

Release Notes for Stateflow®

How to Contact MathWorks



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



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



508-647-7000 (Phone)



508-647-7001 (Fax)



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

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

Release Notes for Stateflow[®]

© COPYRIGHT 2000–2012 by The MathWorks, Inc.

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

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

Trademarks

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

Patents

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

R2012b

New editor for Stateflow charts and Simulink models with tabbed windows and model browser tree	2
Editing assistance through smart guides, drag margins, transition indicator lines, and just-in-time error notifications	7
State transition tables that provide tabular interface to model state machines	8
MATLAB language for state and transition labels with chart syntax auto-correction	9
In-chart debugging with visual breakpoints and datatips	10
Reuse of graphical functions with atomic boxes	14
Fewer restrictions for converting states to atomic subcharts	15
Diagnostic for undirected local event broadcasts	16
Diagnostic for transition action specified before condition action	17
Parentheses to identify function-call output events on chart and truth table block icons	19
Resolution of qualified state and data names	20
Support for simulating charts in a folder that has the # symbol on 32-bit Windows platforms	21
Mac screen menubar enabled when Stateflow is installed	22
Option to print charts to figure windows no longer available	23
End of Broadcast breakpoint no longer available for input events	24

R2012a

API Method for Highlighting Chart Objects	26
API Method for Finding Transitions That Terminate on States, Boxes, or Junctions	27

API Property That Specifies the Destination Endpoint of a Transition	28
Structures and Enumerated Data Types Supported for Inputs and Outputs of Exported Graphical Functions ..	29
Mappings Tab in Atomic Subchart Properties Dialog Lists All Valid Scopes	30
Full Decision Coverage When Suppressing Default Cases in the Generated Code	31
Specification of Custom Header Files in the Configuration Parameters Dialog Box Required for Enumerated Types	32
Removal of 'Use Strong Data Typing with Simulink I/O' in a Future Release	33

R2011b

Chart Property to Control Saturation for Integer Overflow	36
Enhanced User Interface for Logging Data and States ...	37
Control of Default Case Generation for Switch-Case Statements in Generated Code	38
Detection of State Inconsistency Errors at Compile Time Instead of Run Time	40
Ability to Model Persistent Output Data for Mealy and Moore Charts	41
Control of Diagnostic for Multiple Unconditional Transitions from One Source	42
MEX Compilation with Microsoft Windows SDK 7.1 Now Supported	43
Simulation Supported When the Current Folder Is a UNC Path	44
Removal of the Coverage Tab from the Stateflow Debugger	45
Test Point Selection Moved to the Logging Tab in Properties Dialog Boxes	46

R2011a

Migration of Stateflow Coder Features to New Product ...	48
--	----

Embedded MATLAB Functions Renamed as MATLAB	
Functions in Stateflow Charts	49
Use of MATLAB Expressions to Specify Data Size	50
Ability to Change Data Values While Debugging	52
Ability to Debug a Single Chart When Multiple Charts	
Exist in a Model	53
Support for Input Events in Atomic Subcharts	55
Control of Generated Function Names for Atomic	
Subcharts	56
Enhanced Data Sorting in the Stateflow Debugger	57
Option to Maintain Highlighting of Active States After	
Simulation	58
Right-Click Options for Setting Local Breakpoints	59
New Signal Logging Format That Simplifies Access to	
States and Local Data	60
Support for Buses in Data Store Memory	61
Enhanced Readability of State Functions	62
Support for Arrays of Buses as Inputs and Outputs of	
Charts and Functions	63
Default Setting of 'States When Enabling' Chart Property	
Now Held	64
Initial Value Vectors with Fixed-Point or Enumerated	
Values Now Evaluate Correctly	65
Mac Screen Menubar Disabled When Stateflow Is	
Installed	66

R2010bSP2

R2010bSP1

R2010b

New Atomic Subcharts to Create Reusable States for	
Large-Scale Modeling	72

Stateflow Library Charts Now Support Instances with Different Data Sizes, Types, and Complexities	73
Support for Controlling Stateflow Diagnostics in the Configuration Parameters Dialog Box	74
Enhanced Custom-Code Parsing to Improve Reporting of Unresolved Symbols	75
Temporal Logic Conditions Can Now Guard Transitions Originating from Junctions	76
Data Dialog Box Enhancements	77
Branching of Function-Call Output Events No Longer Requires Binding of Event to a State	78
Passing Real Values to Function Inputs of Complex Type Disallowed	79
Using Chart Block That Accesses Global Data in For Each Subsystem Disallowed	80
New and Enhanced Demos	81

R2010a

Support for Combining Actions in State Labels	84
New Diagnostic Detects Unused Data and Events	85
Enhanced Support for Variable-Size Chart Inputs and Outputs	86
Support for Chart-Level Data with Fixed-Point Word Lengths Up to 128 Bits	87
New 'States When Enabling' Property for Charts with Function-Call Input Events	88
Support for Tunable Structures of Parameter Scope in Charts	89
Enhanced Real-Time Workshop Code Generation for Noninlined State Functions	90
Enhanced Real-Time Workshop Code Generation for sizeof Function	91
Enhanced Real-Time Workshop Code Generation for Custom-Code Function Calls	92
Data Change Implicit Event No Longer Supports Machine-Parented Data	93
Support for Machine-Parented Events Completely Removed	94
MEX Compilation with Microsoft Visual Studio .NET 2003 No Longer Supported	95

Code Generation Status Messages No Longer Shown in Command Window	96
Change in Behavior for Appearance of Optimization Parameters	97
Enhanced Inlining of Generated Code That Calls Subfunctions	98
Check Box for 'Treat as atomic unit' Now Always Selected	99
New Demos	100

R2009bSP1

R2009b

Ability to Copy Simulink Function-Call Subsystems and Paste in Stateflow Editor as Simulink Functions, and Vice Versa	104
Ability to Generate Switch-Case Statements for Flow Graphs and Embedded MATLAB Functions Using Real-Time Workshop Embedded Coder Software	105
Support for Creating Switch-Case Flow Graphs Using the Pattern Wizard	106
Support for Using More Than 254 Events in a Chart	107
Improved Panning and Selection of States and Transitions When Using Stateflow Debugger	108
Stateflow Compilation Status Added to Progress Indicator on Simulink Status Bar	109
Support for Chart Inputs and Outputs That Vary in Dimension During Simulation	110
New Compilation Report for Embedded MATLAB Functions in Stateflow Charts	111
Enhanced Support for Replacing Math Functions with Target-Specific Implementations	112
Enhanced Context Menu Support for Adding Flow Graph Patterns to Charts	113
Option to Log Chart Signals Available in the Stateflow Editor	114

Default Precision Set to Double for Calls to C Math Functions	115
Change in Text and Visibility of Parameter Prompt for Easier Use with Fixed-Point Advisor and Fixed-Point Tool	116
Charts Closed By Default When Opening Models Saved in Formats of Earlier Versions	117
New and Enhanced Demos	118

R2009a

Support for Saving the Complete Simulation State at a Specific Time	120
Enhanced Support for Enumerated Data Types	121
New Boolean Keywords in Stateflow Action Language ...	122
Enhanced Control of Inlining State Functions in Generated Code	123
New Diagnostic to Detect Unintended Backtracking Behavior in Flow Graphs	124
Use of Basic Linear Algebra Subprograms (BLAS) Libraries for Speed	125
Enhanced Support for Replacing C Math Functions with Target-Specific Implementations	126
Smart Transitions Now Prefer Straight Lines	127
Clicking Up-Arrow Button in the Stateflow Editor Closes Top-Level Chart	128
Enhanced Type Resolution for Symbols	129
Enhanced Code Generation for Stateflow Events	130
Enhanced Real-Time Workshop Generated Code for Charts with Simulink Functions	131
Use of en, du, ex, entry, during, and exit for Data and Event Names Being Disallowed in a Future Version	132
Support for Machine-Parented Events Being Removed in a Future Version	133

R2008b

Support for Embedding Simulink Function-Call Subsystems in a Stateflow Chart	136
--	-----

Support for Using Enumerated Data Types in a Stateflow Chart	137
New Alignment, Distribution, and Resizing Commands for Stateflow Charts	138
Unified Simulation and Embeddable Code Generation Options for Stateflow Charts and Truth Table Blocks ..	139
New Pattern Wizard for Consistent Creation of Logic Patterns and Iterative Loops	173
Support for Initializing Vectors and Matrices in the Data Properties Dialog Box	174
Change in Default Mode for Ordering Parallel States and Outgoing Transitions	175
Optimized Inlining of Code Generated for Stateflow Charts	176
More Efficient Parsing for Nonlibrary Models	177
Change in Casting Behavior When Calling MATLAB Functions in a Chart	178
Ability to Specify Continuous Update Method for Output Data	179
Use of Output Data with Change Detection Operators Disallowed for Initialize-Outputs-at-Wakeup Mode	180
Parsing a Stateflow Chart Without Simulation No Longer Detects Unresolved Symbol Errors	181
Generation of a Unique Name for a Copied State Limited to States Without Default Labels	182
New Configuration Set Created When Loading Nonlibrary Models with an Active Configuration Reference	183

R2008a+

R2008a	
Support for Data with Complex Data Types	188
Support for Functions with Multiple Outputs	189
Bidirectional Traceability for Navigating Between Generated Code and Stateflow Objects	190
New Temporal Logic Notation for Defining Absolute Time Periods	191

New temporalCount Operator for Counting Occurrences of Events	192
Using a Specific Path to a State for the in Operator	193
Enhanced MISRA C Code Generation Support	194
Enhanced Folder Structure for Generated Code	195
Code Optimization for Simulink Blocks and Stateflow Charts	196
New fitToView Method for Zooming Objects in the Stateflow Editor	197
Generation of a Unique Name for a Copied State	198
New Font Size Options in the Stateflow Editor	199
New Fixed-Point Details Display in the Data Properties Dialog Box	200
“What’s This?” Context-Sensitive Help Available for Simulink Configuration Parameters Dialog	201
Specifying Scaling Explicitly for Fixed-Point Data	202
Use of Data Store Memory Data in Entry Actions and Default Transitions Disallowed for Execute-at-Initialization Mode	203
Enhanced Warning Message for Target Hardware That Does Not Support the Data Type in a Chart	204
Detection of Division-By-Zero Violations When Debugger Is Off	205

R2007b+

R2007b

Enhanced Continuous-Time Support with Zero-Crossing Detection	210
New Super Step Feature for Modeling Asynchronous Semantics	211
Support for Inheriting Data Properties from Simulink Signal Objects Via Explicit Resolution	212
Common Dialog Box Interface for Specifying Data Types in Stateflow Charts and Simulink Models	214
Support for Animating Stateflow Charts in Simulink External Mode	215

Support for Target Function Library	217
Support for Fixed-Point Parameters in Truth Table Blocks	218
Support for Using Custom Storage Classes to Control Stateflow Data in Generated Code	219
Loading 2007b Stateflow Charts in Earlier Versions of Simulink Software	220
Bug Fixed for the History Junction	221

R2007a+

R2007a

New Operators for Detecting Changes in Data Values	226
Elimination of “goto” Statements from Generated Code ..	227

R2006b

Support for Mealy and Moore Charts	230
New Structure Data Type Provides Support for Buses ...	231
Custom Integer Sizes	232

R2006a+

No New Features or Changes

R2006a

Option to Initialize Outputs When Chart Wakes Up	236
Ability to Customize the Stateflow User Interface	237

Using the MATLAB Workspace Browser for Debugging	
Stateflow Charts	238
Chart and Truth Table Blocks Require C Compiler for	
64-Bit Windows Operating Systems	239

R14SP3

Data Handling	242
Truth Table Enhancements	243
API Enhancements	244
Greater Usability	245

R14SP2

User-Specified Transition Execution Order	252
Enhanced Integration of Stateflow Library Charts with	
Simulink Models	253
Stateflow Charts and Embedded MATLAB Functions	
Support Simulink Data Type Aliases	254
Fixed-Point Override Supported for Library Charts	255

R2012b

Version: 8.0
New Features: Yes
Bug Fixes: Yes

New editor for Stateflow charts and Simulink models with tabbed windows and model browser tree

The new editor unifies Stateflow® and Simulink® functionality. The Stateflow Editor shares most of the same menu items with the Simulink Editor, and provides the following enhancements:

- **Unified canvas**, for editing Stateflow charts and Simulink models in the same window.
- **Tabbed windows**, for accessing Stateflow charts in the same context as Simulink models.
- **Model Browser tree**, for browsing the complete model hierarchy, including Stateflow charts.
- **Cross-platform consistency**, for accessing the same functionality on Windows®, UNIX®, and Mac platforms.

Stateflow Editor menu bar changes

Menu bar changes in the new Stateflow Editor are the same as for the new Simulink Editor.

Stateflow Editor context menu changes

All changes for the following three Stateflow Editor context menus are the same as for the new Simulink Editor:


- From the canvas
- From a block
- From a signal

The following sections describe changes to context menus that are specific to Stateflow:

- “From a chart” on page 3
- “From a function” on page 4
- “From a transition” on page 4

- “From a state” on page 4

From a chart

R2012a Stateflow Editor Context Menu	New Stateflow Editor Equivalent
Patterns	Add Patterns.
Add Note	Not available from context menu. From the palette, select the Annotation icon ().
Cut Copy	Not available from context menu. Use the context menu for an item in the chart that you want to cut or copy.
Back Forward Go To Parent	Not available from context menu. From the menu bar, use View > Navigate.
Execution Order	Not available from context menu. From the menu bar, use Chart > Parameters.
Font Size Arrowhead Size	Not available from context menu. From the menu bar, use Chart > Format.
Format > Align Items Format > Distribute Items Format > Resize Items	Not available from context menu. From the menu bar, use Chart > Arrange.
Fit To View	Not available from context menu. From the menu bar, use View > Zoom.

R2012a Stateflow Editor Context Menu	New Stateflow Editor Equivalent
Breakpoints	Set Breakpoints on Chart Entry and Clear Breakpoints.
Debug	Not available from context menu. From the menu bar, use Simulation > Debug.
Find	Not available from context menu. From the menu bar, use Edit > Find.
Edit Library	Library Link.

From a function

In the R2012a Stateflow Editor, right-clicking a Stateflow function displays the chart context menu. In the new Stateflow Editor, each kind of Stateflow function (for example, graphical function or Truth Table) has its own context menu.

From a transition

In the R2012a Stateflow Editor, right-clicking a Stateflow transition displays the chart context menu. In the new Stateflow Editor, a transition has its own context menu.

There is no longer a **Smart** menu item. In the R2012a Stateflow Editor, the **Smart** menu item enabled smart transitions. Smart transitions have ends that slide around the surfaces of states and junctions. When the source and/or destination objects are moved and resized in the chart, these transitions use sliding and other behaviors to enable you to produce an aesthetically pleasing chart. The new Stateflow Editor uses smart transitions all the time.

From a state

In the R2012a Stateflow Editor, right-clicking a Stateflow state displays the chart context menu. In the new Stateflow Editor, a state has its own context menu.

Stateflow keyboard and mouse shortcut changes

The new Simulink Editor and Stateflow Editor use the same navigation shortcuts.

Task	R2012a Stateflow Editor Shortcut	New Stateflow Editor Equivalent
Display the parent of the currently displayed chart or subchart.	.. (two periods)	To navigate to the parent, use the up arrow on the toolbar or use the Escape key.
Zoom in by an incremental amount.	+ or r or R	Use mouse scroll wheel or Ctrl++ (the plus sign)
Zoom out by an incremental amount.	- or v or V	Use mouse scroll wheel or Ctrl+- (the minus sign).
Fit chart to screen.	0 or Space Bar	Use Space Bar or from the menu bar, use View > Zoom > Fit To View .
Zoom to normal view.	1	Alt+1
Move the current view down within the full chart.	2	Use the Stateflow Editor scroll bars.
Move the current view down and right within the full chart.	3	Use the Stateflow Editor scroll bars.
Move the current view left within the full chart.	4	Use the Stateflow Editor scroll bars.
Fit the currently selected object to full view. If no object is selected, the chart is fit to full view.	5	Not yet supported

Task	R2012a Stateflow Editor Shortcut	New Stateflow Editor Equivalent
Move the current view right within the full chart.	6	Use the Stateflow Editor scroll bars.
Move the current view up and left within the full chart.	7	Use the Stateflow Editor scroll bars.
Move the current view up within the full chart.	8	Use the Stateflow Editor scroll bars.
Move the current view up and right within the full chart.	9	Use the Stateflow Editor scroll bars.

Editing assistance through smart guides, drag margins, transition indicator lines, and just-in-time error notifications

The new Stateflow Editor makes it easier to create and modify charts by providing these enhancements:

- **Smart guides**, for aligning objects interactively as you place them in the chart.
- **Drag margins**, which allows all objects within a container to move together. The mouse cursor changes to a double-arrow when you are within the drag margins of an object.
- **Transition indicator lines**, for identifying the label associated with a selected transition.
- **Just-in-time error notification**, for flagging illegal object placement during editing (for example, when two states overlap).

State transition tables that provide tabular interface to model state machines

A state transition table is an alternative way of expressing modal logic. Instead of drawing states and transitions graphically in a Stateflow chart, you express the modal logic in tabular format. Stateflow automatically generates a graphical state chart from the tabular format, so you can use animation and in-chart debugging (described in “In-chart debugging with visual breakpoints and datatips” on page 10).

Benefits of using state transition tables include:

- Ease of modeling train-like state machines, where the modal logic involves transitions from one state to its neighbor
- Concise, compact format for a state machine
- Reduced maintenance of graphical objects

When you add or remove states from a chart, you have to rearrange states, transitions, and junctions. When you add or remove states from a state transition table, you do not have to rearrange any graphical objects.

The new block is available in `sf1lib`. You can add the block to a new model by entering `sfnew(' -STT ')` at the MATLAB® command line.

For more information, see “Tabular Expression of Modal Logic” in the Stateflow documentation.

For more information, see “Tabular Expression of Modal Logic” and “Model Bang-Bang Controller with a State Transition Table”.

MATLAB language for state and transition labels with chart syntax auto-correction

In R2012b, you can use MATLAB as the action language to program Stateflow charts. Benefits of using MATLAB as the action language include:

- MATLAB syntax support in state labels and transition labels
You can use the same MATLAB code that you write in a script or enter at the command line.
- Automatic identification of unresolved symbols in the new Symbol Wizard
When you update the diagram or start simulation, the Symbol Wizard provides a list of unresolved data in your chart and infers the scope.
- Automatic inference of size, type, and complexity for data in the chart, based on usage (unless explicitly defined in the Model Explorer)
- Support for control flow logic in state labels

For example, you can write `if-else` statements directly inside state actions:

```
StateA
du:
if (x > 0)
    x = x + 1;
else
    x = x + 2;
end
```

You do not need to create a separate graphical function to define the flow logic.

- Automatic correction of common syntax errors.

For example, if you type `x++` on a transition segment, the expression is automatically converted to the correct MATLAB syntax, `{x=x+1}`.

For more information, see “MATLAB Syntax for States and Transitions” and “Model Event-Driven System Using MATLAB Expressions”.

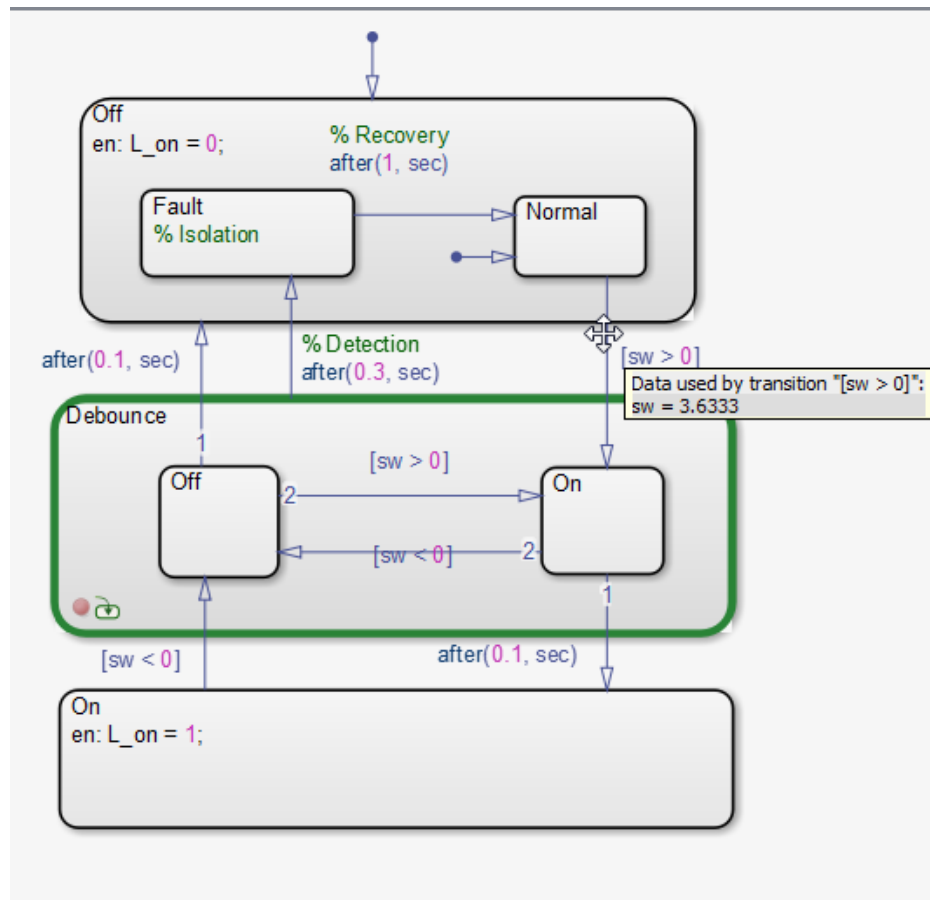
In-chart debugging with visual breakpoints and datatips

Compatibility Considerations: Yes

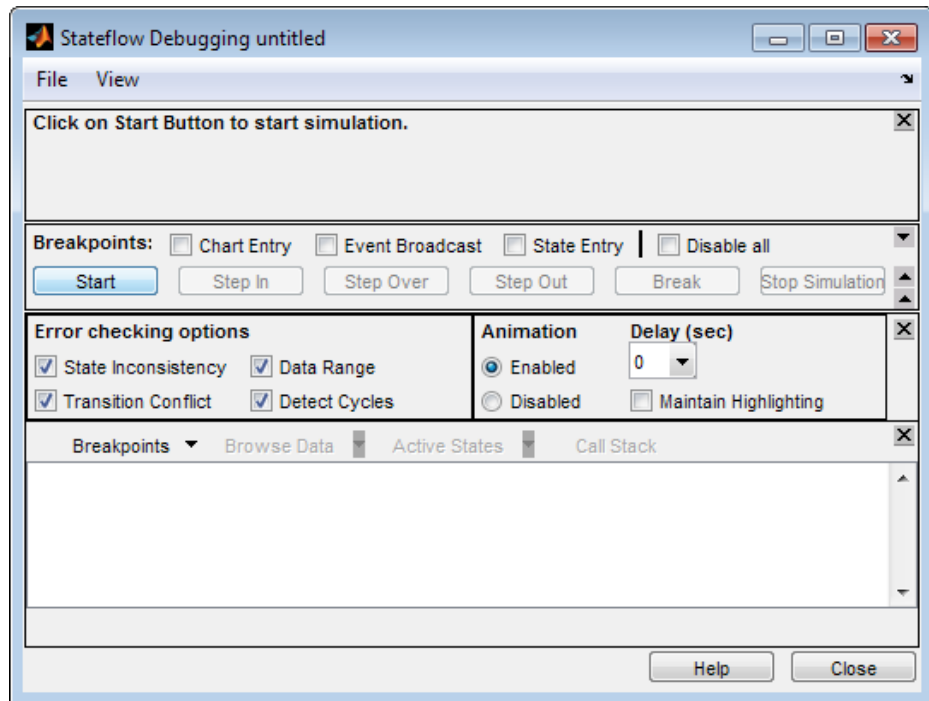
In R2012b, the Stateflow debugger includes the following enhancements:

- Display of data values when hovering over a state or transition

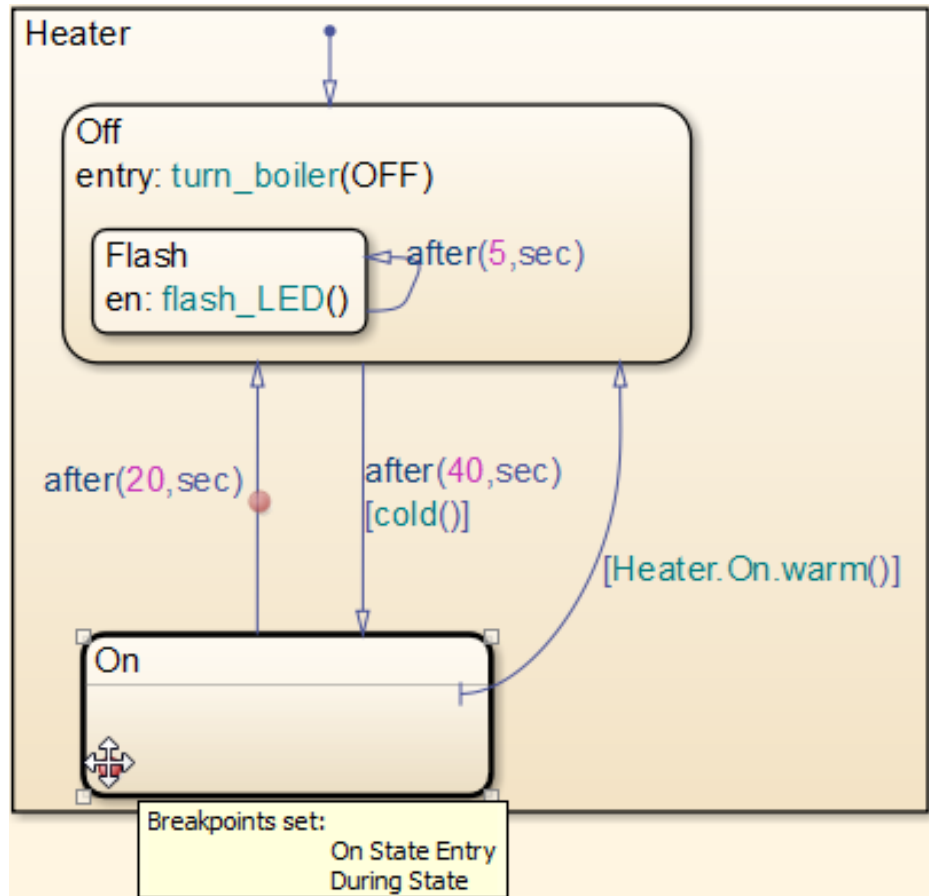
When you hover over a state or transition, all data values in scope for that object appear in a popup list.



- **Step Over** and **Step Out** options on the debugger



- When you click **Step Over**, you can skip the entire execution of a function call when the chart is in debug mode.
- When you click **Step Out**, you can skip the rest of the execution for a function call when the chart is in debug mode.
- Badges on graphical chart objects to indicate breakpoint settings



- When you hover over the badge on an object, you see a list of breakpoints in a popup list.
- To modify breakpoint settings, you can click the badge on an object instead of opening the properties dialog box.

See “Set Breakpoints to Debug Charts”.

Compatibility Considerations

In R2012b, you no longer need to launch the Stateflow debugger to stop chart execution at active breakpoints. Lifting this restriction means that during simulation, chart execution always stops at active breakpoints during simulation, even if the debugger is not running. To prevent unintended interruption of chart execution, Stateflow software automatically disables — but does not delete — existing breakpoints for all objects in charts created in earlier releases. Disabled breakpoints appear as gray badges; you can enable them as needed. See “Relationship Between Breakpoints and the Debugger” and “Set Local Breakpoints”.

In addition, in models created in earlier versions, Stateflow software removes `When Transition is Tested` breakpoints from transitions that do not have conditions. Starting in R2012b, you can set only `When Transition is Valid` breakpoints on transitions with no conditions.

Reuse of graphical functions with atomic boxes

In R2012b, you can use atomic boxes to reuse graphical functions across multiple charts and models. With atomic boxes, you can reuse models with graphical functions multiple times as referenced blocks in a top model. Because there are no *exported* graphical functions, you can use more than one instance of that referenced block in the top model.

For more information, see “Reusing Functions with an Atomic Box” in the Stateflow documentation.

Fewer restrictions for converting states to atomic subcharts

In R2012b, you can convert a state to an atomic subchart when the state accesses chart local data that has any of the following properties:

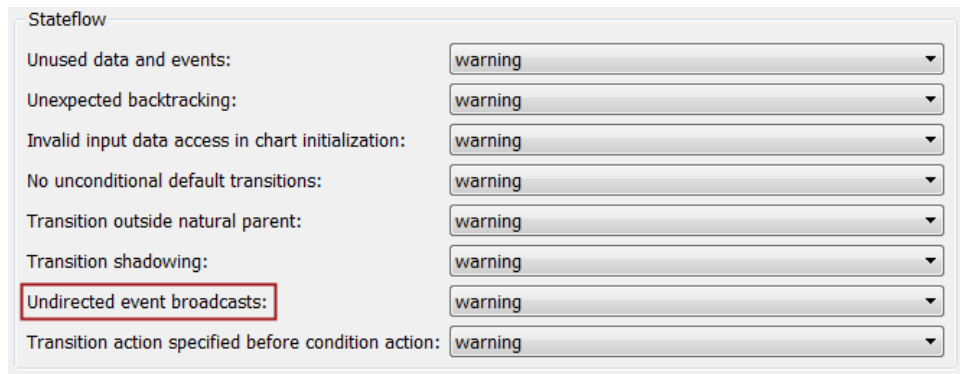
- [M N] size, where M and N are parameters that represent the data dimensions
- One of the following non-built-in data types:
 - Bus type
 - Alias type
 - Fixed-point type of nonzero fraction length, such as `fixdt(1,16,3)`

In previous releases, conversion of a state to an atomic subchart required that the chart local data have a static, deterministic size and a built-in data type.

Diagnostic for undirected local event broadcasts

Compatibility Considerations: Yes

In R2012b, you can detect undirected local event broadcasts using a new diagnostic in the **Diagnostics > Stateflow** pane of the Model Configuration Parameters dialog box. You can set the diagnostic level of **Undirected event broadcasts** to none, warning, or error.



Stateflow	
Unused data and events:	warning
Unexpected backtracking:	warning
Invalid input data access in chart initialization:	warning
No unconditional default transitions:	warning
Transition outside natural parent:	warning
Transition shadowing:	warning
Undirected event broadcasts:	warning
Transition action specified before condition action:	warning

Undirected local event broadcasts can cause unwanted recursive behavior in a chart and inefficient code generation. You can avoid this behavior by using the `send` operator to create *directed* local event broadcasts. For more information, see “Guidelines for Avoiding Unwanted Recursion in a Chart” and “Broadcasting Events to Synchronize States” in the Stateflow documentation.

Compatibility Considerations

For new models created in R2012b and existing models created in previous releases, the default diagnostic setting is `warning` to discourage the use of undirected local event broadcasts. Models that did not warn in previous releases might now issue a warning because the chart contains an undirected local event broadcast.

Diagnostic for transition action specified before condition action

Compatibility Considerations: Yes

In R2012b, you can detect specified transition actions before specified condition actions in transition paths, using a new diagnostic in the **Diagnostics > Stateflow** pane of the Model Configuration Parameters dialog box. You can set the diagnostic level of **Transition action specified before condition action** to none, warning, or error.

Stateflow	
Unused data and events:	warning
Unexpected backtracking:	warning
Invalid input data access in chart initialization:	warning
No unconditional default transitions:	warning
Transition outside natural parent:	warning
Transition shadowing:	warning
Undirected event broadcasts:	warning
Transition action specified before condition action:	warning

In a transition path with multiple transition segments, a specified transition action for a transition segment does not execute until the final destination for the entire transition path becomes valid. A specified condition action for a transition segment executes as soon as the condition becomes true. When a transition with a specified transition action precedes a transition with a specified condition action in the same transition path, the condition action for the succeeding transition might execute before the transition action for the preceding transition. When this diagnostic warns for transition paths containing transition actions specified before condition actions, you can identify out-of-order execution.

For more information, see “Transition Action Types” and “Transitions” in the Stateflow documentation.

Compatibility Considerations

In previous releases, the specification of transition actions before condition actions causes an error during simulation. To suppress this error for all models in future MATLAB sessions, use the following command:

```
sfpref('ignoreUnsafeTransitionActions',1);
```

In R2012b, the `ignoreUnsafeTransitionActions` preference does not exist and the default value of the **Transition action specified before condition action** diagnostic is warning. The warning occurs for all instances of transition actions specified before condition actions, even if you changed the `ignoreUnsafeTransitionActions` preference in a previous release.

Parentheses to identify function-call output events on chart and truth table block icons

In R2012b, function-call output events appear on Chart and Truth Table block icons with parentheses after the event name. This appearance is consistent with the rendering of input triggers on Function-Call Subsystem block icons.

Resolution of qualified state and data names

Compatibility Considerations: Yes

The algorithm for resolving a qualified state or data name performs a localized search for states and data that match the given path by looking in each level of the Stateflow hierarchy between the chart level and the parent of the state or data. The algorithm does not perform an exhaustive search of all states and data in the entire chart.

In previous releases, a warning would appear when the search resulted in no matches or multiple matches. In R2012b, this warning has changed to an error. For more information, see “Checking State Activity” and “Using Dot Notation to Identify Data in a Chart” in the Stateflow documentation.

Compatibility Considerations

Stateflow charts created in earlier releases now generate an error instead of a warning when the search for a qualified state or data name results in no matches or multiple matches.

Support for simulating charts in a folder that has the # symbol on 32-bit Windows platforms

In R2012b, on 32-bit Windows platforms, you can use the lcc compiler to simulate charts in a folder with the # symbol in its name.

Mac screen menubar enabled when Stateflow is installed

In previous releases, the Mac screen menubar was disabled when Stateflow was installed. This behavior was necessary to enable the Stateflow Editor menu options to work normally on a Mac.

In R2012b, the Mac screen menubar is enabled when Stateflow is installed.

Option to print charts to figure windows no longer available

Compatibility Considerations: Yes

In R2012b, printing the current view of a chart to a figure window is no longer available.

Compatibility Considerations

To print the current view of a chart, you can send the output directly to a printer or to a file. Available file formats include PS, EPS, JPG, PNG, and TIFF.

End of Broadcast breakpoint no longer available for input events

Compatibility Considerations: Yes

In R2012b, the option to set a breakpoint at **End of Broadcast** is no longer available for input events.

Compatibility Considerations

In previous releases, you could set both **Start of Broadcast** and **End of Broadcast** breakpoints for input events. Starting in R2012b, Stateflow ignores **End of Broadcast** breakpoints on input events for existing models.

R2012a

Version: 7.9
New Features: Yes
Bug Fixes: Yes

API Method for Highlighting Chart Objects

You can use the new `highlight` method to highlight one of the following objects in a chart:

- Box
- State
- Atomic subchart
- Transition
- Junction
- Graphical function
- MATLAB function
- Simulink function
- Truth table function

For more information, see `highlight` in the Stateflow API documentation.

API Method for Finding Transitions That Terminate on States, Boxes, or Junctions

You can use the new `sinkedTransitions` method to find all inner and outer transitions whose destination is a state, box, or junction.

For more information, see `sinkedTransitions` in the Stateflow API documentation.

API Property That Specifies the Destination Endpoint of a Transition

Transition objects now have a `DestinationEndPoint` property that describes the location of the transition endpoint at the destination object.

For more information, see [Transition Properties](#) in the Stateflow API documentation.

Structures and Enumerated Data Types Supported for Inputs and Outputs of Exported Graphical Functions

In R2012a, inputs and outputs of exported graphical functions can use enumerated data types or structures. For more information, see Rules for Exporting Chart-Level Graphical Functions in the Stateflow documentation.

Mappings Tab in Atomic Subchart Properties Dialog Lists All Valid Scopes

In R2012a, the **Mappings** tab in the atomic subchart properties dialog lists all valid scopes for data and event mapping. All valid scopes appear, regardless of whether a data or event of that scope exists in the chart.

In previous releases, the **Mappings** tab listed only the scopes of data and events that existed in the chart and omitted any scope that did not exist. For more information, see Mapping Variables for Atomic Subcharts in the Stateflow documentation.

Full Decision Coverage When Suppressing Default Cases in the Generated Code

In R2011b, selecting **Suppress generation of default cases for Stateflow switch statements if unreachable** in the Configuration Parameters dialog box would result in decision coverage of less than 100%. In R2012a, you get full decision coverage when suppressing default cases in the generated code for your Stateflow chart.

For more information about decision coverage, see [Model Coverage for Stateflow Charts](#) in the Simulink Verification and Validation documentation.

Specification of Custom Header Files in the Configuration Parameters Dialog Box Required for Enumerated Types

Compatibility Considerations: Yes

If data in your chart uses an enumerated type with a custom header file, include the header information in the **Simulation Target > Custom Code** pane of the Configuration Parameters dialog box. In the **Header file** section, add the following statement:

```
#include "<custom_header_file_for_enum>.h"
```

For more information, see Rules for Using Enumerated Data in a Stateflow Chart in the Stateflow User's Guide.

Compatibility Considerations

In earlier releases, custom header files for enumerated types did not need to appear in the Configuration Parameters dialog box.

Removal of 'Use Strong Data Typing with Simulink I/O' in a Future Release

Compatibility Considerations: Yes

In a future release, the **Use Strong Data Typing with Simulink I/O** check box will be removed from the Chart properties dialog box because strong data typing will always be enabled.

When this check box is cleared, the chart accepts and outputs only signals of type `double`. This setting ensures that charts created prior to R11 can interface with Simulink input and output signals without type mismatch errors. For charts created in R11 and newer releases, disabling strong data typing is unnecessary. Also, many Stateflow features do not work when a chart disables strong data typing.

Compatibility Considerations

In R2012a, updating the diagram causes a parse warning to appear when a chart disables strong data typing. To prevent the parse warning, select the **Use Strong Data Typing with Simulink I/O** check box in the Chart properties dialog box.

R2011b

Version: 7.8
New Features: Yes
Bug Fixes: Yes

Chart Property to Control Saturation for Integer Overflow

A new chart property, **Saturate on integer overflow**, enables you to control the behavior of data with signed integer types when overflow occurs. The check box appears in the Chart properties dialog box.

Check Box	When to Use This Setting	Overflow Handling	Example of a Result
Selected	Overflow is possible for data in your chart and you want explicit saturation protection in the generated code.	Overflows saturate to either the minimum or maximum value that the data type can represent.	An overflow associated with a signed 8-bit integer saturates to -128 or $+127$.
Cleared	You want to optimize efficiency of the generated code.	The behavior depends on the C compiler you use for generating code.	The number 130 does not fit in a signed 8-bit integer and wraps to -126 .

Arithmetic operations for which you can enable saturation protection are:

- Unary minus: $-a$
- Binary operations: $a + b$, $a - b$, $a * b$, a / b , $a ^ b$
- Assignment operations: $a += b$, $a -= b$, $a *= b$, $a /= b$

For new charts, this check box is selected by default. When you open charts saved in previous releases, the check box is cleared to maintain backward compatibility.

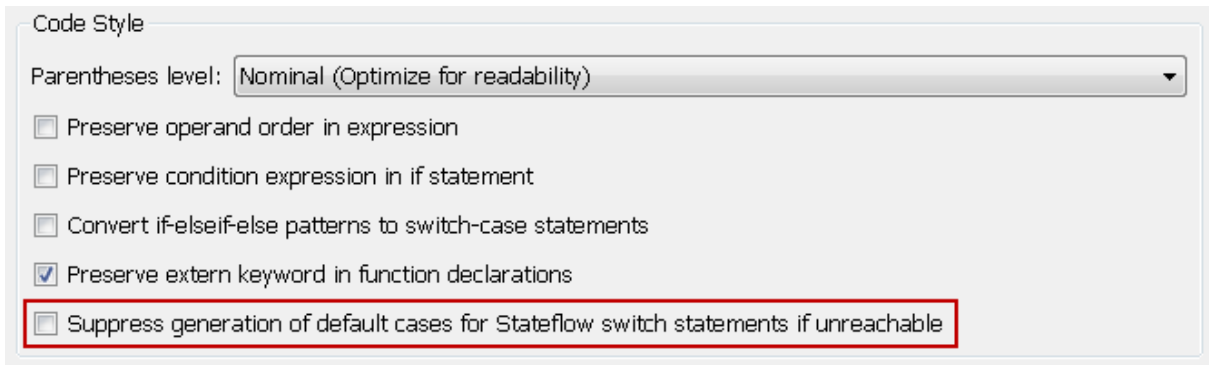
For more information, see *Handling Integer Overflow for Chart Data* in the *Stateflow User's Guide*.

Enhanced User Interface for Logging Data and States

A new **Logging** tab on the Data and State properties dialog boxes enables you to log signals in the same way that you do in Simulink. For more information, see Logging Data Values and State Activity in the Stateflow User's Guide.

Control of Default Case Generation for Switch-Case Statements in Generated Code

You can specify whether or not to always generate default cases for switch-case statements. This optimization works on a per-model basis and applies to the code generated for a state that has multiple substates. Use the following check box on the **Code Generation > Code Style** pane of the Configuration Parameters dialog box:



Check Box	When to Use This Setting	Format of Switch-Case Statements
Selected	Provide better code coverage by ensuring that every branch in the generated code is falsifiable.	Exclude the default case when it is unreachable.
Cleared	Ensure MISRA C [®] compliance and provide a fallback in case of RAM corruption.	Always include a default case.

This readability optimization is available for embedded real-time (ERT) targets and requires a license for Embedded Coder[®] software. For new models, this check box is cleared by default. When you open models saved in previous releases, the check box is also cleared to maintain backward compatibility.

For more information, see [Code Generation Pane: Code Style in the Embedded Coder documentation](#).

Detection of State Inconsistency Errors at Compile Time Instead of Run Time

Compatibility Considerations: Yes

In R2011b, you can detect inconsistency errors earlier in the model development process. If you select **Edit > Update Diagram** in the Simulink Editor, you get an error when Stateflow statically detects that there are no active children during execution of a chart or a state.

Compatibility Considerations

In previous releases, static detection of these inconsistency errors did not occur until run time.

Ability to Model Persistent Output Data for Mealy and Moore Charts

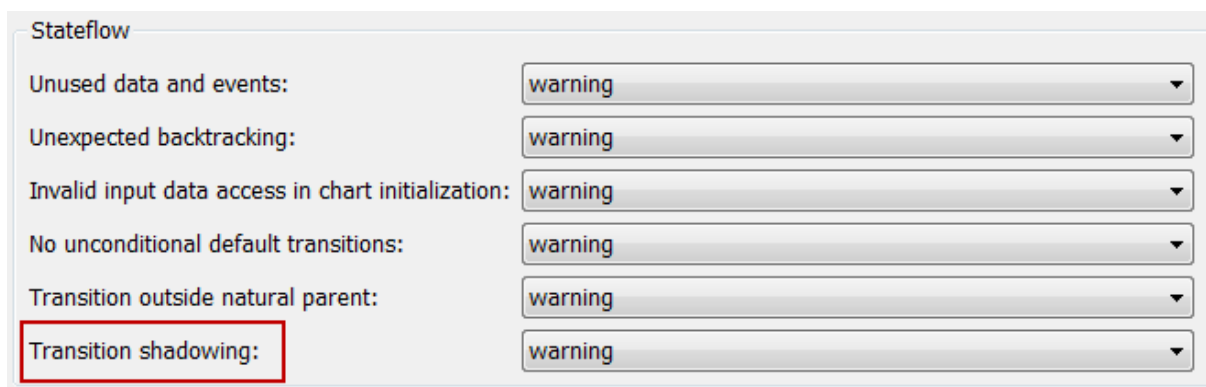
In previous releases, Mealy and Moore charts automatically applied the initial value of outputs every time the chart woke up. Both chart types ensured that outputs did not depend on previous values of outputs by enforcing the chart property **Initialize Outputs Every Time Chart Wakes Up**.

In R2011b, this restriction has been lifted. You can now choose whether or not to initialize outputs every time a Mealy or Moore chart wakes up. If you disable this chart property, you enable latching of outputs (carrying over output values computed in previous time steps). This enhancement enables you to model persistent output data for Mealy and Moore charts.

For more information, see [Building Mealy and Moore Charts in the Stateflow User's Guide](#).

Control of Diagnostic for Multiple Unconditional Transitions from One Source

You can control the behavior of the Stateflow diagnostic that detects multiple unconditional transitions from the same state or the same junction. Set **Transition shadowing** to none, warning, or error on the **Diagnostics > Stateflow** pane of the Configuration Parameters dialog box.



The image shows a screenshot of the 'Stateflow' configuration pane from the Configuration Parameters dialog box. It contains six diagnostic settings, each with a dropdown menu. The 'Transition shadowing' setting is highlighted with a red rectangular box. All settings are currently set to 'warning'.

Diagnostic	Setting
Unused data and events:	warning
Unexpected backtracking:	warning
Invalid input data access in chart initialization:	warning
No unconditional default transitions:	warning
Transition outside natural parent:	warning
Transition shadowing:	warning

For more information, see [Diagnostics Pane: Stateflow](#) in the Simulink Graphical User Interface documentation.

MEX Compilation with Microsoft Windows SDK 7.1 Now Supported

You can use Microsoft® Windows Software Development Kit (SDK) 7.1 as a MEX compiler for simulation on 32- and 64-bit Windows machines. For a list of supported compilers, see [Choosing a Compiler](#) in the Stateflow User's Guide.

Simulation Supported When the Current Folder Is a UNC Path

In R2011b, you can simulate models with Stateflow blocks when the current folder is a UNC path. In previous releases, simulation of those models required that the current folder not be a UNC path.

Removal of the Coverage Tab from the Stateflow Debugger

In R2011b, the **Coverage** tab of the Stateflow debugger has been removed. In previous releases, clicking the **Coverage** tab would show the following message:

```
Coverage feature obsoleted. Please use Simulink Verification
and Validation in order to get complete coverage of
Simulink/Stateflow objects.
```

Test Point Selection Moved to the Logging Tab in Properties Dialog Boxes

The **Test point** check box has moved from the **General** tab to the **Logging** tab on State and Data properties dialog boxes.

R2011a

Version: 7.7
New Features: Yes
Bug Fixes: Yes

Migration of Stateflow Coder Features to New Product

In R2011a, all functionality previously available for the Stateflow Coder™ product is now part of the new Simulink Coder product.

Embedded MATLAB Functions Renamed as MATLAB Functions in Stateflow Charts

In R2011a, Embedded MATLAB functions have been renamed as MATLAB functions in Stateflow charts. This name change has the following effects:

- The function box now shows MATLAB Function instead of eM in the upper-left corner.
- Traceability comments in the generated code for embedded real-time targets now use MATLAB Function instead of Embedded MATLAB Function.
- For truth table functions in your chart, the **Settings > Language** menu now provides Stateflow Classic and MATLAB as the choices.

Scripts that use the `Stateflow.EMFunction` constructor method continue to work. All properties and methods for this object remain the same.

Use of MATLAB Expressions to Specify Data Size

Compatibility Considerations: Yes

In R2011a, you can enter MATLAB expressions in the **Size** field of the Data properties dialog box:

Data airflow

General | Description

Name: airflow

Scope: Output Port: 1

Data must resolve to Simulink signal object

Size: N+2

Complexity: Off

Type: uint8 >>

Lock data type setting against changes by the fixed-point tools

Initial value: Expression

Limit range

Minimum: Maximum:

Test point Watch in debugger

This enhancement enables you to use additional constructs, such as:

- Variables in the MATLAB base workspace
- Enumerated values on the MATLAB search path
- Expressions that use `fi` objects

For more information, see [Sizing Stateflow Data](#) in the Stateflow User's Guide.

Compatibility Considerations

For the **Size** field, name conflict resolution works differently from previous releases. In R2011a, when multiple variables with identical names exist, the variable with the highest priority is used:

- 1** Mask parameters
- 2** Model workspace
- 3** MATLAB base workspace
- 4** Stateflow data

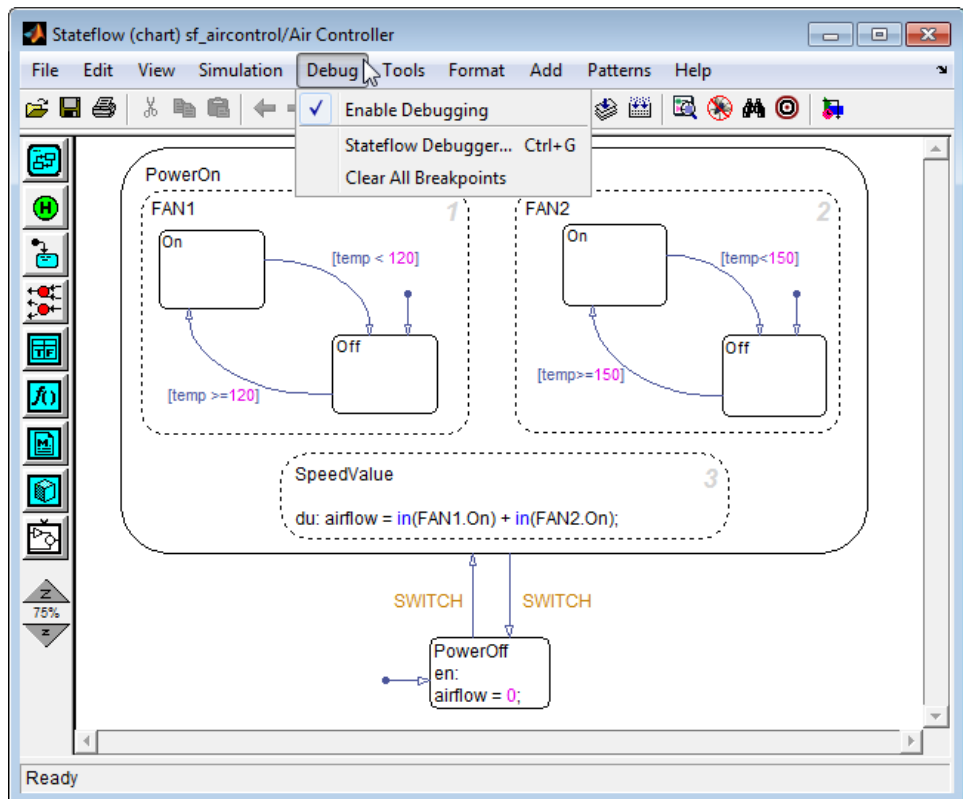
In previous releases, Stateflow data took precedence over all other variables with identical names.

Ability to Change Data Values While Debugging

Previously, you could not change the values of Stateflow data while debugging a chart. Now you can change data values while the chart is in debug mode and see how simulation results change. For more information, see [Changing Data Values During Simulation](#) in the Stateflow User's Guide.

Ability to Debug a Single Chart When Multiple Charts Exist in a Model

When **Enable debugging/animation** is enabled on the **Simulation Target** pane of the Configuration Parameters dialog box, this setting applies to all charts in your model. In R2011a, you can enable or disable debugging on a chart-by-chart basis, using the **Debug** menu in the Stateflow Editor:



This enhancement enables you to focus on debugging a single chart, instead of having to debug all charts in the model. For details, see [How to Enable Debugging for Charts in the Stateflow User's Guide](#).

You can also clear all breakpoints for a specific chart by selecting **Debug > Clear All Breakpoints** in the Stateflow Editor. For more information, see [Clearing All Breakpoints in the Stateflow User's Guide](#).

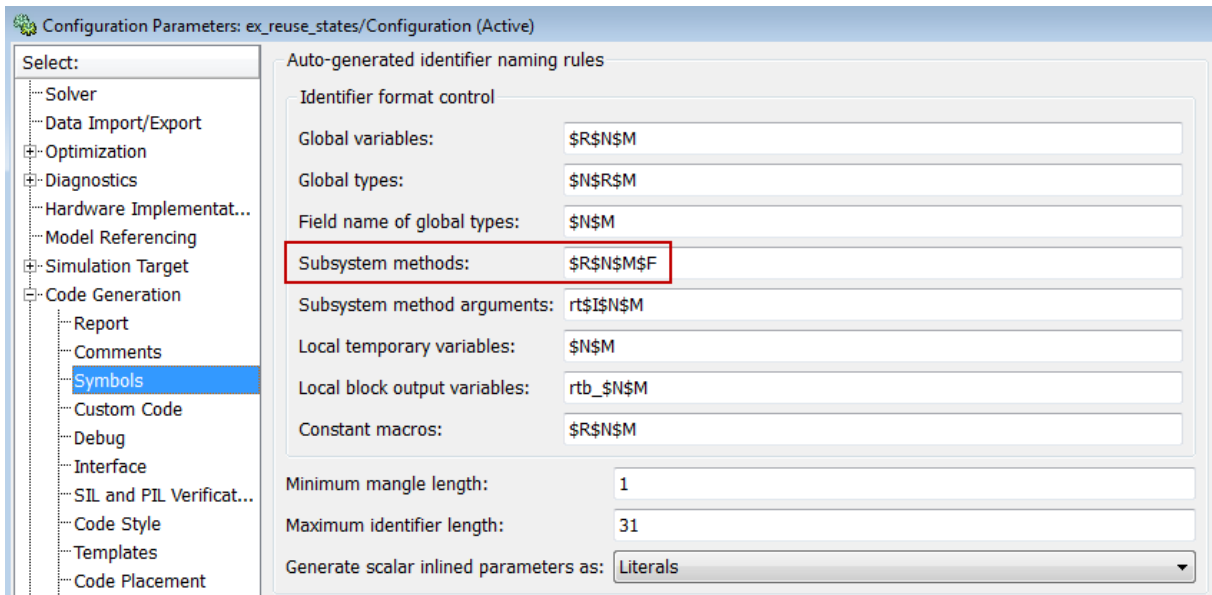
In previous releases, you could open the debugger by selecting **Tools > Debug** in the Stateflow Editor. In R2011a, this menu option has moved to **Debug > Stateflow Debugger**.

Support for Input Events in Atomic Subcharts

In R2011a, you can use input events in atomic subcharts. For more information, see [Making States Reusable with Atomic Subcharts](#) in the Stateflow User's Guide.

Control of Generated Function Names for Atomic Subcharts

In R2011a, the generated function names for atomic subcharts follow the identifier naming rules for subsystem methods on the **Code Generation > Symbols** pane of the Configuration Parameters dialog box:



This enhancement enables you to control the format of generated function names for atomic subcharts when building an embedded real-time (ERT) target. For more information, see *Generating Reusable Code for Unit Testing* in the Stateflow User's Guide.

Enhanced Data Sorting in the Stateflow Debugger

In previous releases, the Stateflow debugger sorted data by scope first, before alphabetically listing data. In R2011a, the debugger sorts data alphabetically in the **Browse Data** section, without regard to scope. This enhancement helps you find specific data quickly when your chart contains many variables, for example, over a hundred.

Data sorting depends solely on the variable name and not on hierarchy. For example, if you have chart-parented data named `arrayOut` and state-parented data named `arrayData`, the list that appears in the **Browse Data** section is:

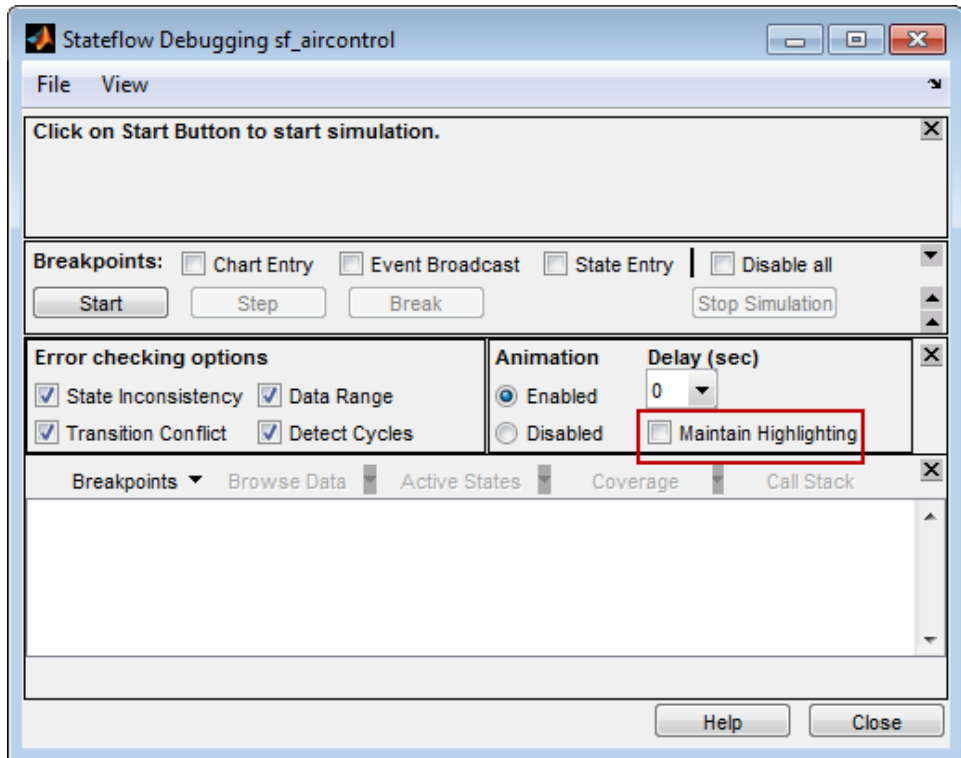
```
S.arrayData  
arrayOut
```

The state name has no effect on data sorting.

For more information, see [Watching Data Values During Simulation](#) in the Stateflow User's Guide.

Option to Maintain Highlighting of Active States After Simulation

In R2011a, you can highlight the states that are active at the end of a simulation by selecting **Maintain Highlighting** in the Stateflow debugger.



This enhancement enables you to inspect the active states of a chart after simulation ends, without having to use the SimState method `highlightActiveStates`. For more information, see *Animating Stateflow Charts* in the Stateflow User's Guide.

Right-Click Options for Setting Local Breakpoints

For graphical chart objects, you can now right-click the object to set local breakpoints. This enhancement enables you to set breakpoints more quickly, without having to open the properties dialog box for:

- Charts
- States
- Transitions
- Graphical functions
- Truth table functions

For more information, see [Setting Local Breakpoints](#) in the Stateflow User's Guide.

New Signal Logging Format That Simplifies Access to States and Local Data

Compatibility Considerations: Yes

You can now select a format for signal logging data. Use the **Signal logging format** parameter on the **Data Import/Export** pane of the Configuration Parameters dialog box to specify the format:

- `ModelDataLogs` — `Simulink.ModelDataLogs` format (the default; before R2011a, it was the only supported format)
- `Dataset` — `Simulink.Simulation.Dataset` format (new in R2011a)

The `Dataset` format:

- Supports logging multiple data values for a given time step, which enhances signal logging of Stateflow data
- Uses MATLAB `timeseries` objects to store logged data (rather than `Simulink.Timeseries` and `Simulink.TsArray` objects), which enables you to work with logged data in MATLAB without a Simulink license
- Avoids the limitations of the `ModelDataLogs` format, which Bug Report 495436 describes

For more information, see [Logging Data Values and State Activity](#).

Compatibility Considerations

In previous releases, selecting **Enable debugging/animation** on the **Simulation Target** pane of the Configuration Parameters dialog box would implicitly set all data and states in a Stateflow chart to be test points. In R2011a, you must select the **Test point** check box explicitly for data and states to appear in the Signal Selector dialog box of a Scope or Floating Scope block.

If you load models from previous releases that rely on the implicit behavior, mark the appropriate data or states as test points to ensure that they appear in the Signal Selector dialog box. For more information, see [Monitoring Test Points in Stateflow Charts](#) in the Stateflow User's Guide.

Support for Buses in Data Store Memory

You can now use buses, but not arrays of buses, as shared data in Stateflow data store memory.

Enhanced Readability of State Functions

In R2011a, state functions are more readable due to improved inlining heuristics.

Support for Arrays of Buses as Inputs and Outputs of Charts and Functions

In R2011a, you can pass arrays of buses as inputs and outputs of the following Stateflow objects:

- Charts
- MATLAB functions
- Simulink functions

Default Setting of 'States When Enabling' Chart Property Now Held

For new charts, the default setting of the **States When Enabling** chart property is **Held**. In previous releases, the default setting was **Inherit**. For more information, see [Controlling States When Function-Call Inputs Reenable Charts](#) in the Stateflow User's Guide.

Initial Value Vectors with Fixed-Point or Enumerated Values Now Evaluate Correctly

Compatibility Considerations: Yes

In previous releases, if you set an initial value vector using fixed-point or enumerated values, all elements of that vector would have the same value as the first element. For example:

For this initial value vector...	The chart used these values...
[fi(1,1,16,0) fi(2,1,16,0)]	[1 1], instead of [1 2]
[Colors.Red Colors.Yellow Colors.Green]	[Colors.Red Colors.Red Colors.Red], instead of [Colors.Red Colors.Yellow Colors.Green]

In R2011a, this bug has been fixed.

Compatibility Considerations

If you have any models that rely on the behavior of initial value vectors from previous releases, these models will behave differently in R2011a.

Mac Screen Menubar Disabled When Stateflow Is Installed

In R2011a, the Mac screen menubar is disabled when Stateflow is installed. This behavior enables Stateflow Editor menu options to work normally on a Mac.

To enable the Mac screen menubar, modify the `java.opts` file by adding the following line:

```
-Dapple.laf.useScreenMenuBar=true
```

To prevent a slowdown in the MATLAB Editor, check that the `java.opts` file contains the following line:

```
-Dapple.awt.graphics.UseQuartz=true
```

A `java.opts` file can reside in the folder from which you launch MATLAB or in the `bin/maci64` subfolder within the MATLAB root folder. A `java.opts` file in the latter location applies to all users, but individual users might not have permissions to modify a `java.opts` file there. If there is a `java.opts` file in both locations with settings that conflict, the setting in the `java.opts` file in the folder from which you launch MATLAB takes precedence. You might want to check both locations to see whether you have existing `java.opts` files and then decide which one to modify.

- To create a new `java.opts` file or modify an existing copy in the folder from which you launch MATLAB:

- 1** Quit MATLAB.

- 2** Relaunch MATLAB and immediately enter the following line in the Command Window:

```
edit java.opts
```

- To create or modify a `java.opts` file that applies to all users, you can enter the following line in the MATLAB Command Window at any time:

```
edit(fullfile(matlabroot,'bin','maci64','java.opts'))
```

R2010bSP2

Version: 7.6.2
New Features: No
Bug Fixes: Yes

R2010bSP1

Version: 7.6.1
New Features: No
Bug Fixes: Yes

R2010b

Version: 7.6
New Features: Yes
Bug Fixes: Yes

New Atomic Subcharts to Create Reusable States for Large-Scale Modeling

In R2010b, you can use atomic subcharts to:

- Break up a chart into standalone parts to facilitate team development
- Reuse states across multiple charts and models
- Animate and debug multiple charts side-by-side during simulation
- Use simulation to test changes, one-by-one, without recompiling the entire chart
- Generate reusable code for specific states or subcharts to enhance unit testing

For more information, see [Making States Reusable with Atomic Subcharts](#) in the Stateflow User's Guide.

Stateflow Library Charts Now Support Instances with Different Data Sizes, Types, and Complexities

In R2010b, you can use library link charts that specify different data sizes, types, and complexities. Previously, all library charts had to use the same settings for data size, type, and complexity. For more information, see *Creating Specialized Chart Libraries for Large-Scale Modeling* in the Stateflow User's Guide.

Support for Controlling Stateflow Diagnostics in the Configuration Parameters Dialog Box

In R2010b, you can control the behavior of the following Stateflow diagnostics in the **Diagnostics > Stateflow** pane of the Configuration Parameters dialog box:

- Unused data and events
- Unexpected backtracking
- Invalid input data access in chart initialization
- No unconditional default transitions
- Transition outside natural parent

For more information, see [Diagnostics Pane: Stateflow in the Simulink Graphical User Interface](#).

Enhanced Custom-Code Parsing to Improve Reporting of Unresolved Symbols

In R2010b, you can resolve symbols in your chart to symbols defined in custom code while parsing the chart. This enhancement enables more accurate and earlier reporting of unresolved symbols. Previously, the parser assumed that any unresolved chart symbols were defined in custom code. You could not resolve chart symbols to symbols in your custom code until make time. If the chart symbols were undefined in the custom code, a make error would appear.

Also, the Symbol Autocreation Wizard was previously available only for 32-bit Windows platforms that use lcc for the mex compiler. In R2010b, the Symbol Autocreation Wizard is available to help you fix unresolved symbols, regardless of the compiler or platform.

To enable or disable custom-code parsing, you can use the **Parse custom code symbols** check box on the **Simulation Target > Custom Code** pane of the Configuration Parameters dialog box.

For more information, see:

- Parse custom code symbols in the Simulink Graphical User Interface
- Resolving Undefined Symbols in Your Chart in the Stateflow User's Guide

Temporal Logic Conditions Can Now Guard Transitions Originating from Junctions

Previously, you could not use temporal logic conditions on transitions that originated from junctions. Now you can use temporal logic conditions on transitions from junctions as long as the full transition path connects two states. For more information, see [Rules for Using Temporal Logic Operators](#) and [Example of Detecting Elapsed Time](#) in the Stateflow User's Guide.

Data Dialog Box Enhancements

In R2010b, the following changes to the Data properties dialog box apply:

Parameters	Location in R2010a	Location in R2010b	Benefit of Location Change
<ul style="list-style-type: none"> • Initial value • Minimum • Maximum 	Value Attributes tab	General tab	Consistent with blocks in the Simulink library that specify these parameters on the same tab as the data type.
<ul style="list-style-type: none"> • Test point • Watch in debugger 	Value Attributes tab	General tab	Increases visibility of commonly used parameters.
<ul style="list-style-type: none"> • Save final value to base workspace • First index • Units 	Value Attributes tab	Description tab	Consolidates parameters related to the data description.

Branching of Function-Call Output Events No Longer Requires Binding of Event to a State

Previously, using a Function-Call Split block to branch a function-call output event from a chart to separate subsystems required binding of the event to a state. In R2010b, binding is no longer required.

Passing Real Values to Function Inputs of Complex Type Disallowed

Compatibility Considerations: Yes

In R2010b, you cannot pass real values to function inputs of complex type. This restriction applies to the following types of chart functions:

- Graphical functions
- Truth table functions
- Embedded MATLAB® functions
- Simulink functions

Compatibility Considerations

If you have existing models that pass real values to function inputs of complex type, an error now appears when you try to simulate your model.

Using Chart Block That Accesses Global Data in For Each Subsystem Disallowed

In R2010b, the following model configuration produces an error during Real-Time Workshop® code generation:

- A Chart block resides in a For Each Subsystem.
- The Chart block tries to access global data from Data Store Memory blocks.

New and Enhanced Demos

The following demos have been added in R2010b:

Demo...	Shows how you can...
sf_atomic_sensor_pair	Model a redundant sensor pair using atomic subcharts
sf_electrohydraulic	Model a servo mechanism for use in electrohydraulic systems

The following demo has been enhanced in R2010b:

Demo...	Now...
sf_elevator	Models an elevator system with two identical lifts using atomic subcharts

R2010a

Version: 7.5
New Features: Yes
Bug Fixes: Yes

Support for Combining Actions in State Labels

You can now combine `entry`, `during`, and `exit` actions in a single line on state labels. This concise syntax provides enhanced readability for your chart and helps eliminate redundant code. For more information, see [Combining State Actions to Eliminate Redundant Code](#) in the Stateflow User's Guide.

New Diagnostic Detects Unused Data and Events

A new diagnostic now detects unused Stateflow data and events during simulation. A warning message appears, alerting you to data and events that you can remove. This enhancement helps you reduce the size of your model by removing objects that have no effect on simulation.

This diagnostic checks for usage of Stateflow data, except for the following types:

- Machine-parented data
- Inputs and outputs of Embedded MATLAB functions

This diagnostic checks for usage of Stateflow events, except for the following type:

- Input events

For more information, see [Diagnostic for Detecting Unused Data](#) and [Diagnostic for Detecting Unused Events](#) in the Stateflow User's Guide.

Enhanced Support for Variable-Size Chart Inputs and Outputs

You can explicitly pass variable-size chart inputs and outputs as inputs and outputs of the following functions:

- Embedded MATLAB functions
- Simulink functions
- Truth table functions that use Embedded MATLAB action language

For more information, see [Using Variable-Size Data in Stateflow Charts](#) in the Stateflow User's Guide.

Support for Chart-Level Data with Fixed-Point Word Lengths Up to 128 Bits

Chart-level data now support up to 128 bits of fixed-point precision for the following scopes:

- Input
- Output
- Parameter
- Data Store Memory

This increase in maximum precision from 32 to 128 bits provides these enhancements:

- Supports generating efficient code for targets with non-standard word sizes
- Allows charts to work with large fixed-point signals

You can explicitly pass chart-level data with these fixed-point word lengths as inputs and outputs of the following functions:

- Embedded MATLAB functions
- Simulink functions
- Truth table functions that use Embedded MATLAB action language

For more information, see [Using Fixed-Point Data in Stateflow Charts](#) in the Stateflow User's Guide.

New 'States When Enabling' Property for Charts with Function-Call Input Events

The new chart property **States When Enabling** helps you specify how states behave when a function-call input event reenables a chart. You can select one of the following settings in the Chart properties dialog box:

- **Held** — Maintain most recent values of the states.
- **Reset** — Revert to the initial conditions of the states.
- **Inherit** — Inherit this setting from the parent subsystem.

This enhancement helps you more accurately control the behavior of a chart with a function-call input event. For more information, see [Controlling States When Function-Call Inputs Reenable Charts](#) and [Setting Properties for a Single Chart](#) in the Stateflow User's Guide.

Support for Tunable Structures of Parameter Scope in Charts

You can now define structures of parameter scope that are tunable. For more information, see [Defining Structures of Parameter Scope in the Stateflow User's Guide](#).

Enhanced Real-Time Workshop Code Generation for Noninlined State Functions

If you prevent inlining for a state, Real-Time Workshop generated code contains a new static function `inner_default_statename` when:

- Your chart contains a flow graph where an inner transition and default transition reach the same junction inside a state.
- This flow graph is complex enough to exceed the inlining threshold.

For more information, see [What Happens When You Prevent Inlining in the Stateflow User's Guide](#).

Enhanced Real-Time Workshop Code Generation for sizeof Function

When you use the `sizeof` function in generated code to determine vector or matrix dimensions, `sizeof` always takes an input argument that evaluates to a data type.

Behavior in Prior Releases	Behavior in R2010a	Benefits of Change in Code
Input argument references the address of the variable, for example: <code>sizeof(&a[0])</code>	Input argument evaluates to the data type of the variable, for example: <code>sizeof(uint8_T [256])</code>	Ensures consistent results between simulation and code generation.

Enhanced Real-Time Workshop Code Generation for Custom-Code Function Calls

When you use custom-code function calls in generated code, vector and matrix input arguments always use pass-by-reference instead of pass-by-value behavior.

Behavior in Prior Releases	Behavior in R2010a	Benefits of Change in Code
<p>Custom-code function calls might use either pass-by-reference or pass-by-value.</p> <p>For pass-by-value, a memcpy operation creates and stores a temporary variable in the generated code, for example:</p> <pre>int t[10]; for (i=0; i<10; i++) { t[i] = y[i]; } fcn(t);</pre>	<p>Custom-code function calls use pass-by-reference, for example:</p> <pre>fcn(&y[0]);</pre>	<ul style="list-style-type: none"> • Ensures consistent results between simulation and code generation. • Less memory usage because a temporary variable is not necessary. • Faster execution of generated code because a memcpy operation is not necessary.

Data Change Implicit Event No Longer Supports Machine-Parented Data

Compatibility Considerations: Yes

The implicit event change (*data_name*) no longer works for machine-parented data. In R2010a, this implicit event works only with data at the chart level or lower in the hierarchy.

Compatibility Considerations

For machine-parented data, consider using change detection operators to determine when data values change. For more information, see [Detecting Changes in Data Values](#) in the Stateflow User's Guide.

Support for Machine-Parented Events Completely Removed

Compatibility Considerations: Yes

Support for machine-parented events has been completely removed. In R2010a, an error message appears when you try to simulate models that contain events at the machine level.

Compatibility Considerations

To prevent undesired behavior for simulation and code generation, do not use machine-parented events. For simulation, broadcasting an event to all charts in your model causes the following to occur:

- Charts wake up without regard to data dependencies.
- Charts that are disabled might wake up.
- Charts that use function-call or edge-triggered events wake up.
- Charts unrelated to the event wake up.
- Infinite recursive cycles can occur because the chart that broadcasts the event wakes up.

For code generation, machine-parented events prevent code reuse for the entire model.

MEX Compilation with Microsoft Visual Studio .NET 2003 No Longer Supported

Support for Microsoft Visual Studio® .NET 2003 as a MEX compiler for simulation has been removed because MATLAB and Simulink no longer support this compiler. For information about alternative compilers, see *Choosing a Compiler* in the Stateflow User's Guide.

Code Generation Status Messages No Longer Shown in Command Window

For Windows platforms, messages about Stateflow or Embedded MATLAB code generation and compilation status now appear only on the status bar of the Simulink Model Editor when you update diagram. Previously, these messages also appeared in the MATLAB Command Window. This enhancement minimizes distracting messages at the command prompt.

Change in Behavior for Appearance of Optimization Parameters

Previously, the Configuration Parameters dialog box showed the Stateflow section of the **Optimization** pane only when both of the following conditions were true:

- Real-Time Workshop and Stateflow licenses were available.
- Your model included Stateflow charts or Embedded MATLAB Function blocks.

In R2010a, the Configuration Parameters dialog box shows the Stateflow section of the **Optimization** pane when both licenses are available. Your model need not include any Stateflow charts or Embedded MATLAB Function blocks.

For a list of optimization parameters, see Optimization Pane: General in the Simulink Graphical User Interface.

Enhanced Inlining of Generated Code That Calls Subfunctions

In R2010a, Real-Time Workshop Embedded Coder software inlines generated code for Stateflow charts, even if the generated code calls a subfunction that accesses global Simulink data. This optimization uses less RAM and ROM.

Check Box for 'Treat as atomic unit' Now Always Selected

In existing models, simulation and code generation of Stateflow charts and Truth Table blocks always behave as if the **Treat as atomic unit** check box in the Subsystem Parameters dialog box is selected. Starting in R2010a, this check box is always selected for consistency with existing behavior.

New Demos

The following demos have been added in R2010a:

Demo...	Shows how you can...
<code>sf_combined_state_actions</code>	Combine entry and during actions in a single line on state labels
<code>sf_variable_size_data</code>	Pass variable-size data as an output of a Simulink function in a Stateflow chart
<code>sf_multiword_fixpt</code>	Pass multiword fixed-point data as an input and an output of a Simulink function in a Stateflow chart

R2009bSP1

Version: 7.4.1
New Features: No
Bug Fixes: Yes

R2009b

Version: 7.4
New Features: Yes
Bug Fixes: Yes

Ability to Copy Simulink Function-Call Subsystems and Paste in Stateflow Editor as Simulink Functions, and Vice Versa

You can copy a function-call subsystem from a model and paste directly in the Stateflow Editor. This enhancement eliminates the steps of manually creating a Simulink function in your chart and pasting the contents of the subsystem into the new function. You can also copy a Simulink function from a chart and paste directly in a model as a function-call subsystem.

For more information, see [Using Simulink Functions in Stateflow Charts](#) in the Stateflow User's Guide.

Ability to Generate Switch-Case Statements for Flow Graphs and Embedded MATLAB Functions Using Real-Time Workshop Embedded Coder Software

If a flow graph or Embedded MATLAB function in your chart uses `if-elseif-else` decision logic, you can choose to generate `switch-case` statements during Real-Time Workshop Embedded Coder code generation. `Switch-case` statements provide more readable and efficient code than `if-elseif-else` statements when multiple decision branches are possible.

When you load models created in R2009a and earlier, this optimization is off to maintain backward compatibility. In previous versions, `if-elseif-else` logic appeared unchanged in generated code.

For more information, see:

- [Enhancing Readability of Generated Code for Flow Graphs](#)
- [Enhancing Readability of Generated Code for MATLAB Functions](#)
- [Code Generation Pane: Code Style](#)

Support for Creating Switch-Case Flow Graphs Using the Pattern Wizard

In the Pattern Wizard, you can now choose to create a flow graph with switch-case decision logic. For more information, see *Modeling Logic Patterns and Iterative Loops Using Flow Graphs* in the Stateflow User's Guide.

Support for Using More Than 254 Events in a Chart

You can now use more than 254 events in a chart. The previous limit of 254 events no longer applies. This enhancement supports large-scale models with charts that send and receive hundreds of events during simulation. Although Stateflow software does not limit the number of events, the underlying C compiler enforces a theoretical limit of $(2^{31})-1$ events for the generated code.

For more information, see [Defining Events in the Stateflow User's Guide](#).

Improved Panning and Selection of States and Transitions When Using Stateflow Debugger

During single-step mode, the Stateflow Debugger no longer zooms automatically to the chart object that is executing. Instead, the debugger opens the subviewer that contains that object. This enhancement minimizes visual disruptions as you step through your analysis of a simulation.

For more information, see *Options to Control Execution Rate in the Debugger* in the Stateflow User's Guide.

Stateflow Compilation Status Added to Progress Indicator on Simulink Status Bar

For Windows platforms, messages about Stateflow compilation status now appear on the status bar of the Simulink Model Editor when you update diagram.

Support for Chart Inputs and Outputs That Vary in Dimension During Simulation

Charts now support input and output data that vary in dimension during simulation. In this release, only Embedded MATLAB functions nested in the charts can manipulate these input and output data.

For more information, see [Using Variable-Size Data in Stateflow Charts and Working with Variable-Size Data in MATLAB Functions](#) in the Stateflow User's Guide.

New Compilation Report for Embedded MATLAB Functions in Stateflow Charts

Compatibility Considerations: Yes

The new compilation report provides compile-time type information for the variables and expressions in your Embedded MATLAB functions. This information helps you find the sources of error messages and understand type propagation issues, particularly for fixed-point data types. For more information, see *Working with MATLAB Function Reports* in the Simulink User's Guide.

Compatibility Considerations

The new compilation report is not supported by the MATLAB internal browser on Sun™ Solaris™ 64-bit platforms. To view the compilation report on Sun Solaris 64-bit platforms, you must have your MATLAB Web preferences configured to use an external browser, for example, Mozilla® Firefox®. To learn how to configure your MATLAB Web preferences, see *Web Preferences* in the MATLAB documentation.

Enhanced Support for Replacing Math Functions with Target-Specific Implementations

You can now replace the following math functions with target-specific implementations:

Function	Data Type Support
atan2	Floating-point
fmod	Floating-point
ldexp	Floating-point
max	Floating-point and integer
min	Floating-point and integer

Replacement of `abs` now works for both floating-point and integer arguments. Previously, replacement of `abs` with a target function worked only for floating-point arguments.

For more information about Target Function Libraries, see:

- Replacement of C Math Library Functions with Target-Specific Implementations
- Replacing Operators with Target-Specific Implementations

Enhanced Context Menu Support for Adding Flow Graph Patterns to Charts

In a chart, you can now right-click at any level of the hierarchy (for example, states and subcharts) to insert flow graphs using the **Patterns** context menu. Previously, options in this context menu were available only if you right-clicked at the chart level.

Option to Log Chart Signals Available in the Stateflow Editor

To log chart signals, you can select **Tools > Log Chart Signals** in the Stateflow Editor. Previously, you had to right-click the Stateflow block in the Model Editor to open the Signal Logging dialog box.

For more information, see [What You Can Log During Chart Simulation](#) in the Stateflow User's Guide.

Default Precision Set to Double for Calls to C Math Functions

When you call C math functions, such as `sin`, `exp`, or `pow`, double precision applies unless the first input argument is explicitly single precision. For example, if you call the `sin` function with an integer argument, a cast of the input argument to a floating-point number of type `double` replaces the original argument. This behavior ensures consistent results between Simulink blocks and Stateflow charts for calls to C math functions.

To force a call to a single-precision version of a C math function, you must explicitly cast the function argument using the `single` cast operator. This method works only when a single-precision version of the function exists in the selected Target Function Library as it would in the 'C99 (ISO)' Target Function Library. For more information, see [Calling C Functions in Actions and Type Cast Operations in the Stateflow User's Guide](#).

Change in Text and Visibility of Parameter Prompt for Easier Use with Fixed-Point Advisor and Fixed-Point Tool

In the Data properties dialog box, the **Lock output scaling against changes by the autoscaling tool** check box is now **Lock data type setting against changes by the fixed-point tools**. Previously, this check box was visible only if you entered an expression or a fixed-point data type, such as `fixdt(1,16,0)`. This check box is now visible for any data type specification. This enhancement enables you to lock the current data type settings on the dialog box against changes that the Fixed-Point Advisor or Fixed-Point Tool chooses.

For more information, see [Fixed-Point Data Properties and Automatic Scaling of Stateflow Fixed-Point Data](#) in the Stateflow User's Guide.

Charts Closed By Default When Opening Models Saved in Formats of Earlier Versions

If you save a model with Stateflow charts in the format of an earlier version, the charts appear closed when you open the new MDL-file.

New and Enhanced Demos

The following demo has been added in R2009b:

Demo...	Shows how you can...
sf_aircraft	Design a fault detection, isolation, and recovery (FDIR) application for a pair of aircraft elevators with redundant actuators

The following demo has been enhanced in R2009b:

Demo...	Now...
sldemo_fuelsys	Uses enumerated data types and Simulink functions in the Stateflow chart to model control logic for the fuel rate control system

R2009a

Version: 7.3
New Features: Yes
Bug Fixes: Yes

Support for Saving the Complete Simulation State at a Specific Time

You can save the complete simulation state at a specific time and then load that state for further simulation. This enhancement provides these benefits:

- Enables running isolated segments of a simulation without starting from time $t = 0$, which saves time
- Enables testing of the same chart configuration with different settings
- Enables testing of hard-to-reach chart configurations by loading a specific simulation state

For more information, see [Saving and Restoring Simulations with SimState](#) in the Stateflow User's Guide.

Enhanced Support for Enumerated Data Types

In R2009a, you can use enumerated data in Embedded MATLAB functions, truth table functions that use Embedded MATLAB action language, and Truth Table blocks. See *Using Enumerated Data in Stateflow Charts* in the *Stateflow User's Guide*.

New Boolean Keywords in Stateflow Action Language

You can now use `true` and `false` as Boolean keywords in Stateflow action language. For more information, see [Supported Symbols in Actions](#) in the [Stateflow User's Guide](#).

Enhanced Control of Inlining State Functions in Generated Code

In R2009a, a new **Function Inline Option** parameter is available in the State properties dialog box. This parameter enables better control of inlining state functions in generated code, which provides these benefits:

- Prevents small changes to a model from causing major changes to the structure of generated code
- Enables easier manual inspection of generated code, because of a one-to-one mapping between the code and the model

For more information, see [Controlling Inlining of State Functions in Generated Code](#) in the Stateflow User's Guide.

New Diagnostic to Detect Unintended Backtracking Behavior in Flow Graphs

A new diagnostic detects unintended backtracking behavior in flow graphs during simulation. A warning message appears, with suggestions on how to fix the flow graph to prevent unintended backtracking. For more information, see Best Practices for Creating Flow Graphs in the Stateflow User's Guide.

Use of Basic Linear Algebra Subprograms (BLAS) Libraries for Speed

Embedded MATLAB functions in Stateflow charts can now use BLAS libraries to speed up low-level matrix operations during simulation. For more information, see [Simulation Target Pane: General](#) in the Simulink documentation.

Enhanced Support for Replacing C Math Functions with Target-Specific Implementations

You can now replace the `pow` function with a target-specific implementation. For more information about Target Function Libraries, see:

- Replacement of C Math Library Functions with Target-Specific Implementations
- Replacing Operators with Target-Specific Implementations

Smart Transitions Now Prefer Straight Lines

In R2009a, the graphical behavior of smart transitions has been enhanced as follows:

- Smart transitions maintain straight lines between states and junctions whenever possible. Previously, smart transitions would preserve curved lines.
- When you drag a smart transition radially around a junction, the end on the junction follows the tip to maintain a straight line by default. Previously, the end on the junction would maintain its original location and not follow the tip of the transition.

For more information, see [What Smart Transitions Do](#) in the Stateflow User's Guide.

Clicking Up-Arrow Button in the Stateflow Editor Closes Top-Level Chart

When a top-level chart appears in the Stateflow Editor, clicking the up-arrow button in the toolbar causes the chart to close and the Simulink model that contains the chart to appear. This behavior is consistent with clicking the up-arrow button in the toolbar of a Simulink subsystem window.

Previously, clicking the up-arrow button for a top-level chart would cause the Simulink model to appear, but the chart would not close. For more information, see *Navigating Subcharts in the Stateflow User's Guide*.

Enhanced Type Resolution for Symbols

In R2009a, type resolution for Stateflow data has been enhanced to support any MATLAB expression that evaluates to a type.

Enhanced Code Generation for Stateflow Events

In R2009a, the generated code for managing Stateflow events uses a deterministic numbering method. This enhancement minimizes unnecessary differences in the generated code for charts between R2009a and any future release.

Enhanced Real-Time Workshop Generated Code for Charts with Simulink Functions

In R2009a, Real-Time Workshop generated code for charts with Simulink functions no longer uses unneeded global variables for the function inputs and outputs. The interface can be represented by local temporary variables or completely eliminated by optimizations, such as expression folding. This enhancement provides reduced RAM consumption and faster execution time.

Use of `en`, `du`, `ex`, `entry`, `during`, and `exit` for Data and Event Names Being Disallowed in a Future Version

Compatibility Considerations: Yes

In a future version of Stateflow software, use of `en`, `du`, `ex`, `entry`, `during`, or `exit` for naming data or events will be disallowed. In R2009a, a warning message appears when you run a model that contains any of these keywords as the names of data or events.

Compatibility Considerations

To avoid warning messages, rename any data or event that uses `en`, `du`, `ex`, `entry`, `during`, or `exit` as an identifier.

Support for Machine-Parented Events Being Removed in a Future Version

Compatibility Considerations: Yes

In a future version of Stateflow software, support for machine-parented events will be removed. In R2009a, a warning message appears when you simulate models that contain events at the machine level.

Compatibility Considerations

To prevent undesired behavior for simulation and code generation, do not use machine-parented events. For simulation, broadcasting an event to all charts in your model causes the following to occur:

- Charts wake up without regard to data dependencies.
- Charts that are disabled might wake up.
- Charts that use function-call or edge-triggered events wake up.
- Charts unrelated to the event wake up.
- Infinite recursive cycles can occur because the chart that broadcasts the event wakes up.

For code generation, machine-parented events prevent code reuse for the entire model.

R2008b

Version: 7.2
New Features: Yes
Bug Fixes: Yes

Support for Embedding Simulink Function-Call Subsystems in a Stateflow Chart

You can use a Simulink function to embed a function-call subsystem in a Stateflow chart. You fill this function with Simulink blocks and call it in state actions and on transitions. Like graphical functions, truth table functions, and Embedded MATLAB functions, you can use multiple return values with Simulink functions.

For more information, see [Using Simulink Functions in Stateflow Charts](#) in the Stateflow User's Guide.

Support for Using Enumerated Data Types in a Stateflow Chart

You can use data of an enumerated type in a Stateflow chart.

For more information, see [Using Enumerated Data in Stateflow Charts](#) in the [Stateflow User's Guide](#) and [Enumerations and Modeling](#) in the [Simulink User's Guide](#).

New Alignment, Distribution, and Resizing Commands for Stateflow Charts

You can use alignment, distribution, and resizing commands on graphical chart objects, such as states, functions, and boxes.

For more information, see [Formatting Chart Objects](#) in the Stateflow User's Guide.

Unified Simulation and Embeddable Code Generation Options for Stateflow Charts and Truth Table Blocks

Compatibility Considerations: Yes

You can use a single dialog box to specify simulation and embeddable code generation options that apply to Stateflow charts and Truth Table blocks. These changes apply:

Type of Model	Simulation Options	Embeddable Code Generation Options
Nonlibrary	Migrated from the Simulation Target dialog box to the Configuration Parameters dialog box See “GUI Changes in Simulation Options for Nonlibrary Models” on page 139	Enhanced with new options in the Real-Time Workshop pane of the Configuration Parameters dialog box See “GUI Enhancements in Real-Time Workshop Code Generation Options for Nonlibrary Models” on page 150
Library	Migrated from the Simulation Target dialog box to the Configuration Parameters dialog box See “GUI Changes in Simulation Options for Library Models” on page 145	Migrated from the RTW Target dialog box to the Configuration Parameters dialog box See “GUI Changes in Real-Time Workshop Code Generation Options for Library Models” on page 154

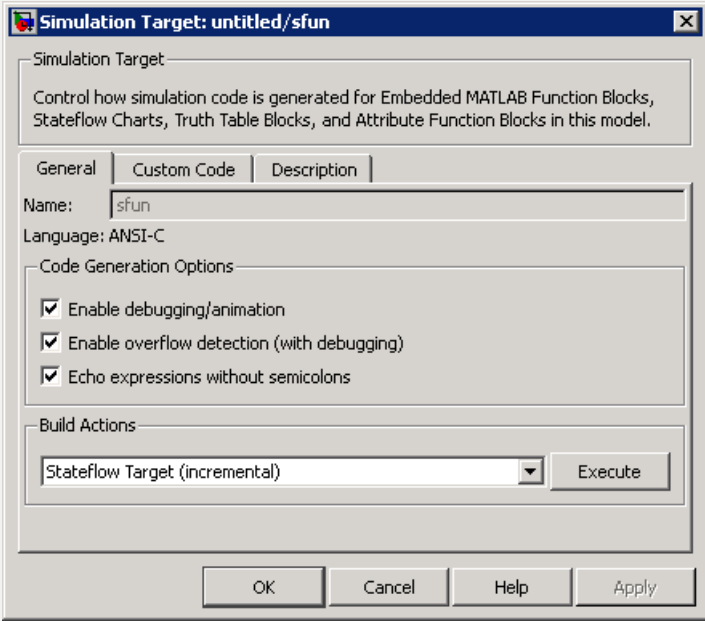
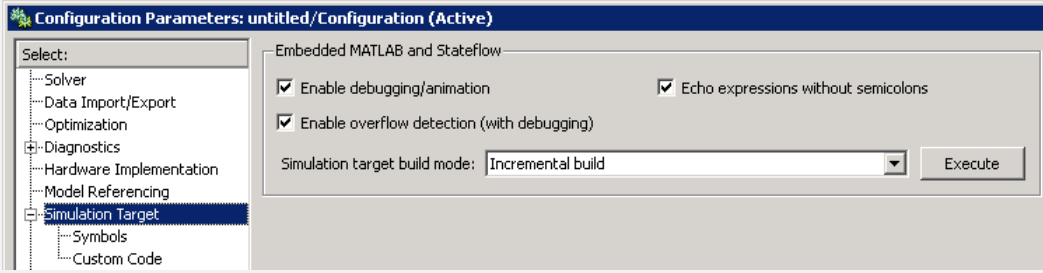
For more information, see Configuration Parameters Dialog Box in the Simulink Graphical User Interface and Building Targets in the Stateflow User’s Guide.

For compatibility information, see .

GUI Changes in Simulation Options for Nonlibrary Models

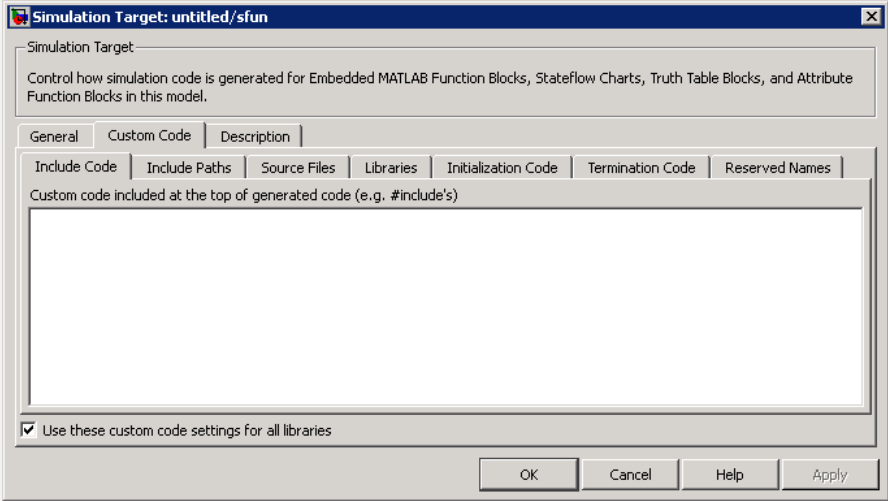
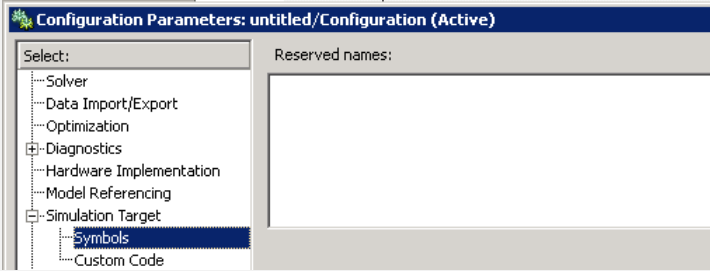
The following sections describe changes in the panes of the Simulation Target dialog box for nonlibrary models.

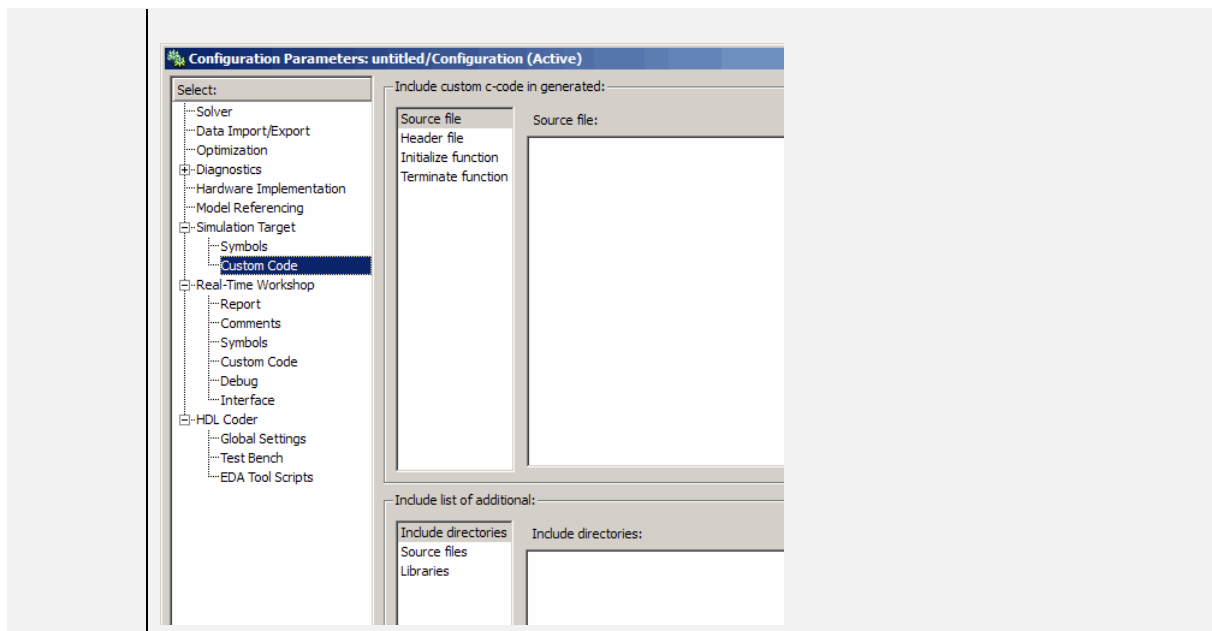
Changes for the General Pane of the Simulation Target Dialog Box

Release	Appearance
Previous	<p>General pane of the Simulation Target dialog box</p> 
New	<p>Simulation Target pane of the Configuration Parameters dialog box</p> 

For details, see “Nonlibrary Models: Mapping of GUI Options from the Simulation Target Dialog Box to the Configuration Parameters Dialog Box” on page 143.

Changes for the Custom Code Pane of the Simulation Target Dialog Box

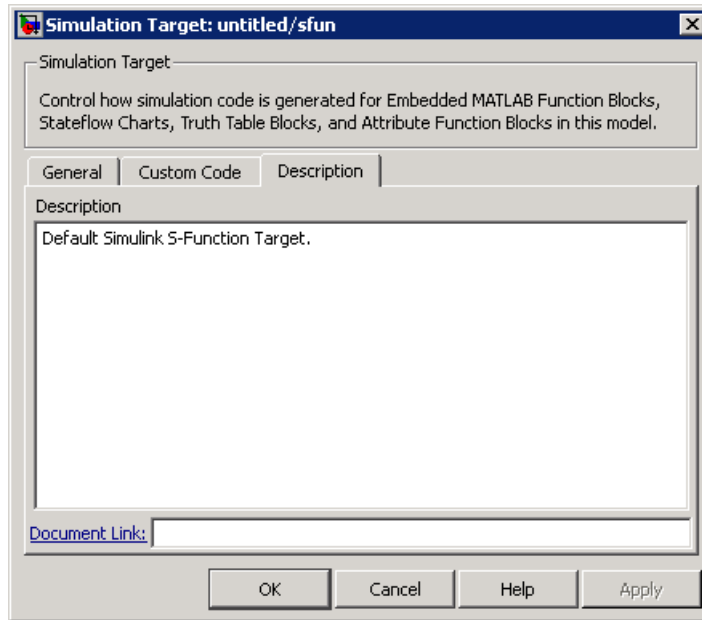
Release	Appearance
Previous	<p>Custom Code pane of the Simulation Target dialog box</p> 
New	<p>Simulation Target > Symbols pane of the Configuration Parameters dialog box</p> 
New	<p>Simulation Target > Custom Code pane of the Configuration Parameters dialog box</p>

Release Appearance

For details, see “Nonlibrary Models: Mapping of GUI Options from the Simulation Target Dialog Box to the Configuration Parameters Dialog Box” on page 143.

Changes for the Description Pane of the Simulation Target Dialog Box

In previous releases, the **Description** pane of the Simulation Target dialog box appeared as follows.



In R2008b, these options are no longer available. For older models where the **Description** pane contained information, the text is now accessible only in the Model Explorer. When you select **Simulink Root > Configuration Preferences** in the **Model Hierarchy** pane, the text appears in the **Description** field for that model.

Nonlibrary Models: Mapping of GUI Options from the Simulation Target Dialog Box to the Configuration Parameters Dialog Box

For nonlibrary models, the following table maps each GUI option in the Simulation Target dialog box to the equivalent in the Configuration Parameters dialog box. The options are listed in order of appearance in the Simulation Target dialog box.

Old Option in the Simulation Target Dialog Box	New Option in the Configuration Parameters Dialog Box	Default Value of New Option
General > Enable debugging / animation	Simulation Target > Enable debugging / animation	on
General > Enable overflow detection (with debugging)	Simulation Target > Enable overflow detection (with debugging)	on
General > Echo expressions without semicolons	Simulation Target > Echo expressions without semicolons	on
General > Build Actions	Simulation Target > Simulation target build mode	Incremental build
None	Simulation Target > Custom Code > Source file	''
Custom Code > Include Code	Simulation Target > Custom Code > Header file	''
Custom Code > Include Paths	Simulation Target > Custom Code > Include directories	''
Custom Code > Source Files	Simulation Target > Custom Code > Source files	''
Custom Code > Libraries	Simulation Target > Custom Code > Libraries	''
Custom Code > Initialization Code	Simulation Target > Custom Code > Initialize function	''
Custom Code > Termination Code	Simulation Target > Custom Code > Terminate function	''
Custom Code > Reserved Names	Simulation Target > Symbols > Reserved names	{}

Old Option in the Simulation Target Dialog Box	New Option in the Configuration Parameters Dialog Box	Default Value of New Option
Custom Code > Use these custom code settings for all libraries	None	Not applicable
Description > Description	None <hr/> Note If you load an older model that contained user-specified text in the Description field, that text now appears in the Model Explorer. When you select Simulink Root > Configuration Preferences in the Model Hierarchy pane, the text appears in the Description field for that model. <hr/>	Not applicable
Description > Document Link	None	Not applicable

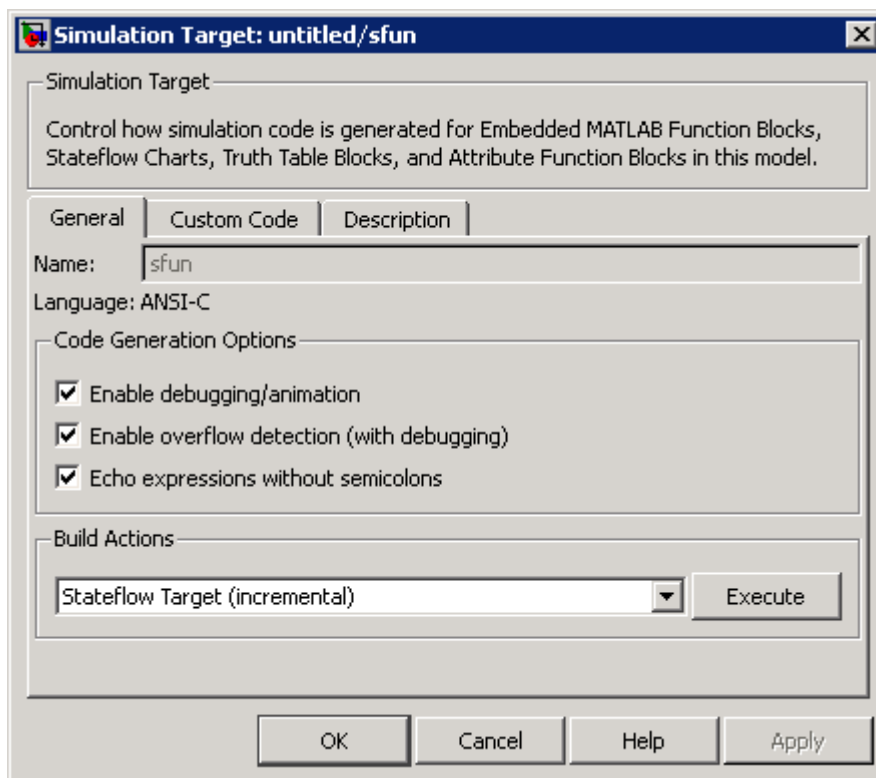
Note For nonlibrary models, **Simulation Target** options in the Configuration Parameters dialog box are also available in the Model Explorer. When you select **Simulink Root > Configuration Preferences** in the **Model Hierarchy** pane, you can select **Simulation Target** in the **Contents** pane to access the options.

GUI Changes in Simulation Options for Library Models

The following sections describe changes in the panes of the Simulation Target dialog box for library models.

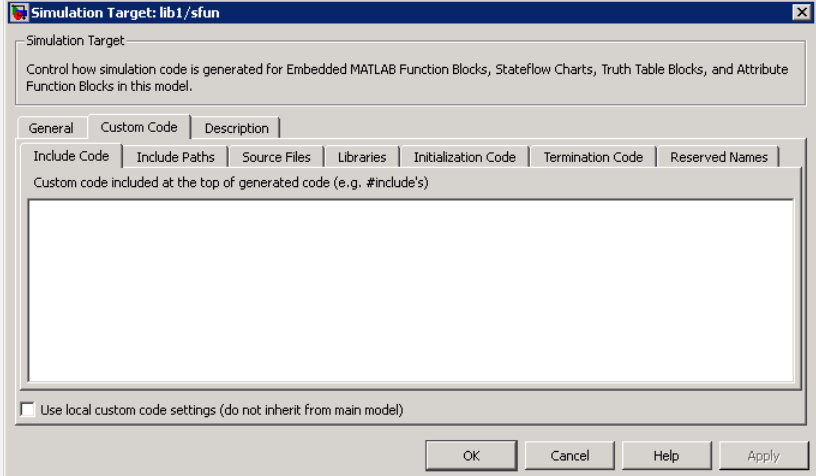
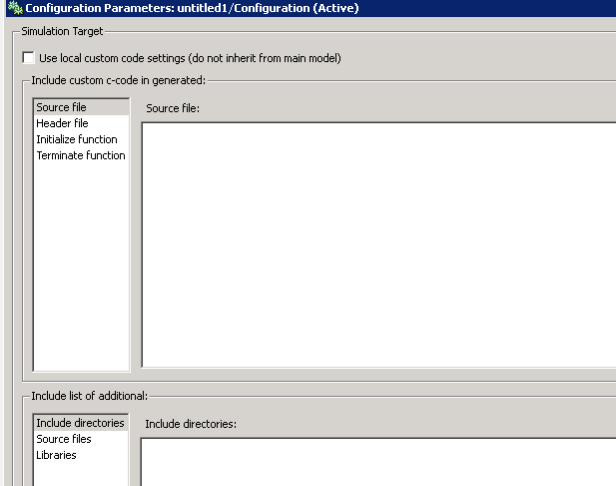
Changes for the General Pane of the Simulation Target Dialog Box

In previous releases, the **General** pane of the Simulation Target dialog box for library models appeared as follows.



In R2008b, these options are no longer available. All library models inherit these option settings from the main model to which the libraries are linked.

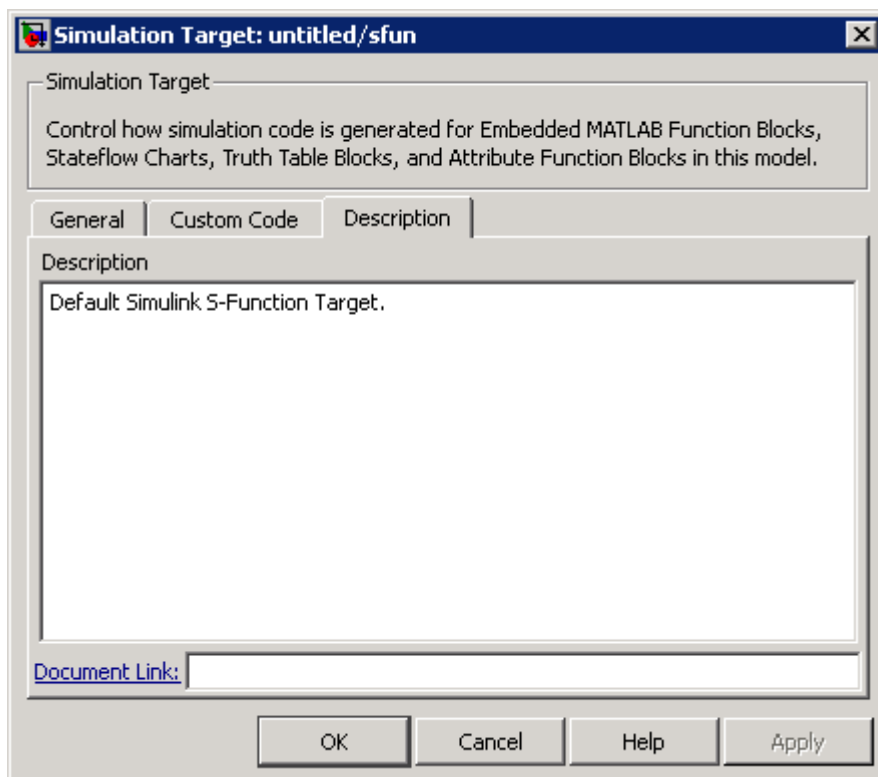
Changes for the Custom Code Pane of the Simulation Target Dialog Box

Release	Appearance
Previous	<p data-bbox="277 395 976 425">Custom Code pane of the Simulation Target dialog box</p> 
New	<p data-bbox="277 963 1139 992">Simulation Target pane of the Configuration Parameters dialog box</p> 

For details, see “Library Models: Mapping of GUI Options from the Simulation Target Dialog Box to the Configuration Parameters Dialog Box” on page 149.

Changes for the Description Pane of the Simulation Target Dialog Box

In previous releases, the **Description** pane of the Simulation Target dialog box appeared as follows.



In R2008b, these options are no longer available. For older models where the **Description** pane contained information, the text is discarded.

Library Models: Mapping of GUI Options from the Simulation Target Dialog Box to the Configuration Parameters Dialog Box

For library models, the following table maps each GUI option in the Simulation Target dialog box to the equivalent in the Configuration Parameters dialog box. The options are listed in order of appearance in the Simulation Target dialog box.

Old Option in the Simulation Target Dialog Box	New Option in the Configuration Parameters Dialog Box	Default Value of New Option
General > Enable debugging / animation	None	Not applicable
General > Enable overflow detection (with debugging)	None	Not applicable
General > Echo expressions without semicolons	None	Not applicable
General > Build Actions	None	Not applicable
None	Simulation Target > Source file	''
Custom Code > Include Code	Simulation Target > Header file	''
Custom Code > Include Paths	Simulation Target > Include directories	''
Custom Code > Source Files	Simulation Target > Source files	''
Custom Code > Libraries	Simulation Target > Libraries	''
Custom Code > Initialization Code	Simulation Target > Initialize function	''
Custom Code > Termination Code	Simulation Target > Terminate function	''
Custom Code > Reserved Names	None	Not applicable

Old Option in the Simulation Target Dialog Box	New Option in the Configuration Parameters Dialog Box	Default Value of New Option
Custom Code > Use local custom code settings (do not inherit from main model)	Simulation Target > Use local custom code settings (do not inherit from main model)	off
Description > Description	None	Not applicable
Description > Document Link	None	Not applicable

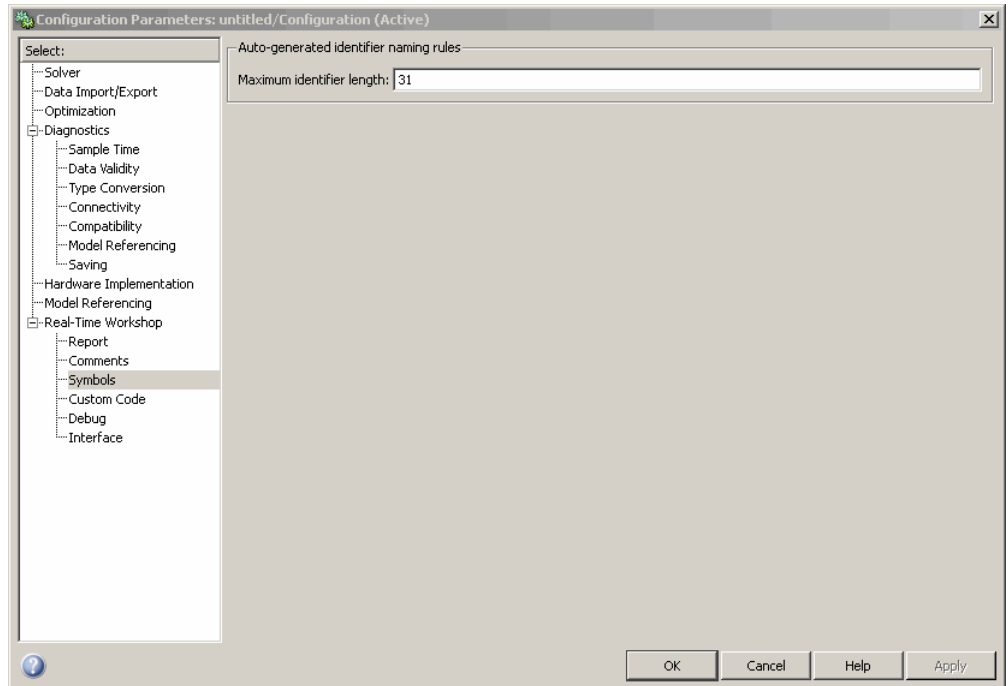
Note For library models, **Simulation Target** options in the Configuration Parameters dialog box are not available in the Model Explorer.

GUI Enhancements in Real-Time Workshop Code Generation Options for Nonlibrary Models

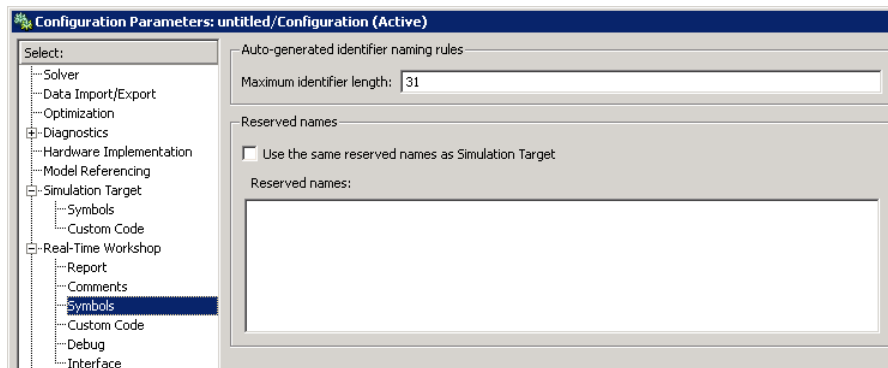
The following sections describe enhancements to the **Real-Time Workshop** pane of the Configuration Parameters dialog box for nonlibrary models.

Enhancement for the Real-Time Workshop: Symbols Pane of the Configuration Parameters Dialog Box

In previous releases, the **Real-Time Workshop > Symbols** pane of the Configuration Parameters dialog box appeared as follows.



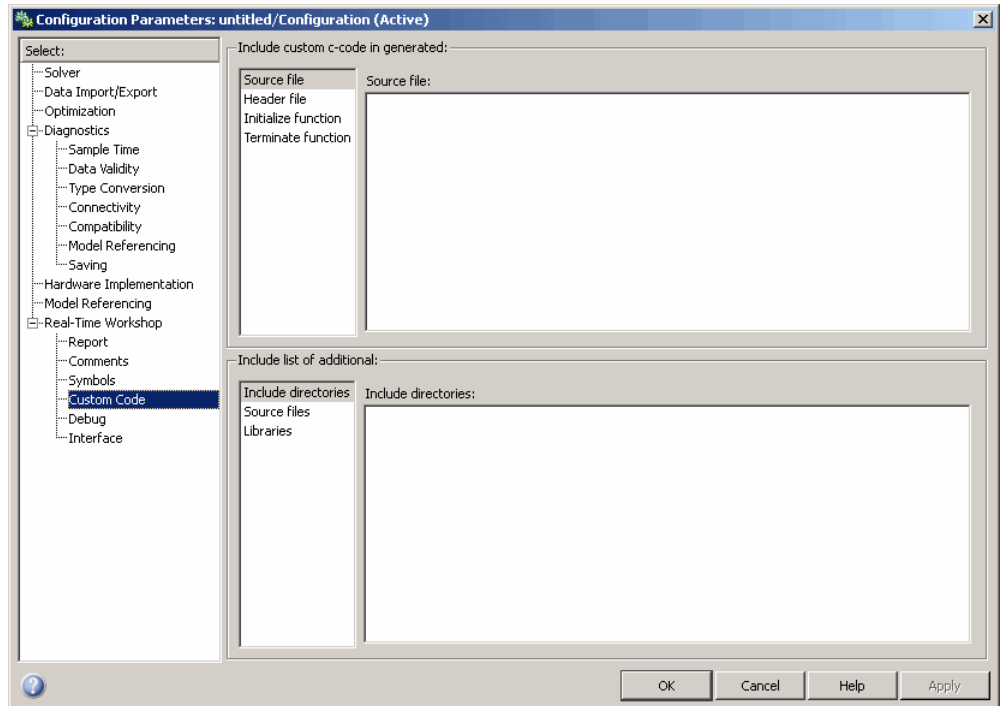
In R2008b, a new option is available in this pane: **Reserved names**. You can use this option to specify a set of keywords that the Real-Time Workshop build process should not use. This action prevents naming conflicts between functions and variables from external environments and identifiers in the generated code.



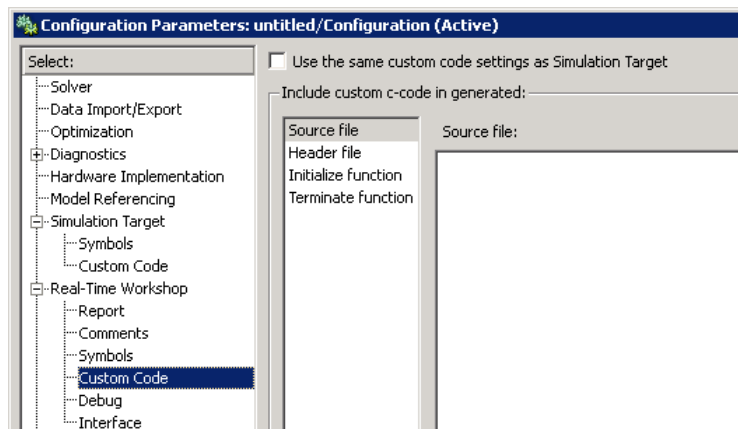
You can also choose to use the reserved names specified in the **Simulation Target > Symbols** pane to avoid entering the same information twice for the nonlibrary model. Select the **Use the same reserved names as Simulation Target** check box.

Enhancement for the Real-Time Workshop: Custom Code Pane of the Configuration Parameters Dialog Box

In previous releases, the **Real-Time Workshop > Custom Code** pane of the Configuration Parameters dialog box appeared as follows.



In R2008b, a new option is available in this pane: **Use the same custom code settings as Simulation Target**. You can use this option to copy the custom code settings from the **Simulation Target > Custom Code** pane to avoid entering the same information twice for the nonlibrary model.

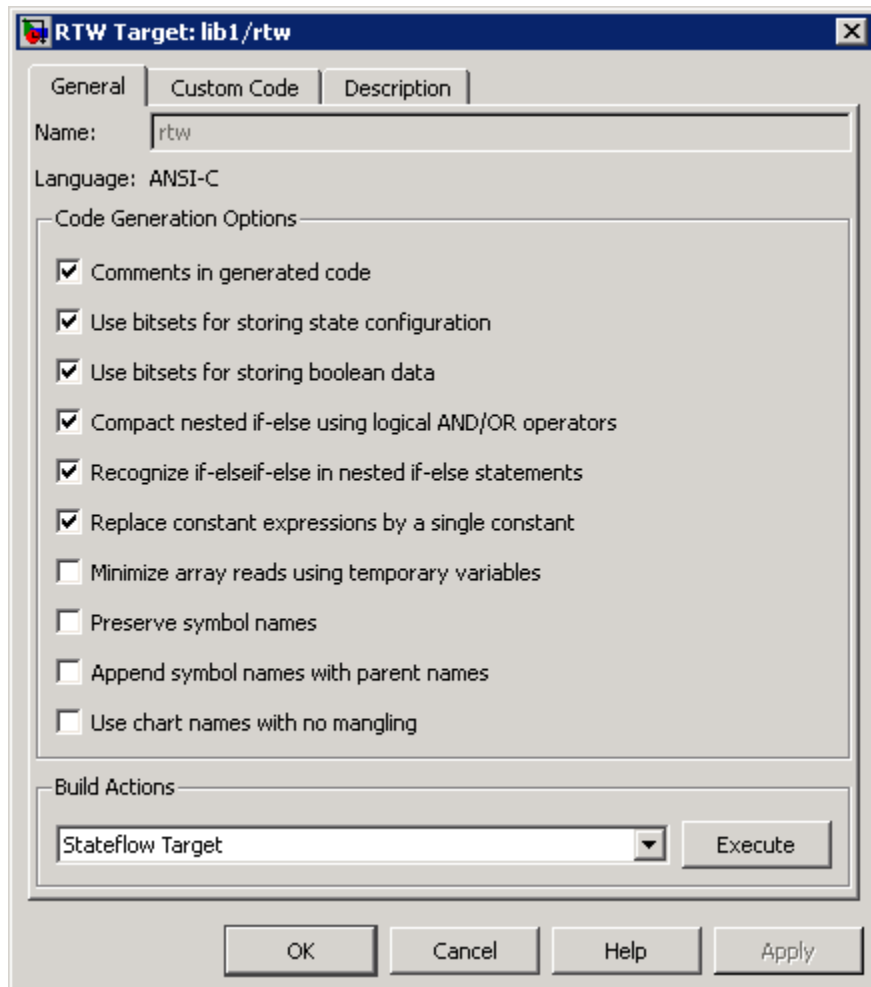


GUI Changes in Real-Time Workshop Code Generation Options for Library Models

The following sections describe changes in the panes of the RTW Target dialog box for library models.

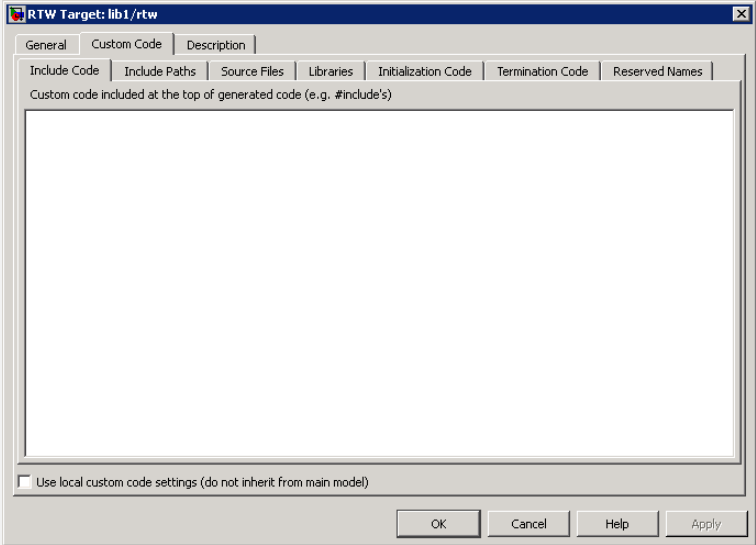
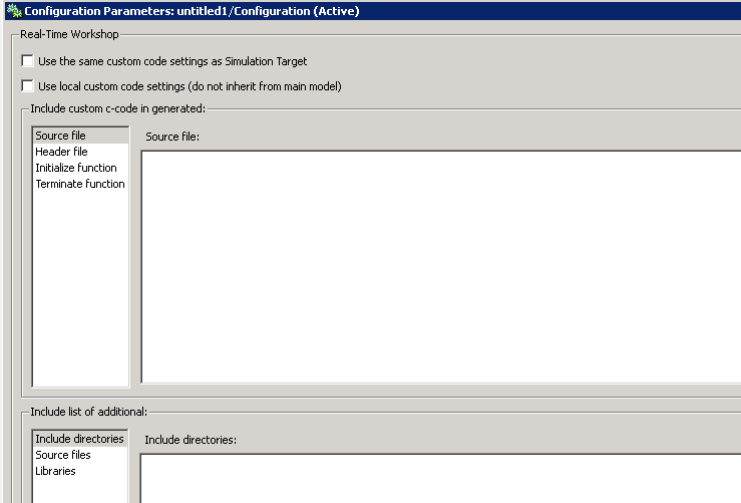
Changes for the General Pane of the RTW Target Dialog Box

In previous releases, the **General** pane of the RTW Target dialog box for library models appeared as follows.



In R2008b, these options are no longer available. During Real-Time Workshop code generation, options specified for the main model are used.

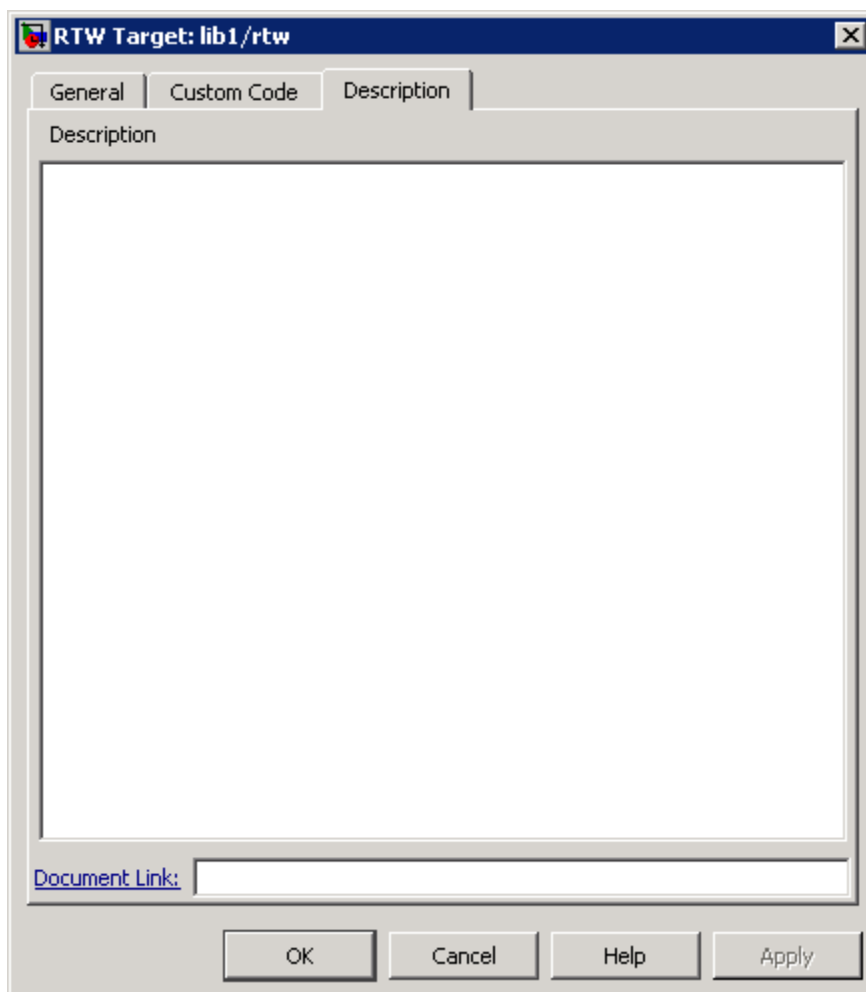
Changes for the Custom Code Pane of the RTW Target Dialog Box

Release	Appearance
Previous	<p data-bbox="278 395 902 423">Custom Code pane of the RTW Target dialog box</p> 
New	<p data-bbox="278 1027 1175 1055">Real-Time Workshop pane of the Configuration Parameters dialog box</p> 

For details, see “Library Models: Mapping of GUI Options from the RTW Target Dialog Box to the Configuration Parameters Dialog Box” on page 158.

Changes for the Description Pane of the RTW Target Dialog Box

In previous releases, the **Description** pane of the RTW Target dialog box appeared as follows.



In R2008b, these options are no longer available. For older models where the **Description** pane contained information, the text is discarded.

Library Models: Mapping of GUI Options from the RTW Target Dialog Box to the Configuration Parameters Dialog Box

For library models, the following table maps each GUI option in the RTW Target dialog box to the equivalent in the Configuration Parameters dialog box. The options are listed in order of appearance in the RTW Target dialog box.

Old Option in the RTW Target Dialog Box	New Option in the Configuration Parameters Dialog Box	Default Value of New Option
General > Comments in generated code	None	Not applicable
General > Use bitsets for storing state configuration	None	Not applicable
General > Use bitsets for storing boolean data	None	Not applicable
General > Compact nested if-else using logical AND/OR operators	None	Not applicable
General > Recognize if-elseif-else in nested if-else statements	None	Not applicable
General > Replace constant expressions by a single constant	None	Not applicable
General > Minimize array reads using temporary variables	None	Not applicable
General > Preserve symbol names	None	Not applicable

Old Option in the RTW Target Dialog Box	New Option in the Configuration Parameters Dialog Box	Default Value of New Option
General > Append symbol names with parent names	None	Not applicable
General > Use chart names with no mangling	None	Not applicable
General > Build Actions	None	Not applicable
None	Real-Time Workshop > Source file	''
Custom Code > Include Code	Real-Time Workshop > Header file	''
Custom Code > Include Paths	Real-Time Workshop > Include directories	''
Custom Code > Source Files	Real-Time Workshop > Source files	''
Custom Code > Libraries	Real-Time Workshop > Libraries	''
Custom Code > Initialization Code	Real-Time Workshop > Initialize function	''
Custom Code > Termination Code	Real-Time Workshop > Terminate function	''
Custom Code > Reserved Names	None	Not applicable
Custom Code > Use local custom code settings (do not inherit from main model)	Real-Time Workshop > Use local custom code settings (do not inherit from main model)	off
None	Real-Time Workshop > Use the same custom code settings as Simulation Target	off

Old Option in the RTW Target Dialog Box	New Option in the Configuration Parameters Dialog Box	Default Value of New Option
Description > Description	None	Not applicable
Description > Document Link	None	Not applicable

Note For library models, **Real-Time Workshop** options in the Configuration Parameters dialog box are not available in the Model Explorer.

Mapping of Target Object Properties to Parameters in the Configuration Parameters Dialog Box

Previously, you could programmatically set options for simulation and embeddable code generation by accessing the API properties of Target objects `sfun` and `rtw`, respectively. In R2008b, the API properties of Target objects `sfun` and `rtw` are replaced by parameters that you configure using the commands `get_param` and `set_param`.

For compatibility details, see [and](#) [.](#)

Mapping of Object Properties to Simulation Parameters for Nonlibrary Models

The following table maps API properties of the Target object `sfun` for nonlibrary models to the equivalent parameters in R2008b. Object properties are listed in alphabetical order; those not listed in the table do not have equivalent parameters in R2008b.

Old sfun Object Property	Old Option in the Simulation Target Dialog Box	New Configuration Parameter	New Option in the Configuration Parameters Dialog Box
CodeFlagsInfo ('debug')	General > Enable debugging / animation	SFSimEnableDebug string - off, on	Simulation Target > Enable debugging / animation
CodeFlagsInfo ('overflow')	General > Enable overflow detection (with debugging)	SFSimOverflowDetection string - off, on	Simulation Target > Enable overflow detection (with debugging)
CodeFlagsInfo ('echo')	General > Echo expressions without semicolons	SFSimEcho string - off, on	Simulation Target > Echo expressions without semicolons
CustomCode	Custom Code > Include Code	SimCustomHeaderCode string - ''	Simulation Target > Custom Code > Header file
CustomInitializer	Custom Code > Initialization Code	SimCustomInitializer string - ''	Simulation Target > Custom Code > Initialize function
CustomTerminator	Custom Code > Termination Code	SimCustomTerminator string - ''	Simulation Target > Custom Code > Terminate function

Old sfun Object Property	Old Option in the Simulation Target Dialog Box	New Configuration Parameter	New Option in the Configuration Parameters Dialog Box
ReservedNames	Custom Code > Reserved Names	SimReservedNameArray <i>string array - {}</i>	Simulation Target > Symbols > Reserved names
UserIncludeDirs	Custom Code > Include Paths	SimUserIncludeDirs <i>string - ''</i>	Simulation Target > Custom Code > Include directories
UserLibraries	Custom Code > Libraries	SimUserLibraries <i>string - ''</i>	Simulation Target > Custom Code > Libraries
UserSources	Custom Code > Source Files	SimUserSources <i>string - ''</i>	Simulation Target > Custom Code > Source files

Mapping of Object Properties to Simulation Parameters for Library Models

The following table maps API properties of the Target object sfun for library models to the equivalent parameters in R2008b. Object properties are listed in alphabetical order; those not listed in the table do not have equivalent parameters in R2008b.

Old sfun Object Property	Old Option in the Simulation Target Dialog Box	New Configuration Parameter	New Option in the Configuration Parameters Dialog Box
CustomCode	Custom Code > Include Code	SimCustomHeaderCode <i>string</i> - ''	Simulation Target > Header file
CustomInitializer	Custom Code > Initialization Code	SimCustomInitializer <i>string</i> - ''	Simulation Target > Initialize function
CustomTerminator	Custom Code > Termination Code	SimCustomTerminator <i>string</i> - ''	Simulation Target > Terminate function
UseLocalCustomCodeSettings	Custom Code > Use local custom code settings (do not inherit from main model)	SimUseLocalCustomCode <i>string</i> - off , on	Simulation Target > Use local custom code settings (do not inherit from main model)
UserIncludeDirs	Custom Code > Include Paths	SimUserIncludeDirs <i>string</i> - ''	Simulation Target > Include directories
UserLibraries	Custom Code > Libraries	SimUserLibraries <i>string</i> - ''	Simulation Target > Libraries
UserSources	Custom Code > Source Files	SimUserSources <i>string</i> - ''	Simulation Target > Source files

Mapping of Object Properties to Code Generation Parameters for Library Models

The following table maps API properties of the Target object `rtw` for library models to the equivalent parameters in R2008b. Object properties are listed in alphabetical order; those not listed in the table do not have equivalent parameters in R2008b.

Old <code>rtw</code> Object Property	Old Option in the RTW Target Dialog Box	New Configuration Parameter	New Option in the Configuration Parameters Dialog Box
<code>CustomCode</code>	Custom Code > Include Code	<code>CustomHeaderCode</code> <i>string</i> - ''	Real-Time Workshop > Header file
<code>CustomInitializer</code>	Custom Code > Initialization Code	<code>CustomInitializer</code> <i>string</i> - ''	Real-Time Workshop > Initialize function
<code>CustomTerminator</code>	Custom Code > Termination Code	<code>CustomTerminator</code> <i>string</i> - ''	Real-Time Workshop > Terminate function
<code>UseLocalCustomCodeSettings</code>	Custom Code > Use local custom code settings (do not inherit from main model)	<code>RTWUseLocalCustomCode</code> <i>string</i> - off , on	Real-Time Workshop > Use local custom code settings (do not inherit from main model)
<code>UserIncludeDirs</code>	Custom Code > Include Paths	<code>CustomInclude</code> <i>string</i> - ''	Real-Time Workshop > Include directories

Old rtw Object Property	Old Option in the RTW Target Dialog Box	New Configuration Parameter	New Option in the Configuration Parameters Dialog Box
UserLibraries	Custom Code > Libraries	CustomLibrary <i>string</i> - ''	Real-Time Workshop > Libraries
UserSources	Custom Code > Source Files	CustomSource <i>string</i> - ''	Real-Time Workshop > Source files

New Parameters in the Configuration Parameters Dialog Box for Simulation and Embeddable Code Generation

In R2008b, new parameters are added to the Configuration Parameters dialog box for simulation and embeddable code generation.

New Simulation Parameters for Nonlibrary Models

The following table lists the new simulation parameters that apply to nonlibrary models.

New Configuration Parameter	New Option in the Configuration Parameters Dialog Box	Description
SimBuildMode <i>string</i> – sf_incremental_build , sf_nonincremental_build , sf_make , sf_make_clean , sf_make_clean_objects	Simulation Target > Simulation target build mode	Specifies how you build the simulation target for a model.
SimCustomSourceCode <i>string</i> - ''	Simulation Target > Custom Code > Source file	Enter code lines to appear near the top of a generated source code file.

New Simulation Parameter for Library Models

The following table lists the new simulation parameter that applies to library models.

New Configuration Parameter	New Option in the Configuration Parameters Dialog Box	Description
SimCustomSourceCode <i>string</i> - ''	Simulation Target > Source file	Enter code lines to appear near the top of a generated source code file.

New Code Generation Parameters for Nonlibrary Models

The following table lists the new code generation parameters that apply to nonlibrary models.

New Configuration Parameter	New Option in the Configuration Parameters Dialog Box	Description
ReservedNameArray <i>string array</i> - {}	Real-Time Workshop > Symbols > Reserved names	Enter the names of variables or functions in the generated code that match the names of variables or functions specified in custom code.
RTWUseSimCustomCode <i>string</i> – off , on	Real-Time Workshop > Custom Code > Use the same custom code settings as Simulation Target	Specify whether to use the same custom code settings as those specified for simulation.
UseSimReservedNames <i>string</i> – off , on	Real-Time Workshop > Symbols > Use the same reserved names as Simulation Target	Specify whether to use the same reserved names as those specified for simulation.

New Code Generation Parameters for Library Models

The following table lists the new code generation parameters that apply to library models.

New Configuration Parameter	New Option in the Configuration Parameters Dialog Box	Description
CustomSourceCode <i>string</i> – ''	Real-Time Workshop > Source file	Enter code lines to appear near the top of a generated source code file.
RTWUseSimCustomCode <i>string</i> – off , on	Real-Time Workshop > Use the same custom code settings as Simulation Target	Specify whether to use the same custom code settings as those specified for simulation.

Compatibility Considerations

Updating Scripts That Set Options Programmatically for Simulation and Embeddable Code Generation

In previous releases, you could use the Stateflow API to set options for simulation and embeddable code generation by accessing the Target object (sfun or rtw) in a Stateflow machine. For example, you could set simulation options programmatically by running these commands in a MATLAB script:

```
r = slroot;
machine = r.find('-isa','Stateflow.Machine','Name','main_mdl');
t_sim = machine.find('-isa','Stateflow.Target','Name','sfun');
t_sim.setCodeFlag('debug',1);
t_sim.setCodeFlag('overflow',1);
t_sim.setCodeFlag('echo',1);
t_sim.getCodeFlag('debug');
t_sim.getCodeFlag('overflow');
t_sim.getCodeFlag('echo');
```

In R2008b, you must update your scripts to use the `set_param` and `get_param` commands to configure simulation and embeddable code generation. For example, you can update the previous script as follows:

```
cs = getActiveConfigSet(gcs);
set_param(cs,'SFSimEnableDebug','on');
set_param(cs,'SFSimOverflowDetection','on');
```

```

set_param(cs, 'SFSimEcho', 'on');
get_param(cs, 'SFSimEnableDebug');
get_param(cs, 'SFSimOverflowDetection');
get_param(cs, 'SFSimEcho');

```

For information about...	See...
Object properties and their equivalent parameters in R2008b	<p>“Mapping of Target Object Properties to Parameters in the Configuration Parameters Dialog Box” on page 160</p> <hr/> <p>Note Properties of Target objects <code>sfun</code> and <code>rtw</code> that are no longer supported in R2008b cannot be updated using the command-line API. For a list of unsupported properties, see .</p>
Using the <code>set_param</code> and <code>get_param</code> commands	Using Command-Line API to Set Simulation and Code Generation Parameters.

Accessing Target Options for Library Models

In previous releases, you could access target options for library models via the **Tools** menu in the Stateflow Editor or the **Contents** pane of the Model Explorer. In R2008b, you must use the **Tools** menu to access target options for library models. For example, to specify parameters for the simulation target, select **Tools > Open Simulation Target** in the Stateflow Editor.

What Happens When You Load an Older Model in R2008b

When you use R2008b to load a model created in an earlier version, dialog box options and the equivalent object properties for simulation and embeddable code generation targets migrate automatically to the Configuration Parameters dialog box, except in the cases that follow.

For the simulation target of a nonlibrary model, these options and properties do not migrate to the Configuration Parameters dialog box. The information is discarded when you load the model, unless otherwise noted.

Option in the Simulation Target Dialog Box of a Nonlibrary Model	Equivalent Object Property
Custom Code > Use these custom code settings for all libraries	ApplyToAllLibs
Description > Description	Description <hr/> Note If you load an older model that contains user-specified text in the Description field, that text now appears in the Model Explorer. When you select Simulink Root > Configuration Preferences in the Model Hierarchy pane, the text appears in the Description field for that model. <hr/>
Description > Document Link	Document

For the simulation target of a library model, these options and properties do not migrate to the Configuration Parameters dialog box. The information is discarded when you load the model.

Option in the Simulation Target Dialog Box of a Library Model	Equivalent Object Property
General > Enable debugging / animation	CodeFlagsInfo('debug')
General > Enable overflow detection (with debugging)	CodeFlagsInfo('overflow')
General > Echo expressions without semicolons	CodeFlagsInfo('echo')
General > Build Actions	None
Custom Code > Reserved Names	ReservedNames
Description > Description	Description
Description > Document Link	Document

For the embeddable code generation target of a library model, these options and properties do not migrate to the Configuration Parameters dialog box. The information is discarded when you load the model.

Option in the RTW Target Dialog Box of a Library Model	Equivalent Object Property
General > Comments in generated code	CodeFlagsInfo('comments')
General > Use bitsets for storing state configuration	CodeFlagsInfo('statebitsets')
General > Use bitsets for storing boolean data	CodeFlagsInfo('databitsets')
General > Compact nested if-else using logical AND/OR operators	CodeFlagsInfo('emitlogicalops')
General > Recognize if-elseif-else in nested if-else statements	CodeFlagsInfo('elseifdetection')
General > Replace constant expressions by a single constant	CodeFlagsInfo('constantfolding')
General > Minimize array reads using temporary variables	CodeFlagsInfo('redundantloadelimination')
General > Preserve symbol names	CodeFlagsInfo('preservenames')
General > Append symbol names with parent names	CodeFlagsInfo('preservenameswithparent')
General > Use chart names with no mangling	CodeFlagsInfo('exportcharts')
General > Build Actions	None
Custom Code > Reserved Names	ReservedNames
Description > Description	Description
Description > Document Link	Document

What Happens When You Save an Older Model in R2008b

When you use R2008b to save a model created in an earlier version, parameters for simulation and embeddable code generation from the Configuration Parameters dialog box are saved. However, properties of API Target objects `sfun` and `rtw` are not saved if those properties do not have an equivalent parameter in the Configuration Parameters dialog box (see). Properties that do not migrate to the Configuration Parameters dialog box are discarded when you load the model. Therefore, old Target object properties are not saved even if you choose to save the model as an older version (for example, R2007a).

Workaround for Library Models If They No Longer Use Local Custom Code Settings

Behavior in R2008a and Earlier Releases

In R2008a and earlier releases, the main model simulation target had a custom code option **Use these custom code settings for all libraries**, or the target property `ApplyToAllLibs`. The library model simulation target had a similar custom code option **Use local custom code settings (do not inherit from main model)**, or the target property `UseLocalCustomCodeSettings`.

The following criteria determined which custom code settings would apply to the library model:

If <code>ApplyToAllLibs</code> for the main model is...	And <code>UseLocalCustomCodeSettings</code> for the library model is...	Then the library model uses...
True	False	Main model custom code
True	True	Local custom code
False	True	Local custom code
False	False	Local custom code (by default, but ambiguous)

The last case was ambiguous, because the main model did not propagate custom code settings and the library model did not specify use of local custom

code settings either. In this case, the default behavior was to use local custom code settings for the library model.

Behavior in R2008b

In R2008b, the **Use these custom code settings for all libraries** option for the main model is removed. The library model either picks up its local custom code settings if specified to do so, or uses the main model custom code settings when the **Use local custom code settings** option is not selected. This change introduces backward incompatibility for older models that use the "False (main model), False (library model)" setup for specifying custom code settings.

Workaround to Prevent Backward Incompatibility

To resolve the ambiguity in older models, you must explicitly select **Use local custom code settings** for the library model when you want the local custom code settings to apply:

- 1** Open the Stateflow simulation target for the library model.
 - a** Load the library model and unlock it.
 - b** Open one of the library charts in the Stateflow Editor.
 - c** Select **Tools > Open Simulation Target**.
- 2** In the dialog box that appears, select **Use local custom code settings (do not inherit from main model)**.

New Pattern Wizard for Consistent Creation of Logic Patterns and Iterative Loops

You can use the Stateflow Pattern Wizard to create commonly used flow graphs such as for-loops in a quick and consistent manner.

For more information, see [Modeling Logic Patterns and Iterative Loops Using Flow Graphs](#).

Support for Initializing Vectors and Matrices in the Data Properties Dialog Box

In the Data properties dialog box, you can initialize vectors and matrices in the **Initial value** field of the **Value Attributes** pane.

For more information, see [How to Define Vectors and Matrices](#).

Change in Default Mode for Ordering Parallel States and Outgoing Transitions

The default mode for ordering parallel states and outgoing transitions is now explicit. When you create a new chart, you define ordering explicitly in the Stateflow Editor. However, if you load a chart that uses implicit ordering, that mode is retained until you switch to explicit ordering.

For more information, see [Execution Order for Parallel States and Evaluation Order for Outgoing Transitions](#).

Optimized Inlining of Code Generated for Stateflow Charts

In R2008b, Real-Time Workshop code generation is enhanced to enable optimized inlining of code generated for Stateflow charts.

More Efficient Parsing for Nonlibrary Models

When you parse a nonlibrary model, library charts that are not linked to this model are ignored. This enhancement enables more efficient parsing for nonlibrary models.

Change in Casting Behavior When Calling MATLAB Functions in a Chart

Compatibility Considerations: Yes

When you call MATLAB functions in a Stateflow chart, scalar inputs are no longer cast automatically to data of type `double`. This behavior applies when you use the `ml` operator to call a built-in or custom MATLAB function. (For details, see `ml` Namespace Operator.)

Compatibility Considerations

Previously, Stateflow generated code for simulation would automatically cast scalar inputs to data of type `double` when calling MATLAB functions in a chart. This behavior has changed. Stateflow charts created in earlier versions now generate errors during simulation if they contain calls to external MATLAB functions that expect scalar inputs of type `double`, but the inputs are of a different data type.

To prevent these errors, you can change the data type of a scalar input to `double` or add an explicit cast to type `double` in the function call. For example, you can change a function call from `ml.function_name(i)` to `ml.function_name(double(i))`.

Ability to Specify Continuous Update Method for Output Data

In R2008b, you can set the **Update Method** of output data in continuous-time charts to **Continuous**. In previous releases, only local data could use a continuous update method.

Use of Output Data with Change Detection Operators Disallowed for Initialize-Outputs-at-Wakeup Mode

Compatibility Considerations: Yes

If you enable the option **Initialize Outputs Every Time Chart Wakes Up** in the Chart properties dialog box, do not use output data as the first argument of a change detection operator. When this option is enabled, the change detection operator returns `false` if the first argument is an output data. In this case, there is no reason to perform change detection. (For details, see [Detecting Changes in Data Values](#).)

Compatibility Considerations

Previously, Stateflow software would allow the use of output data with change detection operators when you enable the option **Initialize Outputs Every Time Chart Wakes Up**. This behavior has changed. Stateflow charts created in earlier versions now generate errors during parsing to prevent such behavior.

Parsing a Stateflow Chart Without Simulation No Longer Detects Unresolved Symbol Errors

To detect unresolved symbol errors in a chart, you must start simulation or update the model diagram. When you parse a chart without simulation or diagram updates, the Stateflow parser does not have access to all the information needed to check for unresolved symbols, such as exported graphical functions from other charts and enumerated data types. Therefore, the parser now skips unresolved symbol detection to avoid generating false error messages. However, if you start simulation or update the model diagram, you invoke the model compilation process, which has full access to the information needed, and unresolved symbols are flagged.

For more information, see [Parsing Stateflow Charts and How to Check for Undefined Symbols](#).

Generation of a Unique Name for a Copied State Limited to States Without Default Labels

If you copy and paste a state in the Stateflow Editor, a unique name is generated for the new state only if the original state does not use the default ? label. For more information, see Copying Graphical Objects.

New Configuration Set Created When Loading Nonlibrary Models with an Active Configuration Reference

Compatibility Considerations: Yes

When you load a nonlibrary model with an active configuration reference for Stateflow charts or Truth Table blocks, a copy of the referenced configuration set is created and attached to your model. The new configuration set is marked active, and the configuration reference is marked inactive. This behavior does not apply to library models.

For information about using configuration references, see [Manage a Configuration Reference](#).

Compatibility Considerations

In previous releases, you could load a nonlibrary model with an active configuration reference for Stateflow charts or Truth Table blocks. In R2008b, the configuration reference becomes inactive after you load the model, and a warning message appears to explain this change in behavior. To restore the configuration reference to its original active state, follow the instructions in the warning message.

For more information, see [Configuration References for Models with Older Simulation Target Settings](#).

R2008a+

Version: 7.1.1
New Features: No
Bug Fixes: Yes

R2008a

Version: 7.1
New Features: Yes
Bug Fixes: Yes

Support for Data with Complex Data Types

Stateflow charts support data with complex data types. You can perform basic arithmetic (addition, subtraction, and multiplication) and relational operations (equal and not equal) on complex data in Stateflow action language. You can also use complex input and output arguments for Embedded MATLAB functions in your chart.

For more information, see [Using Complex Data in Stateflow Charts](#).

Support for Functions with Multiple Outputs

You can specify more than one output argument in graphical functions, truth table functions, and Embedded MATLAB functions. Previously, you could specify only one output for these types of functions.

For more information, see [Graphical Functions for Reusing Logic Patterns and Iterative Loops](#), [Truth Table Functions for Decision-Making Logic](#), and [Using MATLAB Functions in Stateflow Charts](#) in the Stateflow documentation.

Bidirectional Traceability for Navigating Between Generated Code and Stateflow Objects

In previous releases, Real-Time Workshop Embedded Coder software provided bidirectional traceability only for Simulink blocks. In R2008a, bidirectional traceability works between generated code and Stateflow objects.

For embedded real-time (ERT) based targets, you can choose to include traceability comments in the generated code. Using the enhanced traceability report, you can click hyperlinks to go from a line of code to its corresponding object in the model. You can also right-click an object in your model to find its corresponding line of code.

For more information, see [Traceability of Stateflow Objects in Generated Code](#) in the Stateflow documentation.

New Temporal Logic Notation for Defining Absolute Time Periods

You can use a keyword named `sec` to define absolute time periods based on simulation time of your chart. Use this keyword as an input argument for temporal logic operators, such as `after`.

For more information, see [Using Temporal Logic in State Actions and Transitions](#) in the Stateflow documentation.

New temporalCount Operator for Counting Occurrences of Events

You can use the `temporalCount` operator to count occurrences of explicit or implicit events. This operator can also count the seconds of simulation time that elapse during chart execution.

For more information, see [Using Temporal Logic in State Actions and Transitions and Counting Events in the Stateflow documentation](#).

Using a Specific Path to a State for the in Operator

Compatibility Considerations: Yes

When you use the `in` operator to check state activity, you must use a specific path to a state. The operator performs a localized search for states that match the given path by looking in each level of the Stateflow hierarchy between its parent and the chart level. The operator does not do an exhaustive search of all states in the entire chart. If there are no matches or multiple matches, a warning message appears and chart execution stops. The search algorithm must find a unique match to check for state activity.

For more information, see [Checking State Activity in the Stateflow documentation](#).

Compatibility Considerations

Previously, you could use a non-specific path to a state as the argument of the `in` operator, because the operator performed an exhaustive search for all states in the chart that match the given path. In the case of multiple matches, a filtering algorithm broke the tie to produce a unique state for checking activity. This behavior has changed. Stateflow charts created in earlier versions may now generate errors if they contain an `in` operator with a non-specific path to a state.

Enhanced MISRA C Code Generation Support

Stateflow Coder software detects missing `else` statements in `if-else` structures for generated code. This enhancement supports MISRA C rule 14.10.

Enhanced Folder Structure for Generated Code

Code files for simulation and code generation targets now reside in the `slprj` folder. Previously, generated code files resided in the `sfprj` folder.

For more information, see [Generated Code Files for Targets You Build](#) in the Stateflow documentation.

Code Optimization for Simulink Blocks and Stateflow Charts

In R2008a, Real-Time Workshop code generation is enhanced to enable cross-product optimizations between Simulink blocks and Stateflow charts.

New fitToView Method for Zooming Objects in the Stateflow Editor

You can use the API method `fitToView` to zoom in on graphical objects in the Stateflow Editor.

For more information, see [Zooming a Chart Object with the API](#) in the Stateflow documentation.

Generation of a Unique Name for a Copied State

If you copy and paste a state in the Stateflow Editor, a unique name automatically appears for the new state.

For more information, see Copying Graphical Objects in the Stateflow documentation.

New Font Size Options in the Stateflow Editor

In the Stateflow Editor, the font sizes in the **Edit > Set Font Size** menu now include 2-point, 4-point, and 50-point. These font options are also available by right-clicking a text item and choosing **Font Size** from the context menu.

For more information, see [Specifying Colors and Fonts in a Chart](#) in the Stateflow documentation.

New Fixed-Point Details Display in the Data Properties Dialog Box

The Data Type Assistant in the Data properties dialog box now displays status and details of fixed-point data types.

For more information, see [Showing Fixed-Point Details in the Stateflow documentation](#).

“What’s This?” Context-Sensitive Help Available for Simulink Configuration Parameters Dialog

R2008a introduces “What’s This?” context-sensitive help for parameters that appear in the Simulink Configuration Parameters dialog. This feature provides quick access to a detailed description of the parameters, saving you the time it would take to find the information in the Help browser.

To use the “What’s This?” help, do the following:

- 1 Place your cursor over the label of a parameter.
- 2 Right-click. A **What’s This?** context menu appears.

For example, the following figure shows the **What’s This?** context menu appearing after a right-click on the **Start time** parameter in the **Solver** pane.



- 3 Click **What’s This?** A context-sensitive help window appears showing a description of the parameter.

Specifying Scaling Explicitly for Fixed-Point Data

Compatibility Considerations: Yes

When you define a fixed-point data type in a Stateflow chart, you must specify scaling explicitly in the **General** pane of the Data properties dialog box. For example, you cannot enter an incomplete specification such as `fixdt(1,16)` in the **Type** field. If you do not specify scaling explicitly, you will see an error message when you try to simulate your model.

To ensure that the data type definition is valid for fixed-point data, perform one of these steps in the **General** pane of the Data properties dialog box:

- Use a predefined option in the **Type** drop-down menu.
- Use the Data Type Assistant to specify the **Mode** as fixed-point.

For more information, see *Defining Data* in the Stateflow documentation.

Compatibility Considerations

Previously, you could omit scaling in data type definitions for fixed-point data. Such data types were treated as integers with the specified sign and word length. This behavior has changed. Stateflow charts created in earlier versions may now generate errors if they contain fixed-point data with no scaling specified.

Use of Data Store Memory Data in Entry Actions and Default Transitions Disallowed for Execute-at-Initialization Mode

If you enable the option **Execute (enter) Chart At Initialization** in the Chart properties dialog box, you cannot assign data store memory data in state entry actions and default transitions that execute the first time that the chart awakens. You can use data store memory data in state during actions, inner transitions, and outer transitions without any limitations.

Previously, assigning data store memory in state entry actions and default transitions with this option enabled would cause a segmentation violation.

Enhanced Warning Message for Target Hardware That Does Not Support the Data Type in a Chart

If your target hardware does not support the data type you use in a Stateflow chart, a warning message appears when you generate code for that chart. This message appears only if the unsupported data type is present in the chart.

Previously, a warning message appeared if the target hardware did not support a given data type, even when the unsupported data type was not actually used in the chart.

Detection of Division-By-Zero Violations When Debugger Is Off

Stateflow simulation now detects division-by-zero violations in a chart, whether or not you enable the debugger.

Previously, disabling the debugger would prevent detection of division-by-zero violations, which caused MATLAB sessions to crash.

R2007b+

Version: 7.0.1
New Features: No
Bug Fixes: Yes

R2007b

Version: 7.0
New Features: Yes
Bug Fixes: Yes

Enhanced Continuous-Time Support with Zero-Crossing Detection

Compatibility Considerations: Yes

Using enhanced support for modeling continuous-time systems, you can:

- Detect zero crossings on state transitions, enabling accurate simulation of dynamic systems with modal behavior.
- Support the definition of continuous state variables and their derivatives for modeling hybrid systems as state charts with embedded dynamic equations

For more information, see *Modeling Continuous-Time Systems in Stateflow Charts*.

Compatibility Considerations

Previously, Stateflow charts implemented continuous time simulation without maintaining mode in minor time steps or detecting zero crossings. Accurate continuous-time simulation requires several constraints on the allowable constructs in Stateflow charts. Charts created in earlier versions may generate errors if they violate these constraints.

New Super Step Feature for Modeling Asynchronous Semantics

Using a new super step property, you can enable Stateflow charts to take multiple transitions in each simulation time step. For more information, see [Execution of a Chart with Super Step Semantics](#) in the Stateflow documentation.

Support for Inheriting Data Properties from Simulink Signal Objects Via Explicit Resolution

Compatibility Considerations: Yes

You can use a new data property, **Data Must Resolve to Simulink signal object**, to allow local and output data to explicitly inherit the following properties from `Simulink.Signal` objects of the same name that you define in the base workspace or model workspace:

- Size
- Type
- Complexity
- Minimum value
- Maximum value
- Initial value
- Storage class (in Real-Time Workshop generated code)

For more information, see [Resolving Data Properties from Simulink Signal Objects](#) in the Stateflow documentation.

Compatibility Considerations

Stateflow software no longer performs implicit signal resolution, a feature supported for output data only. In prior releases, Stateflow software attempted to resolve outputs implicitly to inherit the size, type, complexity, and storage class of `Simulink.Signal` objects of the same name that existed in the base or model workspace. No other properties could be inherited from Simulink signals.

Now, local as well as output data can inherit additional properties from `Simulink.Signal` objects, but you must enable signal resolution explicitly. In models developed before Version 7.0 (R2007b) that rely on implicit signal resolution, Stateflow charts may not simulate or may generate code with unexpected storage classes. In these cases, Stateflow software automatically disables implicit signal resolution for chart outputs and generates a warning at model load time about possible incompatibilities. Before loading such a

model, make sure you have loaded into the base or model workspace all `Simulink.Signal` objects that will be used for explicit resolution. After loading, resave your model in Version 7.0 (R2007b) of Stateflow software.

Common Dialog Box Interface for Specifying Data Types in Stateflow Charts and Simulink Models

You can use the same dialog box interface for specifying data types in Stateflow charts and Simulink models. For more information, see [Setting Data Properties in the Data Dialog Box](#) in the Stateflow documentation.

Support for Animating Stateflow Charts in Simulink External Mode

When running Simulink models in external mode, you can now animate states, and view Stateflow test points in floating scopes and signal viewers. For more information, see *Animating Stateflow Charts* in the Stateflow documentation.

These Real-Time Workshop targets support Stateflow chart animation in external mode:

Real-Time Workshop Target	External Mode Support	Support for Stateflow Chart Animation in External Mode
GRT (generic real-time)	R10	Yes
VxWorks® / Tornado®	R10	Yes
RTWin (Real-Time Windows)	R11	Yes
xPC	R12 *	No **
ERT (embedded real-time)	R13	Yes
RSIM (rapid simulation)	R13	Yes
MPC5xx	R2007a	No
C166®	R2007a	No
TI's C6000™	R2007a	Yes
TI's C2000™	R2007b	No
Rapid Accelerator	R2007b	Yes
dSPACE® RTI	R12.1 ***	No

Note

- * xPC supported parameter download only from release R12 through R14sp3. As of release R2006a, xPC supports signal upload as well.
 - ** xPC has documented support for `xpcexplr` to display the boolean value of test point Stateflow states. You can also retrieve the state value via the xPC command-line API. There is no documented support for animating a Stateflow chart that is running in Simulink external mode.
 - *** dSPACE RTI supports parameter download only.
-

Support for Target Function Library

Stateflow Coder code generation software supports the Target Function Library published by Real-Time Workshop Embedded Coder software, allowing you to map a subset of built-in math functions and arithmetic operators to target-specific implementations. For more information, see [Replacing Operators with Target-Specific Implementations and Replacement of C Math Library Functions with Target-Specific Implementations](#) in the Stateflow documentation.

Support for Fixed-Point Parameters in Truth Table Blocks

You can now define fixed-point parameters in Truth Table blocks.

Support for Using Custom Storage Classes to Control Stateflow Data in Generated Code

You can use custom storage classes to control Stateflow local data, output data, and data store memory in Real-Time Workshop generated code.

For more information, see Custom Storage Classes in the Real-Time Workshop Embedded Coder documentation.

Loading 2007b Stateflow Charts in Earlier Versions of Simulink Software

If you save a Stateflow chart in release 2007b, you will not be able to load the corresponding model in earlier versions of Simulink software. To work around this issue, save your model in the earlier version before loading it, as follows:

- 1** In the Simulink model window, select **File > Save As**.
- 2** In the **Save as type** field, select the version in which you want to load the model.

For example, if you want to load the model in the R2007a version of Simulink software, select **Simulink 6.6/R2007a Models (#.mdl)**.

Bug Fixed for the History Junction

In previous releases, there was a bug where a default transition action occurred more than once if you used a history junction in a state containing only a single substate. The history junction did not remember the state's last active configuration unless there was more than one substate. This bug has been fixed.

R2007a+

Version: 6.6.1
New Features: No
Bug Fixes: Yes

R2007a

Version: 6.6
New Features: Yes
Bug Fixes: Yes

New Operators for Detecting Changes in Data Values

You can use three new operators for detecting changes in Stateflow data values between time steps:

- `hasChanged`
- `hasChangedFrom`
- `hasChangedTo`

For more information, see “Detecting Changes in Data Values” in the Stateflow documentation.

Elimination of "goto" Statements from Generated Code

The code generation process automatically eliminates goto statements from generated code to produce structured, readable code that better supports MISRA C rules.

R2006b

Version: 6.5
New Features: Yes
Bug Fixes: Yes

Support for Mealy and Moore Charts

You can use a new chart property to constrain finite state machines to use either Mealy or Moore semantics. You can create Stateflow charts that implement pure Mealy or Moore semantics as a subset of Stateflow chart semantics. Mealy and Moore charts can be used in simulation and code generation of C and hardware description language (HDL). See [Building Mealy and Moore Charts](#) in the Stateflow documentation.

New Structure Data Type Provides Support for Buses

You can use a structure data type to interface Simulink bus signals with Stateflow charts and truth tables, and to define local and temporary structures. You specify Stateflow structure data types as `Simulink.Bus` objects. See *Working with Structures and Bus Signals in Stateflow Charts* in the Stateflow documentation.

Note Signal logging is not available for Stateflow structures.

Custom Integer Sizes

Integers are no longer restricted in size to 8, 16, or 32 bits. You can now enter word lengths of any size from one to 32 bits.

R2006a+

Version: 6.4.1
New Features: No
Bug Fixes: No

No New Features or Changes

R2006a

Version: 6.4
New Features: Yes
Bug Fixes: No

Option to Initialize Outputs When Chart Wakes Up

You can use a new chart option **Initialize Outputs Every Time Chart Wakes Up**. Use this to initialize the value of outputs every time a chart wakes up, not only at time 0 (see [Setting Properties for a Single Chart](#) in the online documentation). When you enable this option, outputs are reset whenever the chart is triggered, whether by a function call, edge trigger, or clock tick. The option ensures that outputs are defined in every chart execution and prevents latching of outputs.

Ability to Customize the Stateflow User Interface

You can use MATLAB code to perform the following customizations of the standard Stateflow user interface:

- Add items and submenus that execute custom commands in the Stateflow Editor
- Disable or hide menu items in the Stateflow Editor

Using the MATLAB Workspace Browser for Debugging Stateflow Charts

The MATLAB Workspace Browser is no longer available for debugging Stateflow charts. To view Stateflow data values at breakpoints during simulation, use the MATLAB command line or the Browse Data window in the Stateflow Debugger.

Chart and Truth Table Blocks Require C Compiler for 64-Bit Windows Operating Systems

No C compiler ships with Stateflow software for 64-bit Windows operating systems. Because Stateflow software performs simulation through code generation, you must supply your own MEX-supported C compiler if you wish to use Stateflow Chart and Truth Table blocks. The C compilers available at the time of this writing for 64-bit Windows operating systems include the Microsoft Platform SDK and the Microsoft Visual Studio development system.

R14SP3

Version: 6.3
New Features: Yes
Bug Fixes: No

Data Handling

Sharing Global Data Between Simulink Models and Stateflow Charts

This release provides an interface that gives Stateflow charts access to global variables in Simulink models. A Simulink model implements global variables as *data stores*, created either as data store memory blocks or instances of `Simulink.Signal` objects. Now Stateflow charts can share global data with Simulink models by reading and writing data store memory symbolically using the Stateflow action language. See [Sharing Global Data with Multiple Charts](#) in the Stateflow documentation.

Enhancements to Data Properties Dialog Box

The Stateflow data properties dialog box has been enhanced to:

- Accommodate fixed-point support
- Support parameter expressions in data properties

Stateflow charts now accept Simulink parameters or parameters defined in the MATLAB workspace for the following properties in the data properties dialog box:

- Initial Value
- Minimum
- Maximum

Entries for these parameters can be expressions that meet the following requirements:

- Expressions must evaluate to scalar values.
- For library charts, the expressions for these properties must evaluate to the same value for all instances of the library chart. Otherwise, a compile-time error appears.

See [Defining Data](#) in the Stateflow documentation.

Truth Table Enhancements

Using Embedded MATLAB Action Language in Truth Tables

You can now use the Embedded MATLAB action language in Stateflow truth tables. Previously, you were restricted to the Stateflow action language. The Embedded MATLAB action language offers the following advantages:

- Supports the use of control loops and conditional constructs in truth table actions
- Provides direct access to all MATLAB functions

See Truth Table Functions for Decision-Making Logic in the Stateflow documentation.

Embedded MATLAB Truth Table Block in Simulink Models

A truth table function block is now available as an element in the Simulink library. With this new block, you can call a truth table function directly from your Simulink model. Previously, there was a level of indirection. Your Simulink model had to include a Stateflow block that called a truth table function.

The Simulink truth table block supports the Embedded MATLAB language subset only. You must have a Stateflow software license to use the Truth Table block in Simulink models.

See Truth Table Functions for Decision-Making Logic in the Stateflow documentation.

API Enhancements

Retrieving Object Handles of Selected Stateflow Objects

A new Stateflow function `sfgo` retrieves the object handles of the most recently selected objects in a Stateflow chart.

Default Case Handling in Generated Code

Stateflow Coder software now implements a default case in generated switch statements to account for corrupted memory at runtime. In this situation, the default case performs a recovery operation by calling the child entry functions of the state whose variable is out of bounds. Reentering the state resets the variable to a valid value.

This recovery operation is *not* performed if a Stateflow chart contains any of the following elements:

- Local events
- Machine-parented events
- Implicit events, such as state entry, state exit, and data change

If any of these conditions exist in a chart, state machine processing can become recursive, causing variables to temporarily assume values that are out of range. However, when processing finishes, the variables return to valid values.

Greater Usability

Specifying Execution Order of Parallel States Explicitly

You can specify the execution order of parallel states explicitly in Stateflow charts. Previously, the execution order of parallel states was governed solely by implicit rules, based on geometry. A disadvantage of implicit ordering is that it creates a dependency between design layout and execution priority. When you rearrange parallel states in your chart, you may inadvertently change order of execution and affect simulation results. Explicit ordering gives you more control over your designs. See Execution Order for Parallel States in the Stateflow documentation.

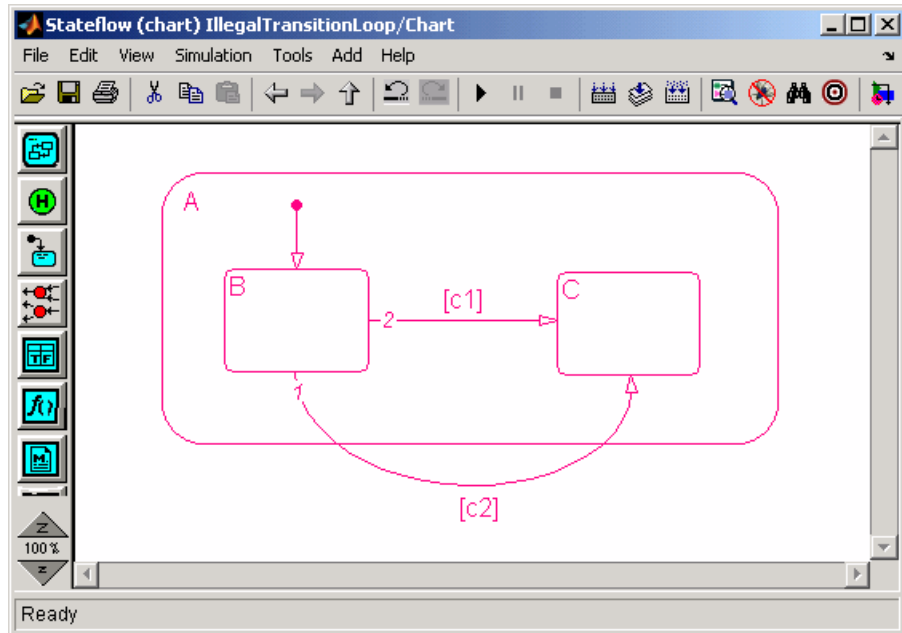
Hyperlinking Simulink Subsystems from Stateflow Events

You can now directly hyperlink the Simulink subsystem connected to a Stateflow output event by using the context menu option **Explore** for any state or transition broadcasting event. See Accessing Simulink Subsystems Triggered By Output Events in the Stateflow documentation.

Warnings for Transitions Looping Out of Logical Parent

A common modeling error is to create charts where a transition loops out of the logical parent of the source and destination objects. The *logical parent* is either a common parent of the source and destination objects, or if there is no common parent, the nearest common ancestor.

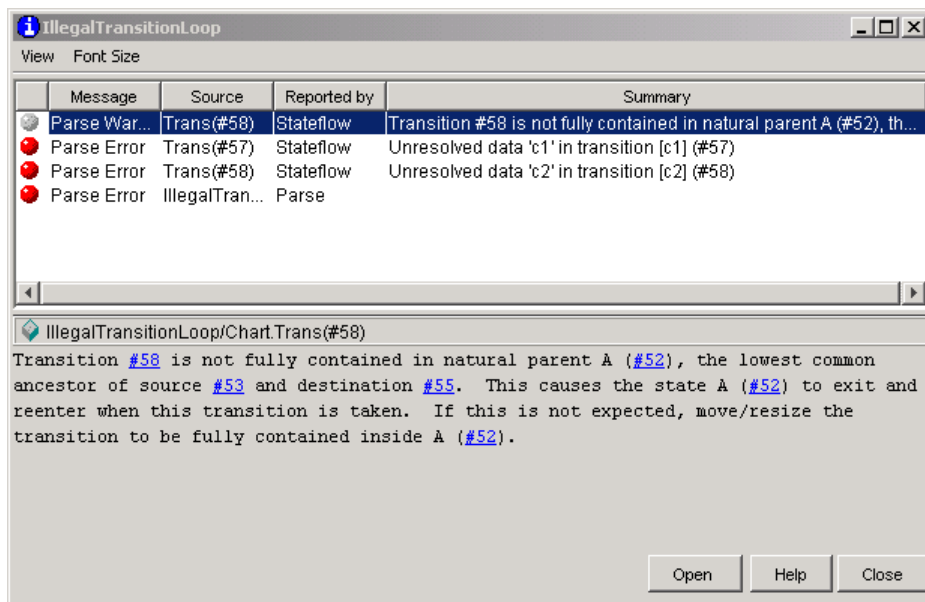
Consider the following example:



In this chart, transition 1 loops outside of logical parent A, which is the common parent of transition source B and destination C.

This type of illegal looping causes the parent to deactivate and then reactivate. In the previous example, if transition 1 is taken, the exit action of A executes and then the entry action of A executes. Executing these actions unintentionally can cause side effects.

This situation is now detected as a parser warning that indicates how to fix the model. Here is the warning associated with the earlier example:



Differentiating Syntax Elements in the Stateflow Action Language

You can now use color highlighting to differentiate syntax elements in the Stateflow action language. Syntax highlighting is enabled by default. To specify highlighting preferences, select **Highlighting Preferences** from the chart **Edit** menu, and then click the colors you want to change. See Differentiating Syntax Elements in the Stateflow Action Language in the Stateflow documentation.

Stateflow Chart Notes Click Function

This release introduces enhancements to Stateflow chart notes. The chart notes property dialog box now has a **ClickFcn** section, which includes the following options:

- Use display text as click callback check box
- ClickFcn edit field

See Annotations Properties Dialog Box in the Simulink documentation for a description of these new options.

Chart Viewing Enhancements

This release adds the following chart viewing enhancements:

- “View Command History” on page 248
- “New View Menu Viewing Commands” on page 248
- “New Shortcut Menu Commands” on page 248
- “View Command Shortcut Keys” on page 249

View Command History

This release enhances the chart viewing commands. You can now maintain a history of the chart viewing commands, i.e., pan and zoom, that you execute for each chart window. The history allows you to quickly return to a previous view in a window, using commands for traversing the history (see “New View Menu Viewing Commands” on page 248).

New View Menu Viewing Commands

This release adds the following viewing commands to the chart’s View menu:

- **View > Back**
Displays the previous view in the view history.
- **View > Forward**
Displays the next view in the view history.
- **View > Go To Parent**
Goes to the parent of the current subchart.

New Shortcut Menu Commands

The shortcut menu now has **Forward** and **Go To Parent** commands. The **Back** command has been moved to be with these new commands. These commands are the same as those described in “New View Menu Viewing Commands” on page 248.

View Command Shortcut Keys

This release adds the following viewing command shortcut keys for users running the UNIX operating system or the Windows operating system:

Shortcut Key	Command
d or Ctrl+Left Arrow	Pan left
g or Ctrl+Right Arrow	Pan right
e or Ctrl+Up Arrow	Pan up
c or Ctrl+Down Arrow	Pan down
b	Go back in pan/zoom history
t	Go forward in pan/zoom history

Note These shortcut keys, together with the existing zoom shortcuts (**r** or **+** for zoom in, **v** or **-** for zoom out), allow you to pan and zoom a model with one hand (your left hand).

R14SP2

Version: 6.2
New Features: Yes
Bug Fixes: No

User-Specified Transition Execution Order

Stateflow charts now support a mode where you can explicitly specify the testing/execution order of transitions sourced by states and junctions. This is called the Explicit mode. The Implicit mode retains the old functionality, where the transition execution order is determined based on a set of rules (parent depth, triggered/conditional properties, and geometry around the source). In addition, the transition numbers, according to their execution order, are now displayed on the Stateflow Editor at all times, both in Implicit and Explicit modes.

Note that the old models created in earlier releases load in Implicit mode, thus yielding identical simulation results. Any new charts created are in Implicit mode by default. To change to Explicit mode, use the Chart Properties dialog box.

Enhanced Integration of Stateflow Library Charts with Simulink Models

Compatibility Considerations: Yes

Charts in library models do not require full specification of data type and size. During simulation, library charts can inherit data properties from the main model in which you link them.

This enhancement also affects code generation in library charts. When building simulation and code generation targets, only the library charts that you link in the main model participate in code generation.

Compatibility Considerations

In previous releases, library charts required complete specification of data properties. You had to enter these properties for both the library chart and the main model before simulation.

Stateflow Charts and Embedded MATLAB Functions Support Simulink Data Type Aliases

Data in Stateflow charts and Embedded MATLAB functions may now be explicitly typed using the same aliased types that a Simulink model uses. Also, inherited and parameterized data types in Stateflow charts and Embedded MATLAB functions support propagation of aliased types. However, code generated for Stateflow charts and Embedded MATLAB functions does not yet preserve aliased data types.

Fixed-Point Override Supported for Library Charts

You can now specify fixed-point override for Stateflow library charts.