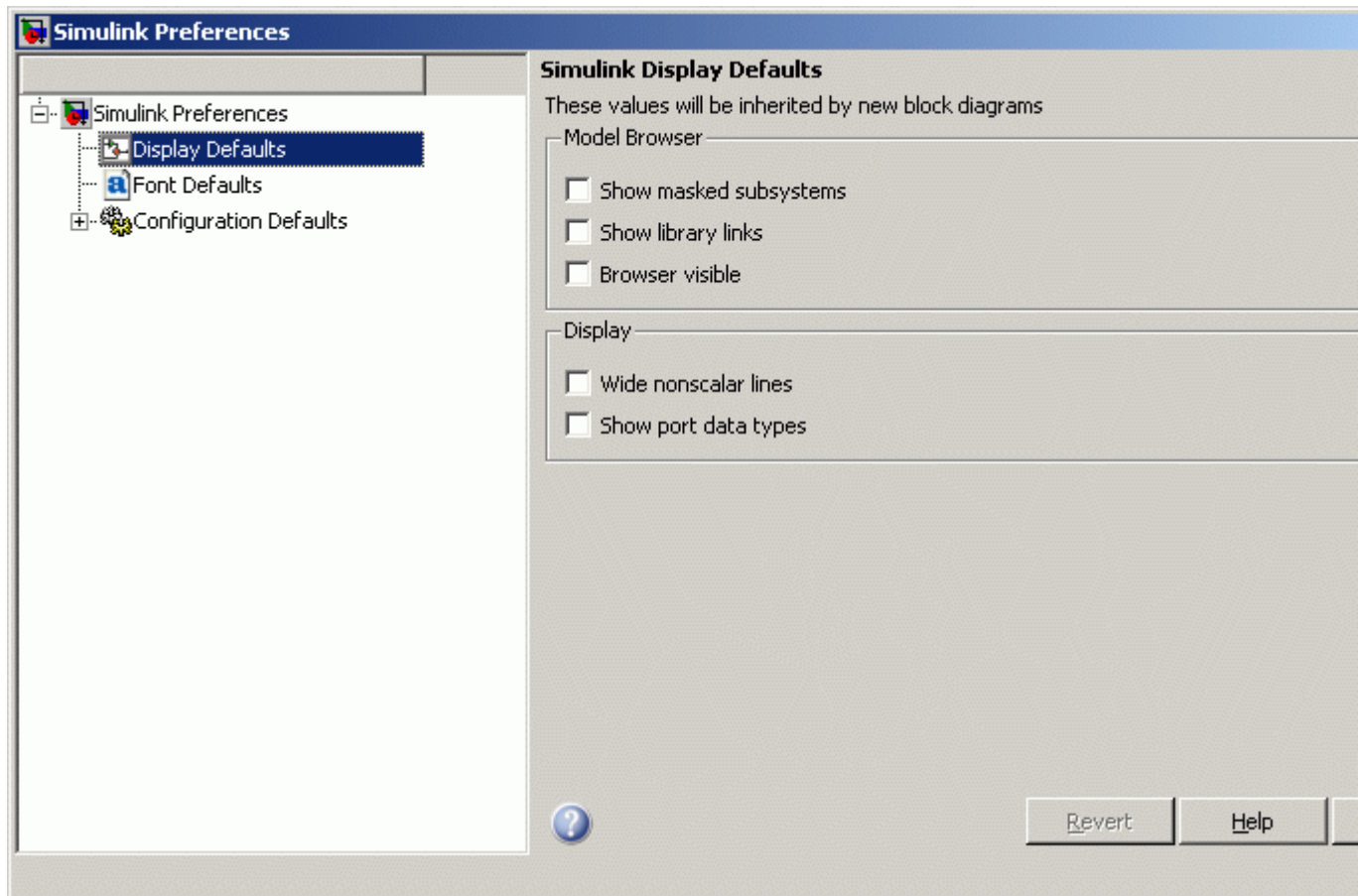
Parameter: CallbackTracing

Type: string

Value: 'on' | 'off'

Default: 'off'

Simulink® Preferences Window: Display Defaults Pane



In this section...

“Simulink® Display Defaults Overview” on page 4-17

“Show masked subsystems” on page 4-18

“Show library links” on page 4-19

“Browser visible” on page 4-20

In this section...

“Wide nonscalar lines” on page 4-21

“Show port data types” on page 4-22

Simulink® Display Defaults Overview

Configure display options for the Model Browser, block connection lines and port data types.

Configuration

- 1 Select check boxes to configure display properties that will be applied to all new block diagrams.
- 2 Close the window to apply your changes.

Click **Apply** to apply your changes and keep the window open. These values will be inherited by new block diagrams.

See Also

- “The Model Browser” (Windows® only)
- Signal Display Options

Show masked subsystems

Specify whether masked subsystems and their contents are shown in the Model Browser (Windows only).

Settings

Default: Off



On

Display masked subsystems and their contents in the Model Browser.



Off

Do not display masked subsystems and their contents in the Model Browser.

Command-Line Information

Parameter: BrowserLookUnderMasks

Type: string

Value: 'on' | 'off'

Default: 'off'

Show library links

Specify whether library links and their contents are shown in the Model Browser (Windows only).

Settings

Default: Off



On

Display library links and their contents in the Model Browser.



Off

Do not display library links and their contents in the Model Browser.

Command-Line Information

Parameter: BrowserShowLibraryLinks

Type: string

Value: 'on' | 'off'

Default: 'off'

Browser visible

Specify whether the Model Browser (Windows only) is shown when you open a model.

Settings

Default: Off



On

Display the Model Browser.



Off

Do not display the Model Browser.

Command-Line Information

Parameter: ModelBrowserVisibility

Type: string

Value: 'on' | 'off'

Default: 'off'

Wide nonscalar lines

Specify whether to show thick lines for nonscalar connections between blocks.

Settings

Default: Off

On
Show thick lines for nonscalar connections between blocks

Off
Do not show thick lines for nonscalar connections between blocks

Command-Line Information

Parameter: WideVectorLines

Type: string

Value: 'on' | 'off'

Default: 'off'

Show port data types

Specify whether to show the data type on each block port

Settings

Default: Off



On

Display the data type for each port on each block.



Off

Do not display data types on block ports.

Command-Line Information

Parameter: ShowPortDataTypes

Type: string

Value: 'on' | 'off'

Default: 'off'