

# **Stateflow<sup>®</sup> and Stateflow<sup>®</sup> Coder<sup>™</sup> Release Notes**

---

## How to Contact The MathWorks



[www.mathworks.com](http://www.mathworks.com) Web  
[comp.soft-sys.matlab](mailto:comp.soft-sys.matlab) Newsgroup  
[www.mathworks.com/contact\\_TS.html](http://www.mathworks.com/contact_TS.html) Technical Support



[suggest@mathworks.com](mailto:suggest@mathworks.com) Product enhancement suggestions  
[bugs@mathworks.com](mailto:bugs@mathworks.com) Bug reports  
[doc@mathworks.com](mailto:doc@mathworks.com) Documentation error reports  
[service@mathworks.com](mailto:service@mathworks.com) Order status, license renewals, passcodes  
[info@mathworks.com](mailto: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.

*Stateflow<sup>®</sup> and Stateflow<sup>®</sup> Coder<sup>™</sup> Release Notes*

© COPYRIGHT 2000–2010 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](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

### Patents

The MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

<b>Summary by Version</b> .....	<b>1</b>
<b>Version 7.5 (R2010a) Stateflow and Stateflow<sup>®</sup> Coder Software</b> .....	<b>5</b>
<b>Version 7.4 (R2009b) Stateflow and Stateflow<sup>®</sup> Coder Software</b> .....	<b>14</b>
<b>Version 7.3 (R2009a) Stateflow and Stateflow<sup>®</sup> Coder Software</b> .....	<b>21</b>
<b>Version 7.2 (R2008b) Stateflow and Stateflow<sup>®</sup> Coder Software</b> .....	<b>27</b>
<b>Version 7.1.1 (R2008a+) Stateflow and Stateflow<sup>®</sup> Coder Software</b> .....	<b>68</b>
<b>Version 7.1 (R2008a) Stateflow and Stateflow<sup>®</sup> Coder Software</b> .....	<b>69</b>
<b>Version 7.0.1 (R2007b+) Stateflow and Stateflow<sup>®</sup> Coder Software</b> .....	<b>76</b>
<b>Version 7.0 (R2007b) Stateflow and Stateflow<sup>®</sup> Coder Software</b> .....	<b>77</b>
<b>Version 6.6.1 (R2007a+) Stateflow and Stateflow<sup>®</sup> Coder Software</b> .....	<b>83</b>
<b>Version 6.6 (R2007a) Stateflow and Stateflow<sup>®</sup> Coder Software</b> .....	<b>84</b>
<b>Version 6.5 (R2006b) Stateflow and Stateflow<sup>®</sup> Coder Software</b> .....	<b>85</b>

<b>Version 6.4.1 (R2006a+) Stateflow and Stateflow® Coder Software</b> .....	<b>86</b>
<b>Version 6.4 (R2006a) Stateflow and Stateflow® Coder Software</b> .....	<b>87</b>
<b>Version 6.3 (R14SP3) Stateflow and Stateflow® Coder Software</b> .....	<b>89</b>
<b>Version 6.2 (R14SP2) Stateflow and Stateflow® Coder Software</b> .....	<b>97</b>
<b>Version 6.1 (R14SP1) Stateflow and Stateflow® Coder Software</b> .....	<b>99</b>
<b>Version 6.0 (R14) Stateflow and Stateflow® Coder Software</b> .....	<b>100</b>
<b>Version 5.1.1 (R13SP1) Stateflow and Stateflow® Coder Software</b> .....	<b>113</b>
<b>Version 5.1 (R13+) Stateflow and Stateflow® Coder Software</b> .....	<b>115</b>
<b>Version 5.0 (R13) Stateflow and Stateflow® Coder Software</b> .....	<b>117</b>
<b>Version 4.1 (R12.1) Stateflow and Stateflow® Coder Software</b> .....	<b>123</b>
<b>Version 4.0 (R12) Stateflow and Stateflow® Coder Software</b> .....	<b>128</b>
<b>Version 3.0 (R11) Stateflow and Stateflow® Coder Software</b> .....	<b>130</b>
<b>Compatibility Summary for Stateflow and Stateflow® Coder Software</b> .....	<b>134</b>

## Summary by Version

This table provides quick access to what's new in each version. For clarification, see "Using Release Notes" on page 2.

<b>Version (Release)</b>	<b>New Features and Changes</b>	<b>Version Compatibility Considerations</b>	<b>Fixed Bugs and Known Problems</b>	<b>Related Documentation at Web Site</b>
<b>Latest version V7.5 (R2010a)</b>	Yes Details	Yes Summary	Bug Reports Includes fixes	Printable Release Notes: PDF  Current product documentation
V7.4 (R2009b)	Yes Details	Yes Summary	Bug Reports Includes fixes	No
V7.3 (R2009a)	Yes Details	Yes Summary	Bug Reports Includes fixes	No
V7.2 (R2008b)	Yes Details	Yes Summary	Bug Reports Includes fixes	No
V7.1.1 (R2008a+)	No	No	Bug Reports Includes fixes	No
V7.1 (R2008a)	Yes Details	Yes Summary	Bug Reports Includes fixes	No
V7.0.1 (R2007b+)	No	No	Bug Reports Includes fixes	No
V7.0 (R2007b)	Yes Details	Yes Summary	Bug Reports Includes fixes	No
V6.6.1 (R2007a+)	No	No	Bug Reports Includes fixes	No
V6.6 (R2007a)	Yes Details	No	Bug Reports Includes fixes	No
V6.5 (R2006b)	Yes Details	No	Bug Reports Includes fixes	No

<b>Version (Release)</b>	<b>New Features and Changes</b>	<b>Version Compatibility Considerations</b>	<b>Fixed Bugs and Known Problems</b>	<b>Related Documentation at Web Site</b>
V6.4.1 (R2006a+)	No	No	Bug Reports at Web site	No
V6.4 (R2006a)	Yes Details	No	Bug Reports at Web site	No
V6.3 (R14SP3)	Yes Details	No	Bug Reports at Web site	No
V6.2 (R14SP2)	Yes Details	No	Bug Reports at Web site	No
V6.1 (R14SP1)	Yes Details	No	Fixed Bugs	No
V6.0 (R14)	Yes Details	Yes Summary	Fixed Bugs	No
V5.1.1 (R13SP1)	Yes Details	Yes Summary	No	No
V5.1 (R13+)	Yes Details	No	Fixed Bugs	No
V5.0 (R13)	Yes Details	Yes Summary	Fixed Bugs	Printable User's Guide: PDF V5.0 product documentation
V4.1 (R12.1)	Yes Details	Yes Summary	"Fixed Bugs" on page 125	No
V4.0 (R12)	Yes Details	Yes Summary	No	No
V3.0 (R11)	Yes Details	No	No	No

## Using Release Notes

Use release notes when upgrading to a newer version to learn about:

- New features
- Changes
- Potential impact on your existing files and practices

Review the release notes for other MathWorks™ products required for this product (for example, MATLAB® or Simulink®). Determine if enhancements, bugs, or compatibility considerations in other products impact you.

If you are upgrading from a software version other than the most recent one, review the current release notes and all interim versions. For example, when you upgrade from V1.0 to V1.2, review the release notes for V1.1 and V1.2.

## What Is in the Release Notes

### New Features and Changes

- New functionality
- Changes to existing functionality

### Version Compatibility Considerations

When a new feature or change introduces a reported incompatibility between versions, the **Compatibility Considerations** subsection explains the impact.

Compatibility issues reported after the product release appear under Bug Reports at The MathWorks™ Web site. Bug fixes can sometimes result in incompatibilities, so review the fixed bugs in Bug Reports for any compatibility impact.

### Fixed Bugs and Known Problems

The MathWorks offers a user-searchable Bug Reports database so you can view Bug Reports. The development team updates this database at release time and as more information becomes available. Bug Reports include provisions for any known workarounds or file replacements. Information is available for bugs existing in or fixed in Release 14SP2 or later. Information is not available for all bugs in earlier releases.

Access Bug Reports using your MathWorks Account.



## Version 7.5 (R2010a) Stateflow and Stateflow Coder Software

This table summarizes what's new in V7.5 (R2010a):

<b>New Features and Changes</b>	<b>Version Compatibility Considerations</b>	<b>Fixed Bugs and Known Problems</b>	<b>Related Documentation at Web Site</b>
Yes Details below	Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary.	Bug Reports Includes fixes	Printable Release Notes: PDF  Current product documentation

New features and changes introduced in this version are:

- “Support for Combining Actions in State Labels” on page 6
- “New Diagnostic Detects Unused Data and Events” on page 6
- “Enhanced Support for Variable-Size Chart Inputs and Outputs” on page 7
- “Support for Chart-Level Data with Fixed-Point Word Lengths Up to 128 Bits” on page 7
- “New ‘States When Enabling’ Property for Charts with Function-Call Input Events” on page 8
- “Support for Tunable Structures of Parameter Scope in Stateflow Charts” on page 8
- “Enhanced Real-Time Workshop Code Generation for Noninlined State Functions” on page 8
- “Enhanced Real-Time Workshop Code Generation for sizeof Function” on page 9
- “Enhanced Real-Time Workshop Code Generation for Custom-Code Function Calls” on page 9
- “Data Change Implicit Event No Longer Supports Machine-Parented Data” on page 10

- “Support for Machine-Parented Events Completely Removed” on page 10
- “MEX Compilation with Microsoft® Visual Studio .NET 2003 No Longer Supported” on page 11
- “Code Generation Status Messages No Longer Shown in Command Window” on page 11
- “Change in Behavior for Appearance of Optimization Parameters” on page 12
- “Enhanced Inlining of Generated Code That Calls Subfunctions” on page 12
- “Check Box for ‘Treat as atomic unit’ Now Always Selected” on page 12
- “New Demos” on page 12

## **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® and Stateflow® Coder™ 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 and Stateflow Coder 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 and Stateflow Coder 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 and Stateflow Coder 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 Stateflow 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 and Stateflow Coder User’s Guide*.

## **Support for Tunable Structures of Parameter Scope in Stateflow Charts**

You can now define structures of parameter scope that are tunable. For more information, see “Defining Structures of Parameter Scope” in the *Stateflow and Stateflow Coder 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 and Stateflow Coder 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(&amp;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&lt;10; i++) { t[i] = y[i]; } fcn(t);</pre>	<p>Custom-code function calls use pass-by-reference, for example:</p> <pre>fcn(&amp;y[0]);</pre>	<ul style="list-style-type: none"> <li>• Ensures consistent results between simulation and code generation.</li> <li>• Less memory usage because a temporary variable is not necessary.</li> <li>• Faster execution of generated code because a memcpy operation is not necessary.</li> </ul>

## Data Change Implicit Event No Longer Supports Machine-Parented Data

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 “Using Change Detection in Actions” in the *Stateflow and Stateflow Coder User’s Guide*.

## Support for Machine-Parented Events Completely Removed

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 and Stateflow Coder 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” 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:



<b>Demo...</b>	<b>Shows how you can...</b>
<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

## Version 7.4 (R2009b) Stateflow and Stateflow Coder Software

This table summarizes what's new in V7.4 (R2009b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary.	Bug Reports Includes fixes	No

New features and changes introduced in this version are:

- “Ability to Copy Simulink Function-Call Subsystems and Paste in Stateflow Editor as Simulink Functions, and Vice Versa” on page 15
- “Ability to Generate Switch-Case Statements for Flow Graphs and Embedded MATLAB Functions Using Real-Time Workshop® Embedded Coder Software” on page 15
- “Support for Creating Switch-Case Flow Graphs Using the Pattern Wizard” on page 16
- “Support for Using More Than 254 Events in a Chart” on page 16
- “Improved Panning and Selection of States and Transitions When Using Stateflow Debugger” on page 16
- “Stateflow Compilation Status Added to Progress Indicator on Simulink Status Bar” on page 17
- “Support for Chart Inputs and Outputs That Vary in Dimension During Simulation” on page 17
- “New Compilation Report for Embedded MATLAB Functions in Stateflow Charts” on page 17
- “Enhanced Support for Replacing Math Functions with Target-Specific Implementations” on page 18

- “Enhanced Context Menu Support for Adding Flow Graph Patterns to Charts” on page 18
- “Option to Log Chart Signals Available in the Stateflow Editor” on page 19
- “Default Precision Set to Double for Calls to C Math Functions” on page 19
- “Change in Text and Visibility of Parameter Prompt for Easier Use with Fixed-Point Advisor and Fixed-Point Tool” on page 19
- “Charts Closed By Default When Opening Models Saved in Formats of Earlier Versions” on page 20
- “New and Enhanced Demos” on page 20

## **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 and Stateflow Coder 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 Embedded MATLAB Functions”
- “Real-Time Workshop 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 and Stateflow Coder 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 and Stateflow Coder 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 Debugging Window” in the *Stateflow and Stateflow Coder 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**

Stateflow 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 Embedded MATLAB Functions” in the *Stateflow and Stateflow Coder User’s Guide*.

## **New Compilation Report for Embedded MATLAB Functions in Stateflow Charts**

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 Compilation 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:

- “Example: Mapping Math Functions to Target-Specific Implementations”
- “Replacement of C Math Library Functions with Target-Specific Implementations”
- “Replacing Operators with Target Functions”

## Enhanced Context Menu Support for Adding Flow Graph Patterns to Charts

In the Stateflow Editor, you can now right-click at any level of the chart 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 “Logging Chart Signals Using the Signal Logging Dialog Box” in the *Stateflow and Stateflow Coder 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 and Stateflow Coder 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 and Stateflow Coder 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



## Version 7.3 (R2009a) Stateflow and Stateflow Coder Software

This table summarizes what's new in V7.3 (R2009a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary.	Bug Reports Includes fixes	No

New features and changes introduced in this version are:

- “Support for Saving the Complete Simulation State at a Specific Time” on page 22
- “Enhanced Support for Enumerated Data Types” on page 22
- “New Boolean Keywords in Stateflow Action Language” on page 22
- “Enhanced Control of Inlining State Functions in Generated Code” on page 23
- “New Diagnostic to Detect Unintended Backtracking Behavior in Flow Graphs” on page 23
- “Use of Basic Linear Algebra Subprograms (BLAS) Libraries for Speed” on page 23
- “Enhanced Support for Replacing C Math Functions with Target-Specific Implementations” on page 23
- “Smart Transitions Now Prefer Straight Lines” on page 24
- “Clicking Up-Arrow Button in the Stateflow Editor Closes Top-Level Chart” on page 24
- “Enhanced Type Resolution for Symbols” on page 24
- “Enhanced Code Generation for Stateflow Events” on page 25

- “Enhanced Real-Time Workshop Generated Code for Charts with Simulink Functions” on page 25
- “Use of en, du, ex, entry, during, and exit for Data and Event Names Being Disallowed in a Future Version” on page 25
- “Support for Machine-Parented Events Being Removed in a Future Version” on page 25

## **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 Using SimState” in the *Stateflow and Stateflow Coder 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 and Stateflow Coder 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 “Symbols in Action Language” in the *Stateflow and Stateflow Coder 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 and Stateflow Coder 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 and Stateflow Coder 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:

- “Example: Mapping Math Functions to Target-Specific Implementations”

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

## 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 and Stateflow Coder 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 and Stateflow Coder 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 Stateflow 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**

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**

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.

## Version 7.2 (R2008b) Stateflow and Stateflow Coder Software

This table summarizes what's new in V7.2 (R2008b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary.	Bug Reports Includes fixes	No

New features and changes introduced in this version are:

- “Support for Embedding Simulink Function-Call Subsystems in a Stateflow Chart” on page 28
- “Support for Using Enumerated Data Types in a Stateflow Chart” on page 28
- “New Alignment, Distribution, and Resizing Commands for Stateflow Charts” on page 28
- “Unified Simulation and Embeddable Code Generation Options for Stateflow Charts and Truth Table Blocks” on page 29
- “New Pattern Wizard for Consistent Creation of Logic Patterns and Iterative Loops” on page 63
- “Support for Initializing Vectors and Matrices in the Data Properties Dialog Box” on page 64
- “Change in Default Mode for Ordering Parallel States and Outgoing Transitions” on page 64
- “Optimized Inlining of Code Generated for Stateflow Charts” on page 64
- “More Efficient Parsing for Nonlibrary Models” on page 64
- “Change in Casting Behavior When Calling MATLAB Functions in a Chart” on page 64

- “Use of Output Data with Change Detection Operators Disallowed for Initialize-Outputs-at-Wakeup Mode” on page 65
- “Parsing a Stateflow Chart Without Simulation No Longer Detects Unresolved Symbol Errors” on page 66
- “Generation of a Unique Name for a Copied State Limited to States Without Default Labels” on page 66
- “New Configuration Set Created When Loading Nonlibrary Models with an Active Configuration Reference” on page 66

## **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 and Stateflow Coder 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 and Stateflow Coder User’s Guide* and “Using Enumerated Data” in the *Simulink User’s Guide*. For information on how enumerated data types appear in Real-Time Workshop generated code, see “Enumerated Data Type Considerations” in the *Real-Time Workshop 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 and Stateflow Coder User’s Guide*.

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

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 29	Enhanced with new options in the <b>Real-Time Workshop</b> pane of the Configuration Parameters dialog box  See “GUI Enhancements in Real-Time Workshop Code Generation Options for Nonlibrary Models” on page 40
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 35	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 44

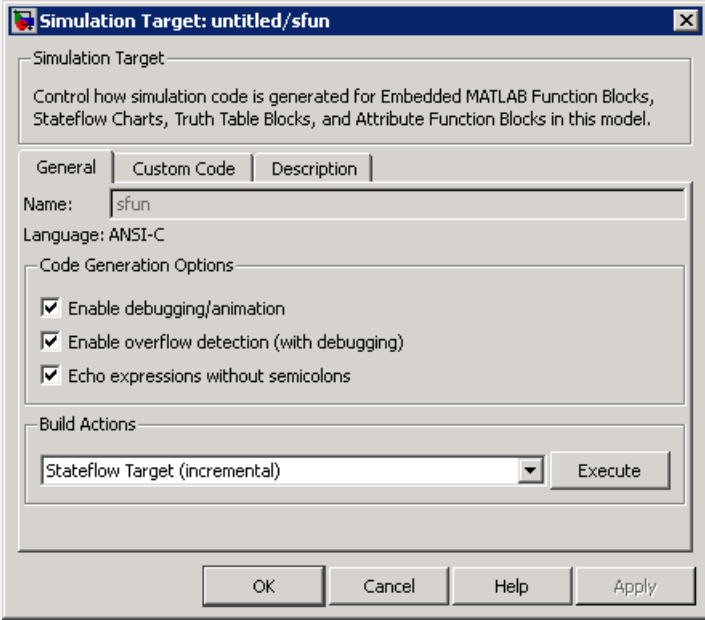
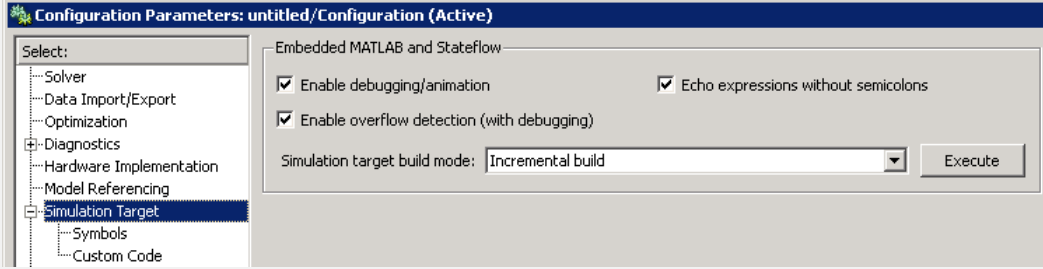
For more information, see “Configuration Parameters Dialog Box” in the *Simulink Graphical User Interface* and “Building Targets” in the *Stateflow and Stateflow Coder User’s Guide*.

For compatibility information, see “Compatibility Considerations” on page 58.

### GUI Changes in Simulation Options for Nonlibrary Models

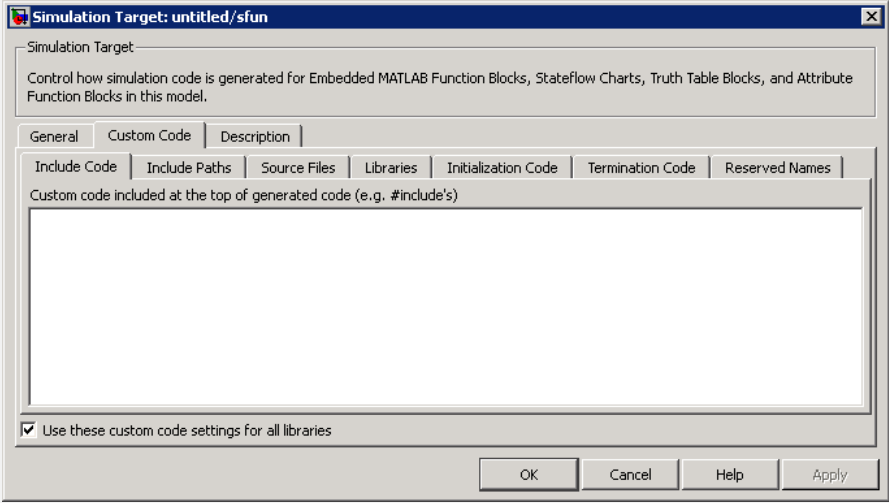
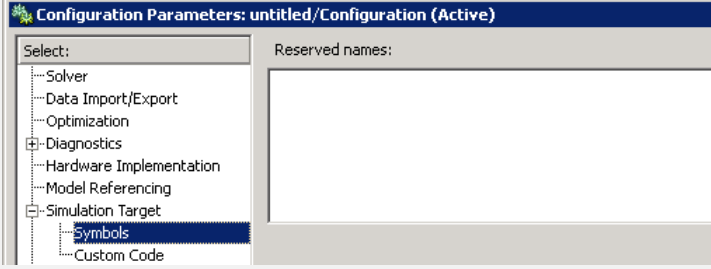
The following sections describe changes in the panes of the Simulation Target dialog box for nonlibrary models.

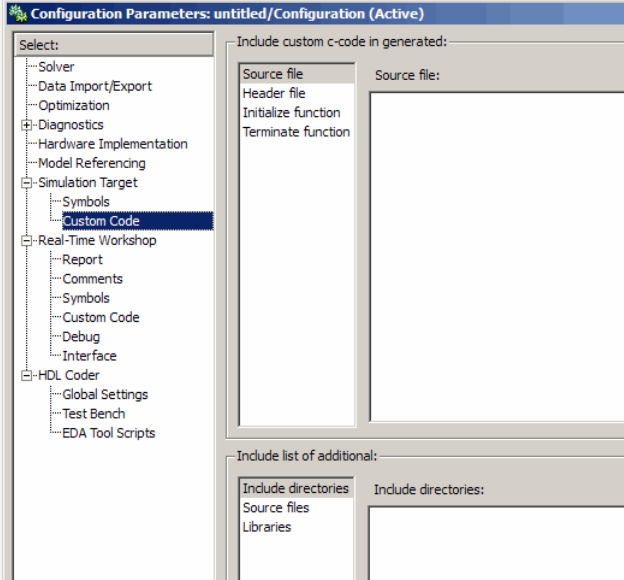
### Changes for the General Pane of the Simulation Target Dialog Box.

Release	Appearance
Previous	<p data-bbox="276 361 902 390"><b>General</b> pane of the Simulation Target dialog box</p> 
New	<p data-bbox="276 1062 1139 1093"><b>Simulation Target</b> 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 33.

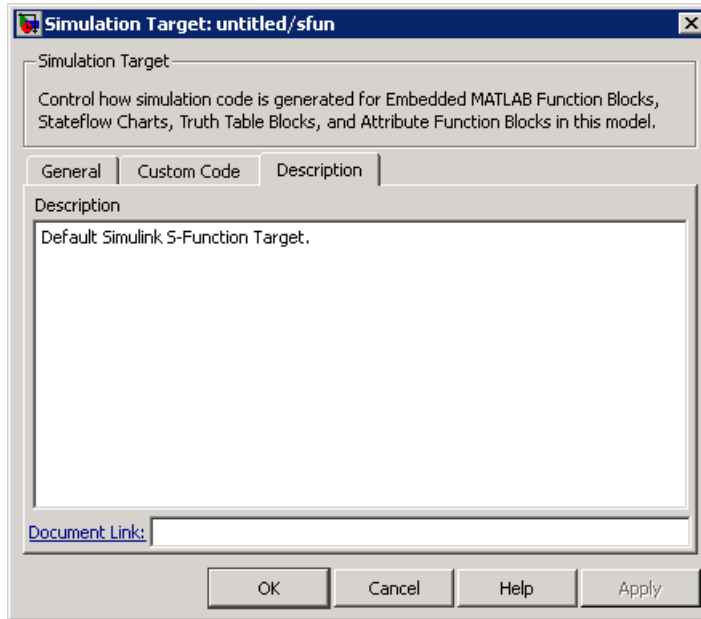
### Changes for the Custom Code Pane of the Simulation Target Dialog Box.

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

Release	Appearance
New	<p data-bbox="276 302 1307 361"><b>Simulation Target &gt; Custom Code</b> 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 33.

**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.

<b>Old Option in the Simulation Target Dialog Box</b>	<b>New Option in the Configuration Parameters Dialog Box</b>	<b>Default Value of New Option</b>
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	{}

<b>Old Option in the Simulation Target Dialog Box</b>	<b>New Option in the Configuration Parameters Dialog Box</b>	<b>Default Value of New Option</b>
Custom Code > Use these custom code settings for all libraries	None	Not applicable
Description > Description	<p data-bbox="543 475 609 501">None</p> <hr data-bbox="543 562 921 565"/> <p data-bbox="543 571 914 951"><b>Note</b> If you load an older model that contained user-specified text in the <b>Description</b> field, that text now appears in the Model Explorer. When you select <b>Simulink Root &gt; Configuration Preferences</b> in the <b>Model Hierarchy</b> pane, the text appears in the <b>Description</b> field for that model.</p> <hr data-bbox="543 961 921 965"/>	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.

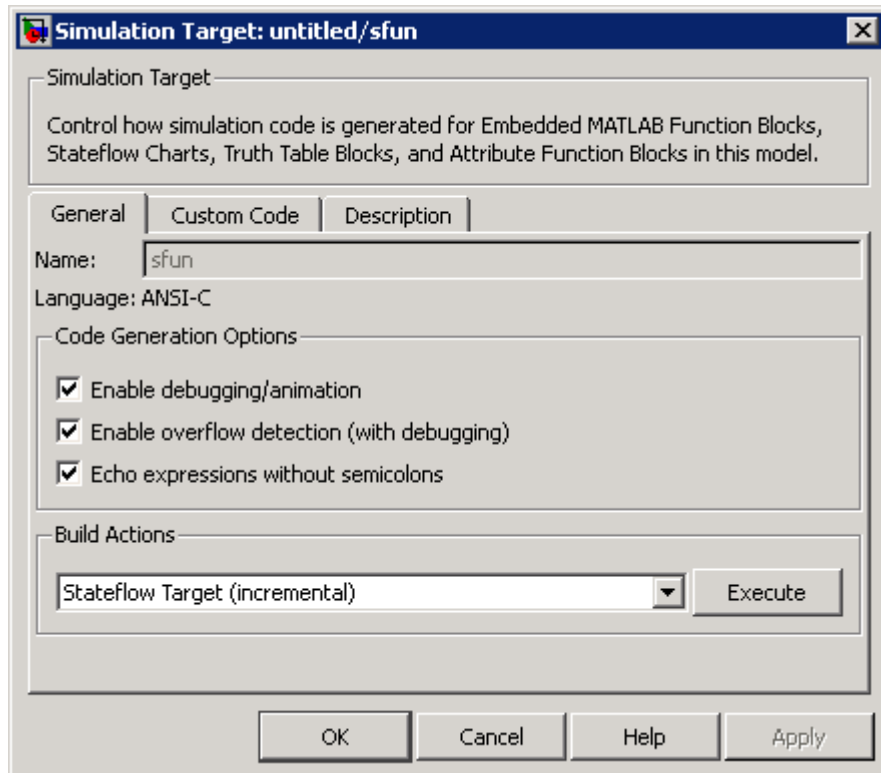
---

### 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.**

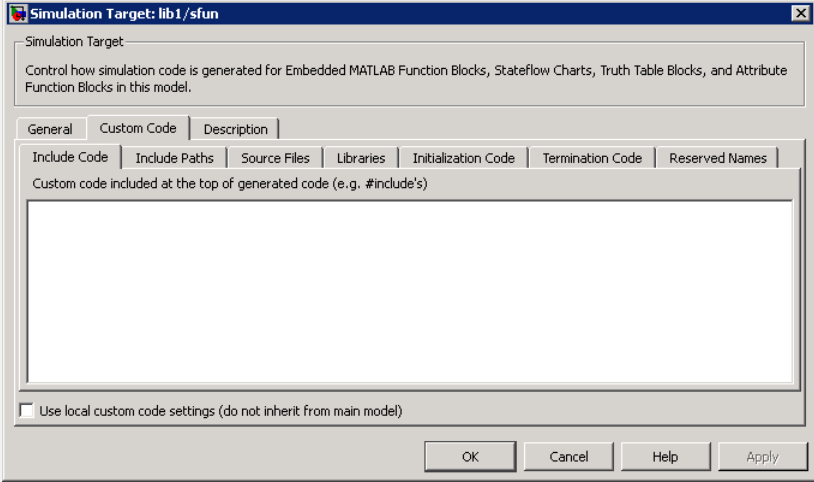
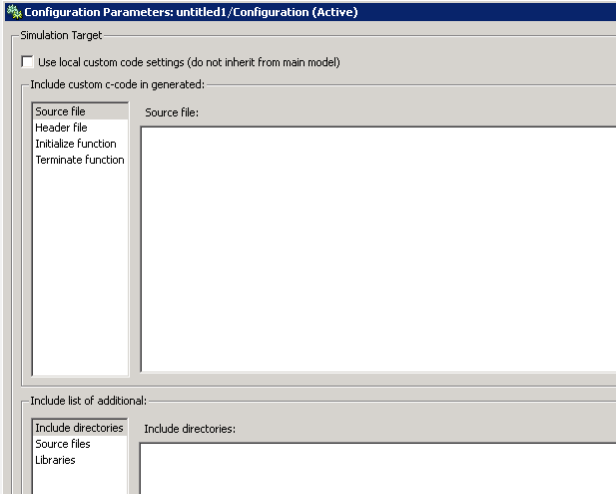
In previous releases, the **General** pane of the Simulation Target dialog box for library models appeared as follows.



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.

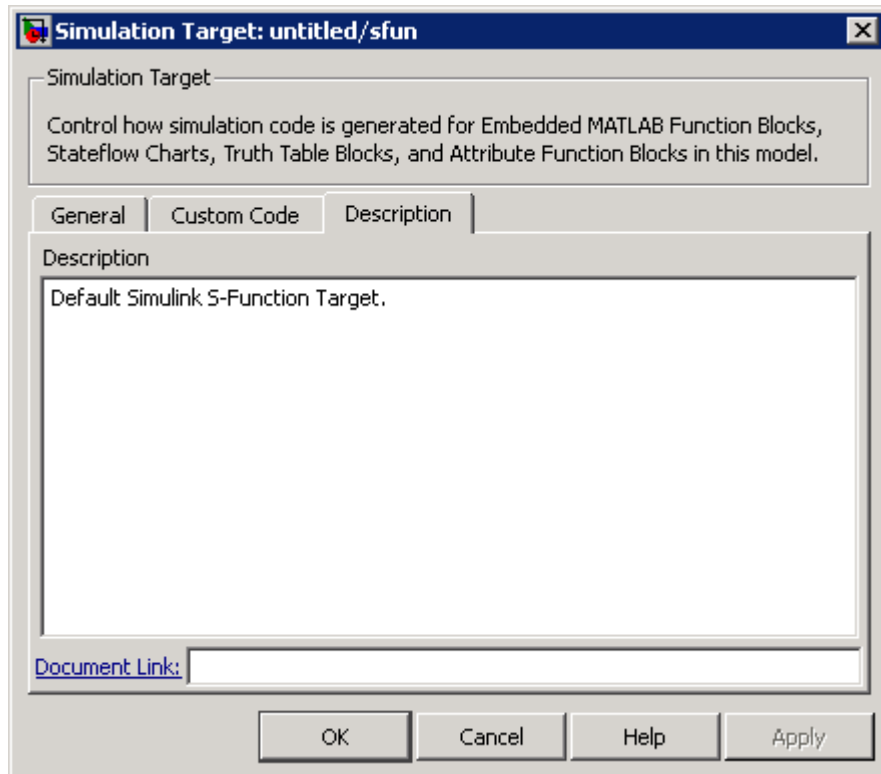


## Changes for the Custom Code Pane of the Simulation Target Dialog Box.

Release	Appearance
Previous	<p><b>Custom Code</b> pane of the Simulation Target dialog box</p> 
New	<p><b>Simulation Target</b> 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 39.

**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.

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

<b>Old Option in the Simulation Target Dialog Box</b>	<b>New Option in the Configuration Parameters Dialog Box</b>	<b>Default Value of New Option</b>
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

---

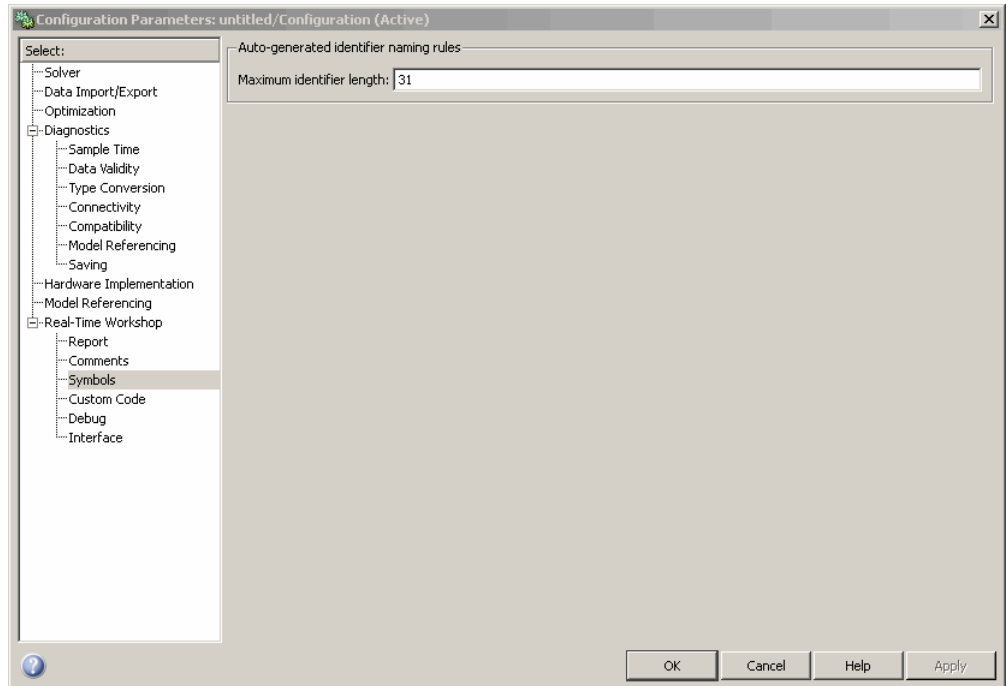
**Note** For library models, **Simulation Target** options in the Configuration Parameters dialog box are not available in the Model Explorer.

---

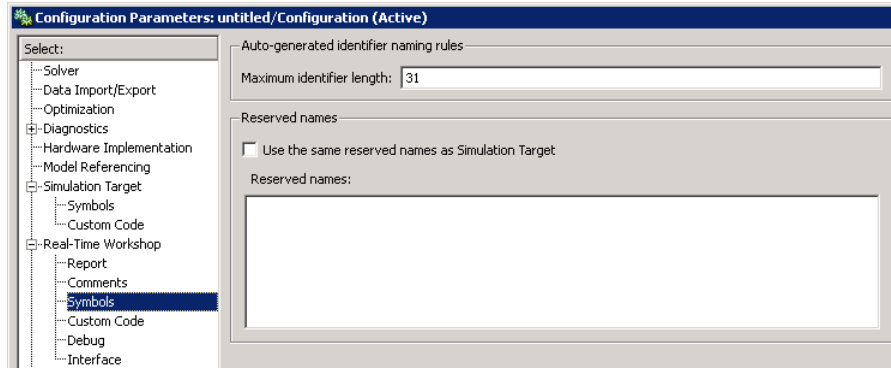
### **GUI Enhancements in Real-Time Workshop Code Generation Options for Nonlibrary Models**

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.



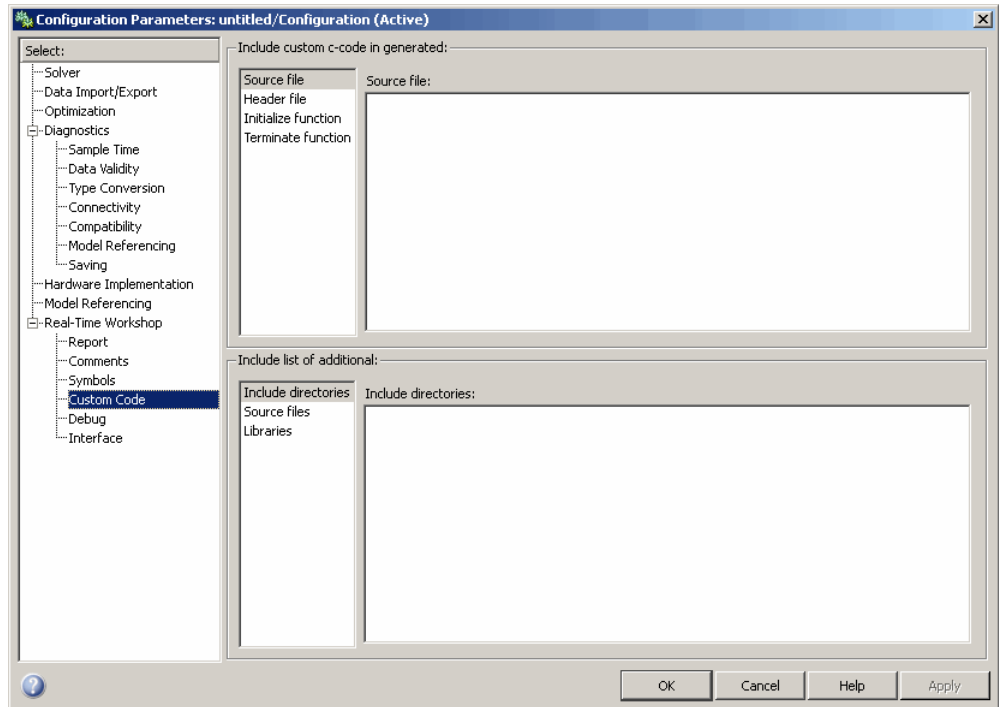
In R2008b, a new option is available in this pane: **Reserved names**. You can use this option to specify a set of keywords that the Real-Time Workshop build process should not use. This action prevents naming conflicts between functions and variables from external environments and identifiers in the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



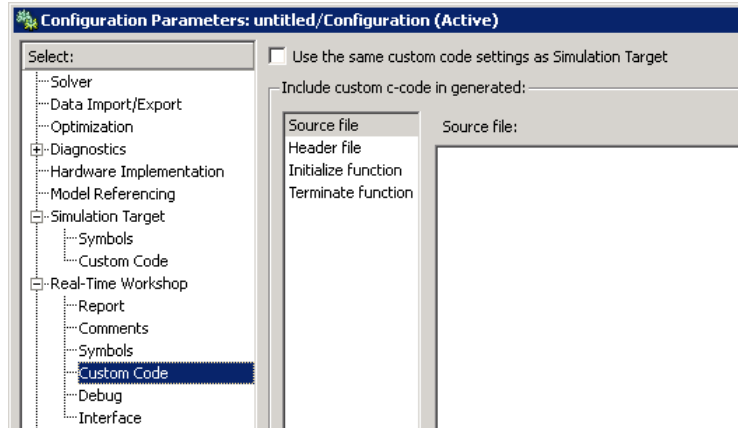
You can also choose to use the reserved names specified in the **Simulation Target > Symbols** pane to avoid entering the same information twice for the nonlibrary model. Select the **Use the same reserved names as Simulation Target** check box.

For more information, see “Reserved names” in the Real-Time Workshop Reference.

**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.



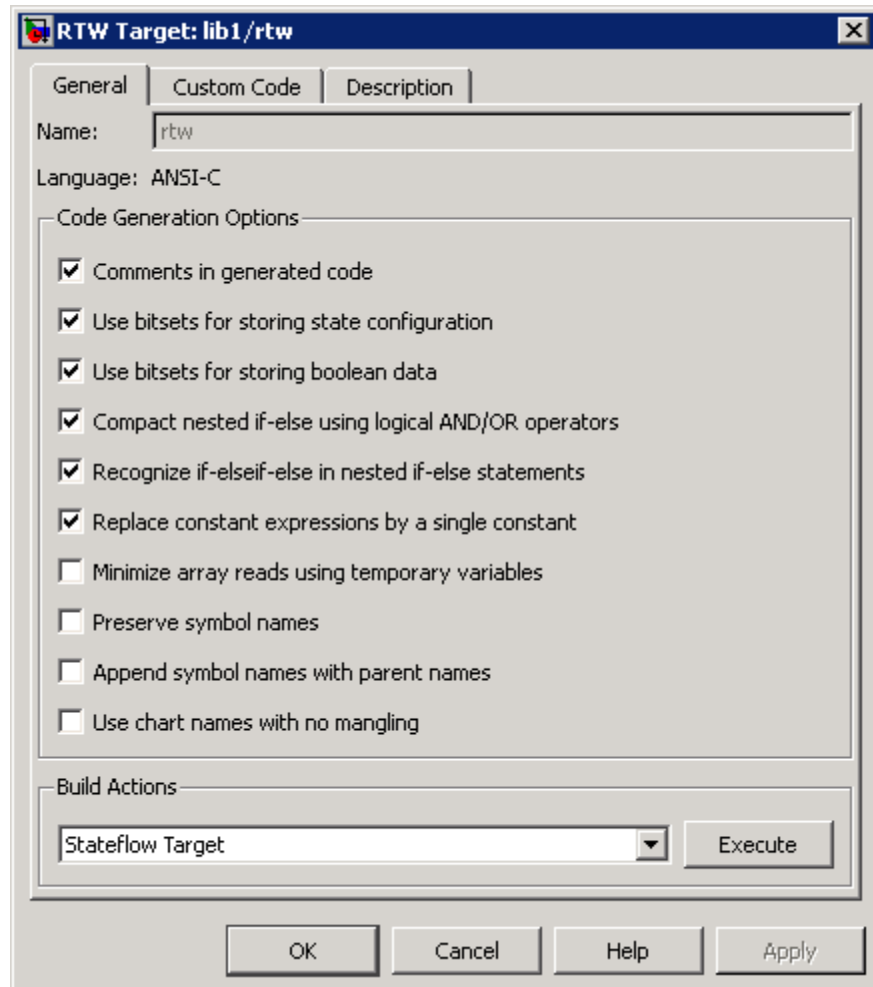
For more information, see “Use the same custom code settings as Simulation Target” in the Real-Time Workshop Reference.

## 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.

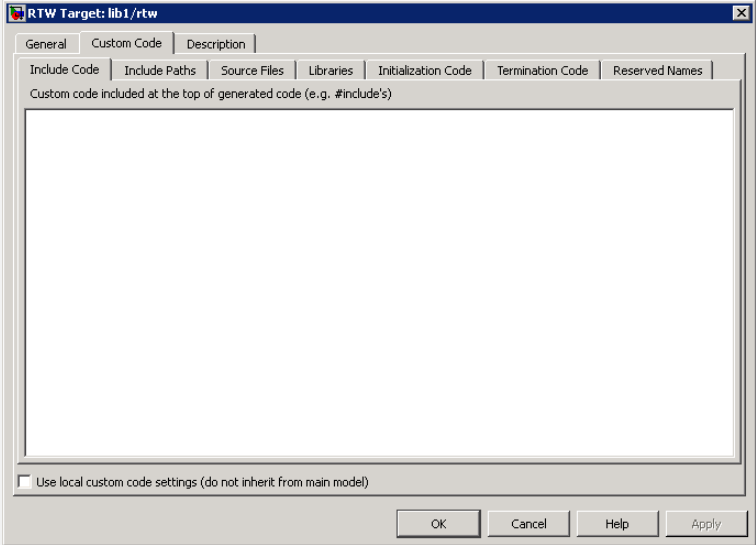


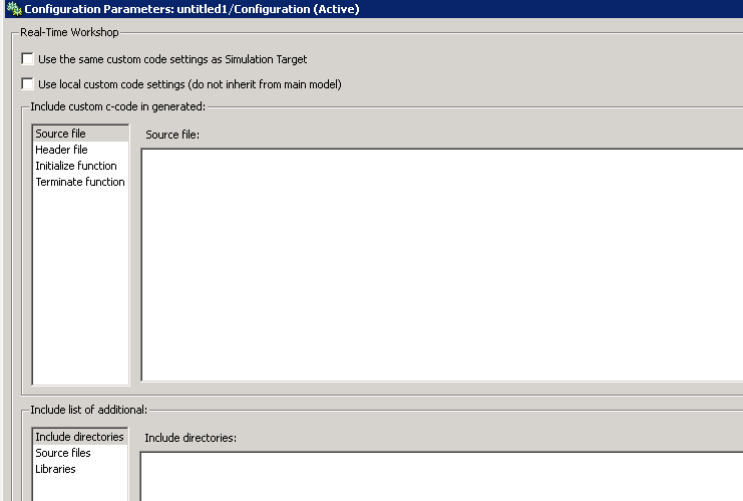
**Changes for the General Pane of the RTW Target Dialog Box.** In previous releases, the **General** pane of the RTW Target dialog box for library models appeared as follows.



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.

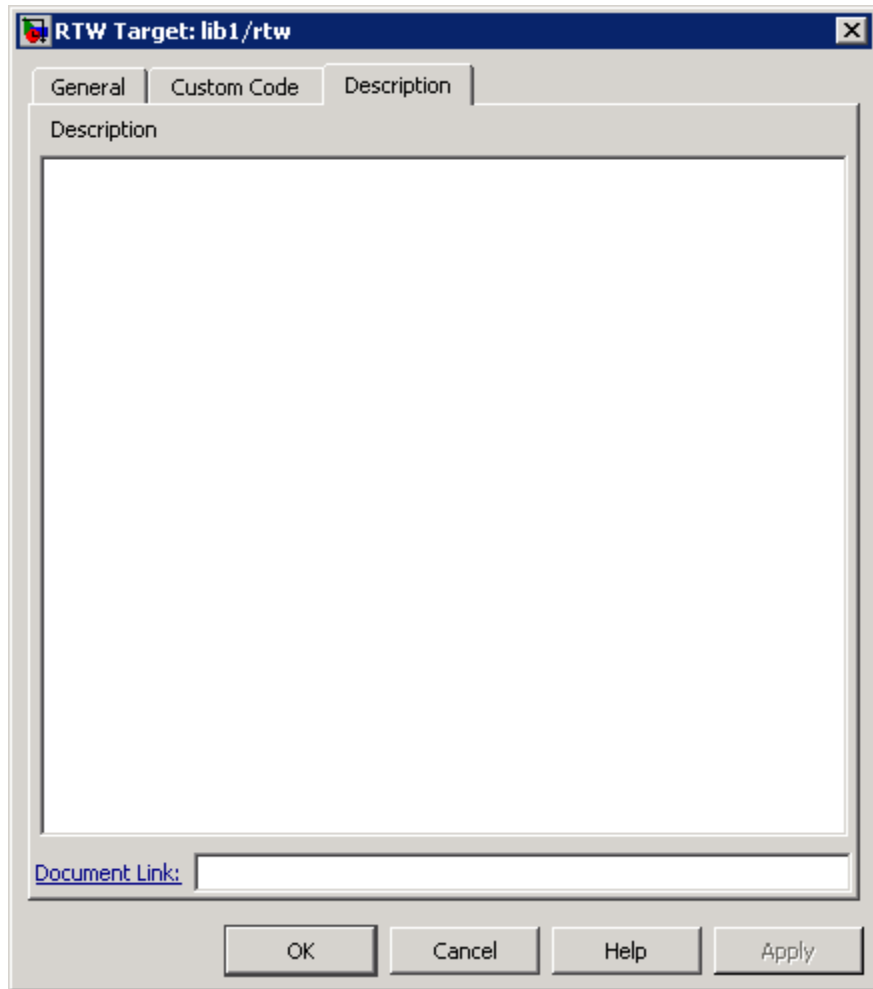
Release	Appearance
Previous	<p data-bbox="278 361 902 390"><b>Custom Code</b> pane of the RTW Target dialog box</p> 

Release	Appearance
New	<p data-bbox="278 305 1176 331"><b>Real-Time Workshop</b> pane of the Configuration Parameters dialog box</p>  <p data-bbox="278 366 638 387">% Configuration Parameters: untitled1 / Configuration (Active)</p> <p data-bbox="293 392 397 406">Real-Time Workshop:</p> <p data-bbox="293 418 584 434"><input type="checkbox"/> Use the same custom code settings as Simulation Target</p> <p data-bbox="293 444 617 460"><input type="checkbox"/> Use local custom code settings (do not inherit from main model)</p> <p data-bbox="293 470 480 486">Include custom c-code in generated:</p> <p data-bbox="308 496 362 512">Source file</p> <p data-bbox="308 513 362 529">Header file</p> <p data-bbox="308 531 406 546">Initialize Function</p> <p data-bbox="308 548 406 564">Terminate Function</p> <p data-bbox="421 496 480 512">Source file:</p> <p data-bbox="293 774 421 789">Include list of additional:</p> <p data-bbox="308 800 406 815">Include directories:</p> <p data-bbox="308 817 362 833">Source files</p> <p data-bbox="308 835 362 850">Libraries</p> <p data-bbox="421 800 510 815">Include directories:</p>

For details, see “Library Models: Mapping of GUI Options from the RTW Target Dialog Box to the Configuration Parameters Dialog Box” on page 49.

**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.

<b>Old Option in the RTW Target Dialog Box</b>	<b>New Option in the Configuration Parameters Dialog Box</b>	<b>Default Value of New Option</b>
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
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

<b>Old Option in the RTW Target Dialog Box</b>	<b>New Option in the Configuration Parameters Dialog Box</b>	<b>Default Value of New Option</b>
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
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 “Updating Scripts That Set Options Programmatically for Simulation and Embeddable Code Generation” on page 58 and “What Happens When You Load an Older Model in R2008b” on page 59.

**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 <code>sfun</code> 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	SFSimOverflowDetection string - off, on	Simulation Target > Enable overflow

<b>Old sfun Object Property</b>	<b>Old Option in the Simulation Target Dialog Box</b>	<b>New Configuration Parameter</b>	<b>New Option in the Configuration Parameters Dialog Box</b>
	(with debugging)		detection (with debugging)
CodeFlagsInfo ('echo')	<b>General &gt; Echo expressions without semicolons</b>	SFSimEcho string - off, <b>on</b>	<b>Simulation Target &gt; Echo expressions without semicolons</b>
CustomCode	<b>Custom Code &gt; Include Code</b>	SimCustomHeaderCode string -	<b>Simulation Target &gt; Custom Code &gt; Header file</b>
CustomInitializer	<b>Custom Code &gt; Initialization Code</b>	SimCustomInitializer string -	<b>Simulation Target &gt; Custom Code &gt; Initialize function</b>
CustomTerminator	<b>Custom Code &gt; Termination Code</b>	SimCustomTerminator string -	<b>Simulation Target &gt; Custom Code &gt; Terminate function</b>
ReservedNames	<b>Custom Code &gt; Reserved Names</b>	SimReservedNameArray string array - {}	<b>Simulation Target &gt; Symbols &gt; Reserved names</b>



<b>Old sfun Object Property</b>	<b>Old Option in the Simulation Target Dialog Box</b>	<b>New Configuration Parameter</b>	<b>New Option in the Configuration Parameters Dialog Box</b>
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.

<b>Old sfun Object Property</b>	<b>Old Option in the Simulation Target Dialog Box</b>	<b>New Configuration Parameter</b>	<b>New Option in the Configuration Parameters Dialog Box</b>
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

<b>Old sfun Object Property</b>	<b>Old Option in the Simulation Target Dialog Box</b>	<b>New Configuration Parameter</b>	<b>New Option in the Configuration Parameters Dialog Box</b>
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> - <b>off</b> , 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.

<b>Old rtw Object Property</b>	<b>Old Option in the RTW Target Dialog Box</b>	<b>New Configuration Parameter</b>	<b>New Option in the Configuration Parameters Dialog Box</b>
CustomCode	Custom Code > Include Code	CustomHeaderCode <i>string</i> -	Real-Time Workshop > Header file
CustomInitializer	Custom Code > Initialization Code	CustomInitializer <i>string</i> -	Real-Time Workshop > Initialize function
CustomTerminator	Custom Code > Termination Code	CustomTerminator <i>string</i> -	Real-Time Workshop > Terminate function
UseLocalCustomCodeSettings	Custom Code > Use local custom code settings (do not inherit from main model)	RTWUseLocalCustomCode <i>string</i> - <b>off</b> , on	Real-Time Workshop > Use local custom code settings (do not inherit from main model)
UserIncludeDirs	Custom Code > Include Paths	CustomInclude <i>string</i> -	Real-Time Workshop > Include directories
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 string – <b>sf_incremental_build</b> , sf_nonincremental_build, sf_make, sf_make_clean, sf_make_clean_objects	<b>Simulation Target &gt; Simulation target build mode</b>	Specifies how you build the simulation target for a model.
SimCustomSourceCode <i>string</i> -	<b>Simulation Target &gt; Custom Code &gt; Source file</b>	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> -	<b>Simulation Target &gt; Source file</b>	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.

<b>New Configuration Parameter</b>	<b>New Option in the Configuration Parameters Dialog Box</b>	<b>Description</b>
ReservedNameArray <i>string array - {}</i>	<b>Real-Time Workshop &gt; Symbols &gt; Reserved names</b>	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 – off, on</i>	<b>Real-Time Workshop &gt; Custom Code &gt; Use the same custom code settings as Simulation Target</b>	Specify whether to use the same custom code settings as those specified for simulation.
UseSimReservedNames <i>string – off, on</i>	<b>Real-Time Workshop &gt; Symbols &gt; Use the same reserved names as Simulation Target</b>	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.

<b>New Configuration Parameter</b>	<b>New Option in the Configuration Parameters Dialog Box</b>	<b>Description</b>
CustomSourceCode <i>string –</i>	<b>Real-Time Workshop &gt; Source file</b>	Enter code lines to appear near the top of a generated source code file.
RTWUseSimCustomCode <i>string – off, on</i>	<b>Real-Time Workshop &gt; Use the same custom code settings as Simulation Target</b>	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 51</p> <hr/> <p><b>Note</b> 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 “What Happens When You Load an Older Model in R2008b” on page 59.</p> <hr/>
Using the <code>set_param</code> and <code>get_param</code> commands	<p>“Using the Command-Line API to Set Parameters for Simulation and Embeddable Code Generation” in the <i>Stateflow and Stateflow Coder User’s Guide</i></p>

### 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.

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

<b>Option in the Simulation Target Dialog Box of a Library Model</b>	<b>Equivalent Object Property</b>
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.

<b>Option in the RTW Target Dialog Box of a Library Model</b>	<b>Equivalent Object Property</b>
<b>General &gt; Comments in generated code</b>	CodeFlagsInfo('comments')
<b>General &gt; Use bitsets for storing state configuration</b>	CodeFlagsInfo('statebitsets')
<b>General &gt; Use bitsets for storing boolean data</b>	CodeFlagsInfo('databitsets')
<b>General &gt; Compact nested if-else using logical AND/OR operators</b>	CodeFlagsInfo('emitlogicalops')
<b>General &gt; Recognize if-elseif-else in nested if-else statements</b>	CodeFlagsInfo('elseifdetection')
<b>General &gt; Replace constant expressions by a single constant</b>	CodeFlagsInfo('constantfolding')
<b>General &gt; Minimize array reads using temporary variables</b>	CodeFlagsInfo('redundantloadelimination')
<b>General &gt; Preserve symbol names</b>	CodeFlagsInfo('preservenames')
<b>General &gt; Append symbol names with parent names</b>	CodeFlagsInfo('preservenameswithparent')
<b>General &gt; Use chart names with no mangling</b>	CodeFlagsInfo('exportcharts')
<b>General &gt; Build Actions</b>	None
<b>Custom Code &gt; Reserved Names</b>	ReservedNames
<b>Description &gt; Description</b>	Description
<b>Description &gt; Document Link</b>	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 “What Happens When You Load an Older Model in R2008b” on page 59). 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:

<b>If <code>ApplyToAllLibs</code> for the main model is...</b>	<b>And <code>UseLocalCustomCodeSettings</code> for the library model is...</b>	<b>Then the library model uses...</b>
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” in the *Stateflow and Stateflow Coder User’s Guide*.

## **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” in the *Stateflow and Stateflow Coder User’s Guide*.

## **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” in the *Stateflow and Stateflow Coder User’s Guide*.

## **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**

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 `m1` operator to call a built-in or custom MATLAB function.

(For details, see “ml Namespace Operator” in the *Stateflow and Stateflow Coder User’s Guide*.)

### 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))`.

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

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 “Using Change Detection in Actions” in the *Stateflow and Stateflow Coder User’s Guide*.)

### 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 “Resolving Symbols” in the *Stateflow and Stateflow Coder User’s Guide*.

## **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” in the *Stateflow and Stateflow Coder User’s Guide*.

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

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 “Referencing Configuration Sets” in the *Simulink User’s Guide*.

## **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” in the *Simulink User’s Guide*.

## Version 7.1.1 (R2008a+) Stateflow and Stateflow Coder Software

This table summarizes what's new in V7.1.1 (R2008a+):

<b>New Features and Changes</b>	<b>Version Compatibility Considerations</b>	<b>Fixed Bugs and Known Problems</b>	<b>Related Documentation at Web Site</b>
No	No	Bug Reports Includes fixes	No



## Version 7.1 (R2008a) Stateflow and Stateflow Coder Software

This table summarizes what's new in V7.1 (R2008a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary.	Bug Reports Includes fixes	No

New features and changes introduced in this version are:

- “Support for Data with Complex Data Types” on page 70
- “Support for Functions with Multiple Outputs” on page 70
- “Bidirectional Traceability for Navigating Between Generated Code and Stateflow Objects” on page 70
- “New Temporal Logic Notation for Defining Absolute Time Periods” on page 71
- “New temporalCount Operator for Counting Occurrences of Events” on page 71
- “Using a Specific Path to a State for the in Operator” on page 71
- “Enhanced MISRA C Code Generation Support” on page 72
- “Enhanced Folder Structure for Generated Code” on page 72
- “Code Optimization for Simulink Blocks and Stateflow Charts” on page 72
- “New fitToView Method for Zooming Objects in the Stateflow Editor” on page 73
- “Generation of a Unique Name for a Copied State” on page 73
- “New Font Size Options in the Stateflow Editor” on page 73

- “New Fixed-Point Details Display in the Data Properties Dialog Box” on page 73
- ““What’s This?” Context-Sensitive Help Available for Simulink Configuration Parameters Dialog” on page 73
- “Specifying Scaling Explicitly for Fixed-Point Data” on page 74
- “Use of Data Store Memory Data in Entry Actions and Default Transitions Disallowed for Execute-at-Initialization Mode” on page 75
- “Enhanced Warning Message for Target Hardware that Does Not Support the Data Type in a Chart” on page 75
- “Detection of Division-By-Zero Violations When Debugger Is Off” on page 75

## **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” in the Stateflow and Stateflow Coder User’s Guide.

## **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 “Using Graphical Functions to Extend Actions”, “Truth Table Functions”, and “Using Embedded MATLAB Functions in Stateflow Charts” in the Stateflow and Stateflow Coder User’s Guide.

## **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 Real-Time Workshop Generated Code” in the Stateflow and Stateflow Coder User’s Guide.

## **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 and Stateflow Coder User’s Guide.

## **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 and Stateflow Coder User’s Guide.

## **Using a Specific Path to a State for the in Operator**

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 and Stateflow Coder User’s Guide.

### **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 and Stateflow Coder User’s Guide.

### **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 object method `fitToView` to zoom in on graphical objects in the Stateflow Editor.

For more information, see “Zooming a Chart Object Using the Stateflow API” in the Stateflow and Stateflow Coder User’s Guide.

## **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 and Stateflow Coder User’s Guide.

## **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 the Stateflow Editor” in the Stateflow and Stateflow Coder User’s Guide.

## **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 and Stateflow Coder User’s Guide.

## **“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

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 and Stateflow Coder User's Guide.

## **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.

## Version 7.0.1 (R2007b+) Stateflow and Stateflow Coder Software

This table summarizes what's new in V7.0.1 (R2007b+):

<b>New Features and Changes</b>	<b>Version Compatibility Considerations</b>	<b>Fixed Bugs and Known Problems</b>	<b>Related Documentation at Web Site</b>
No	No	Bug Reports Includes fixes	No



## Version 7.0 (R2007b) Stateflow and Stateflow Coder Software

This table summarizes what's new in V7.0 (R2007b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary.	Bug Reports Includes fixes	No

New features and changes introduced in this version are:

- “Enhanced Continuous-Time Support with Zero-Crossing Detection” on page 78
- “New Super Step Feature for Modeling Asynchronous Semantics” on page 78
- “Support for Inheriting Data Properties from Simulink Signal Objects Via Explicit Resolution” on page 78
- “Common Dialog Box Interface for Specifying Data Types in Stateflow Charts and Simulink Models” on page 79
- “Support for Animating Stateflow Charts in Simulink External Mode” on page 80
- “Support for Target Function Library” on page 81
- “Support for Fixed-Point Parameters in Truth Table Blocks” on page 81
- “Support for Using Custom Storage Classes to Control Stateflow Data in Generated Code” on page 81
- “Loading 2007b Stateflow Charts in Earlier Versions of Simulink Software” on page 82
- “Bug Fixed for the History Junction” on page 82

## Enhanced Continuous-Time Support with Zero-Crossing Detection

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

- 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” in the Stateflow and Stateflow Coder User’s Guide documentation.

## 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 and Stateflow Coder User’s Guide.

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

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 and Stateflow Coder User’s Guide.

### **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 and Stateflow Coder User’s Guide.

## 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 and Stateflow Coder User’s Guide.

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 Functions” and “Replacement of C Math Library Functions with Target-Specific Implementations” in the Stateflow and Stateflow Coder User’s Guide.

**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 “Creating and Using Custom Storage Classes” in the Real-Time Workshop Embedded Coder User’s Guide.

## 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.

## Version 6.6.1 (R2007a+) Stateflow and Stateflow Coder Software

This table summarizes what's new in V6.6.1 (R2007a+):

<b>New Features and Changes</b>	<b>Version Compatibility Considerations</b>	<b>Fixed Bugs and Known Problems</b>	<b>Related Documentation at Web Site</b>
No	No	Bug Reports Includes fixes	No

## Version 6.6 (R2007a) Stateflow and Stateflow Coder Software

This table summarizes what's new in Version 6.6 (R2007a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports Includes fixes	No

New features and changes introduced in this version are described here:

### 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 “Using Change Detection in Actions” in the Stateflow and Stateflow Coder User’s Guide.

### 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.



## Version 6.5 (R2006b) Stateflow and Stateflow Coder Software

This table summarizes what's new in Version 6.5 (R2006b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports Includes fixes	No

New features and changes introduced in this version are described here:

### 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 and Stateflow Coder User’s Guide.

### 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 and Stateflow Coder User’s Guide.

---

**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.

## Version 6.4.1 (R2006a+) Stateflow and Stateflow Coder Software

This table summarizes what's new in V6.4.1 (R2006a+):

<b>New Features and Changes</b>	<b>Version Compatibility Considerations</b>	<b>Fixed Bugs and Known Problems</b>	<b>Related Documentation at Web Site</b>
No	No	Bug Reports at Web site	No

## Version 6.4 (R2006a) Stateflow and Stateflow Coder Software

This table summarizes what's new in V6.4 (R2006a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports at Web site	No

New features and changes introduced in this version are described here:

### 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

See “Customizing the Stateflow Editor” in the online documentation.

## **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.

## Version 6.3 (R14SP3) Stateflow and Stateflow Coder Software

This table summarizes what's new in V6.3 (R14SP3):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports at Web site	No

New features and changes introduced in this version are organized by these topics:

- “Data Handling” on page 89
- “Truth Table Enhancements” on page 90
- “API Enhancements” on page 91
- “Greater Usability” on page 92

### 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 Simulink Models” in the Stateflow and Stateflow Coder User’s Guide 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 and Stateflow Coder User’s Guide.

## 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” in the Stateflow and Stateflow Coder User’s Guide.

## 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” in the Stateflow and Stateflow Coder User’s Guide.

## API Enhancements

### Retrieving Object Handles of Selected Stateflow Objects

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

### Default Case Handling in Generated Code

Stateflow Coder code generation 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 and Stateflow Coder User’s Guide.

### Hyperlinking Simulink Subsystems from Stateflow Events

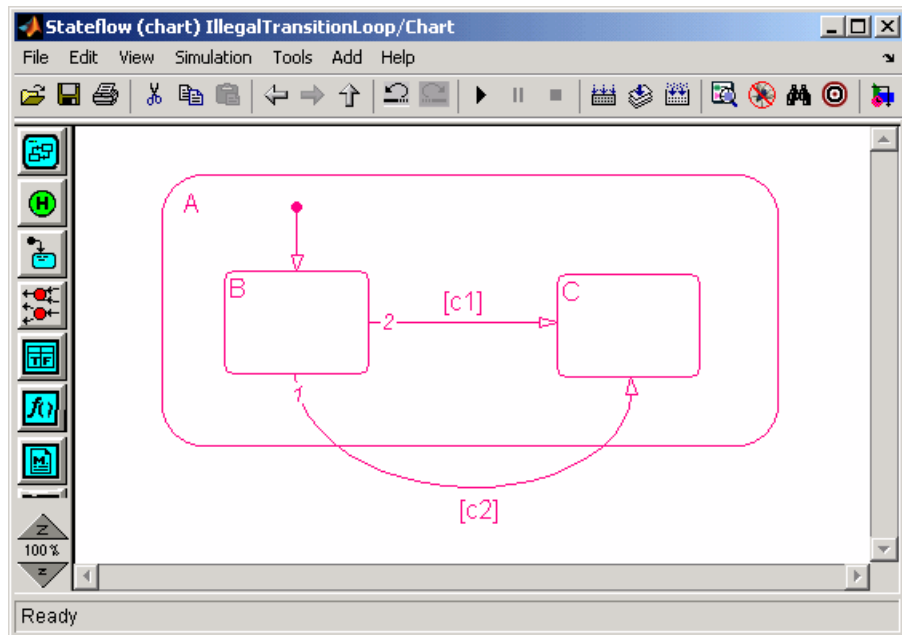
You can now directly hyperlink the Simulink subsystem connected to a Stateflow output event by using the context menu option **Explore** for any state or transition broadcasting event. See “Accessing Simulink Subsystems Triggered By Output Events” in the Stateflow and Stateflow Coder User’s Guide.

### 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.



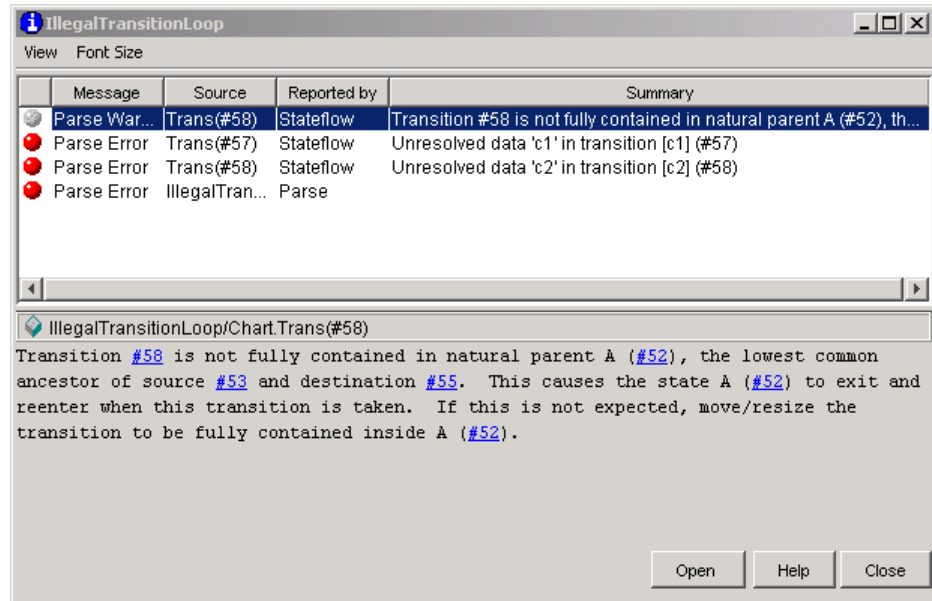
Consider the following example:



In this Stateflow 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 Stateflow chart **Edit** menu, and then click the colors you want to change. See “Differentiating Syntax Elements in the Stateflow Action Language” in the Stateflow and Stateflow Coder User’s Guide.

## 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 software documentation for a description of these new options.

## Chart Viewing Enhancements

This release adds the following chart viewing enhancements:

- “View Command History” on page 95
- “New View Menu Viewing Commands” on page 95
- “New Shortcut Menu Commands” on page 95
- “View Command Shortcut Keys” on page 95

**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 95).

**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 95.

**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:

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

---

## Version 6.2 (R14SP2) Stateflow and Stateflow Coder Software

This table summarizes what's new in V 6.2 (R14SP2):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports at Web site	No

New features introduced in this version are described here:

### 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

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.

## Version 6.1 (R14SP1) Stateflow and Stateflow Coder Software

This table summarizes what's new in V6.1 (R14SP1):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Fixed Bugs	No

New features and changes introduced in this version are described here:

### Code Generation For Embedded Targets Fails for Charts with No Output Data

In this release, embedded code generation sometimes fails when the model contains one or more Stateflow charts with no output data, and one or more testpointed local data objects or states. The error message displayed in the MATLAB Command Window has the following signature:

```
Error: File: d:\R14\matlab\rtw\c\tlc\mw\capi.tlc Line: 359
Column: 44
Undefined identifier constString
Error: File: d:\R14\matlab\rtw\c\tlc\mw\capi.tlc Line: 359
Column: 62
The + operator only works on numeric arguments
```

To work around this problem, create a dummy (unused) output data object for the charts that have no output data.

## Version 6.0 (R14) Stateflow and Stateflow Coder Software

This table summarizes what's new in V6.0 (R14):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary.	Fixed Bugs  See also “Additional Bug Fixes” on page 111	No

New features and changes introduced in this version are:

- “Code Generation” on page 100
- “Data Handling” on page 104
- “API Enhancements” on page 106
- “Enhanced Support for Functions” on page 107
- “Greater Usability” on page 108
- “Additional Bug Fixes” on page 111

### Code Generation

#### Multibyte Comment Characters in Generated Code

You can optionally include Stateflow action language comments in generated code for an embedded target or a custom target. When you do this, multibyte characters in Stateflow action language comments are preserved as multibyte characters in the generated code. This means that you can have comments in code with characters from non-English alphabets such as Japanese Kanji or Chinese Hanzi characters.

See “Comment Symbols, %, //, /\*” in the Stateflow and Stateflow Coder User’s Guide for more information on inserting comments into Stateflow action language and optionally into Stateflow generated code.



## Target Configuration Integrated with Simulink Targets

Configuring simulation and code generation targets for the Stateflow blocks in nonlibrary models has been integrated with Simulink target configuration in the Configuration Parameters dialog box. For library models, you still configure targets for Stateflow blocks as you did for previous versions of Stateflow software. This also applies to the configuration of Stateflow custom targets.

For more information on configuring targets, see “Building Targets” in the Stateflow and Stateflow Coder User’s Guide.

**Compatibility Considerations.** As a result of this change, properties of the old Stateflow code generation target now map to the Simulink target configuration set. Note the following compatibility considerations:

- You must update existing API scripts that modify properties of Stateflow code generation targets.
- Custom code settings are no longer copied from the simulation target to the code generation target for nonlibrary models.

## Updating Scripts that Modify Code on Stateflow Code Generation Targets.

If you have existing MATLAB scripts that use the Stateflow programmatic API to set up custom-code settings on the Stateflow code generation target, you must modify these scripts to use the Simulink target configuration set. The following example illustrates how the properties on the old code generation target object map to the new configuration set object.

Here is a sample old script modifying custom code properties of a code generation target (named `rtw`) in Release 13 and earlier:

```
rt = sfroot;
trgt_sfmachine = rt.find('-isa', 'Stateflow.Machine', '-and',
'Name', bdroot);
rtw_target = trgt_sfmachine.find('-isa', 'Stateflow.Target',
'-and', 'Name', 'rtw');
rtw_target.set('UserSources',foo.c');
rtw_target.set('CustomCode','#include "myhdr.h"');
rtw_target.set('UserIncludeDirs','./myincludedir');
rtw_target.set('UserLibraries','mylib.lib');
```

```
rtw_target.set('CustomInitializer','call_my_init_fcn()');  
rtw_target.set('CustomTerminator','call_my_term_fcn()');
```

Use the following format for scripts in Release 14 and beyond. Modify all appropriate scripts to use this updated convention.

```
configset = getActiveConfigSet(bdroot);  
set_param(configset,'CustomSource','foo.c');  
set_param(configset,'CustomHeaderCode','#include "myhdr.h"');  
set_param(configset,'CustomInclude','./myincludedir');  
set_param(configset,'CustomLibrary','mylib.lib');  
set_param(configset,'CustomInitializer','call_my_init_fcn()');  
set_param(configset,'CustomTerminator','call_my_term_fcn()');
```

### **Copying Code Settings from the Simulation Target to the Code Generation Target.**

In prior versions of Stateflow software, creating a code generation target copied custom code settings from the simulation target to the new code generation target. This is no longer true for nonlibrary models, which configure a code generation target in the Configuration Parameters dialog box. However, for library models, you can create a code generation target that still copies the custom code settings of the Stateflow simulation target.

### **Custom C++ Code Included with Stateflow Generated Code**

With the proper modifications, you can now include custom C++ code with Stateflow generated code. See “Building Targets” in the Stateflow and Stateflow Coder User’s Guide.

### **Object Descriptions in Generated Code for Embedded Targets**

The object descriptions entered in the property dialogs for charts, states, transitions, and graphical functions are now optionally propagated into generated code for embedded targets with a Real-Time Workshop Embedded Coder software license. You can enable this option in the Configuration Parameters dialog box as follows:

- 1** In the **Select** pane, select the **Real-Time Workshop** node.
- 2** Under the **Real-Time Workshop** node, select the **Comments** node.

**3** In the right properties pane, select **Stateflow object descriptions**.

### **Unified Code Generation Settings for Simulink Models and Stateflow Charts**

All Stateflow code generation settings for Real-Time Workshop code generation are now consolidated into the new Configuration Parameters dialog box. This enables users to specify code generation settings for Simulink models and Stateflow charts in a single location.

In addition, custom code settings (code inclusion, user source files, and so on) are now supported by the Configuration Parameters dialog box, although they can still be entered on the Real-Time Workshop target object. All custom code settings and Stateflow optimization settings from old models are grandfathered into the new Configuration Parameters dialog box at load time.

### **Faster Real-Time Workshop Code Generation**

Code generation consists of two phases. During the first phase, a TLC file is generated from every chart. These TLC files are converted into C code in the second phase. In R13SP1 and prior versions, Stateflow software regenerated these TLC files for charts even when they had not changed from a previous code generation session. In Release 14, the first phase of TLC generation is incremental so that the TLC file for a chart is regenerated only if there is a change in the chart since the last code generation session. This incremental TLC generation significantly improves the speed of Real-Time Workshop code generation for models with a large number of Stateflow charts.

### **Unified Identifier Naming Scheme in Generated Code**

The names of identifiers (variable names, structure names, field names in structures, function names) in the generated code are now uniformly managed for a common look and feel, and to prevent naming conflicts.

## Data Handling

### Data Type and Size Inheritance from Simulink Block Connections

A Stateflow chart inherits the type and size for input and output data from the signals that connect to them in a Simulink model. This feature minimizes the double data entry that was previously required for defining Stateflow input and output data. To inherit input or output data type from their block connections in a Simulink model, enter *Inherited* for their *Type* property. To inherit input or output data size from their block connections in a Simulink model, enter *-1* for their *Size* property.

See “Inheriting Data Types from Simulink Objects” and “Inheriting Input and Output Data Size from Simulink Signals” in the Stateflow and Stateflow Coder User’s Guide for more detailed descriptions.

### Data Sized by Expression

You can use expressions in the *Size* property for Stateflow data. Expressions can include numeric constants, arithmetic operators, Stateflow data of scope *Parameter* or *Constant*, and calls to the MATLAB functions *min*, *max*, and *size*. See “Sizing Stateflow Data” in the Stateflow and Stateflow Coder User’s Guide for more details.

### Parameter Scope for Simulink Parameters in Stateflow Charts

You can specify Stateflow constant read-only data that is initialized from the MATLAB workspace by specifying data with the scope *Parameter*. The MATLAB workspace can include Simulink parameters for masked subsystems that contain the Stateflow block or variables in the MATLAB base workspace. See “Initializing Data from the MATLAB Base Workspace” in the Stateflow and Stateflow Coder User’s Guide for more information.

In previous versions, the goal was to create data of type *Constant* and enable the *Initialize From Workspace* property. This action initialized the constant with the value of a variable of the same name in the MATLAB workspace, in this case, a Simulink parameter. This method had several limitations:

- 1 The parameters had to be scalars.

- 2 The parameters were not tunable.
- 3 The Stateflow parameter name resolution did not honor the Simulink name resolution. If a parameter could not be resolved in the immediate mask workspace containing the chart, an error message appeared.
- 4 Library charts that contain the parameters could not participate in code reuse.

All of the preceding limitations are now fixed. Furthermore, all existing models that have constant data initialized from the workspace are automatically redefined with the new Parameter scope.

**Compatibility Considerations.** Existing API scripts looking for Constant data initialized from the workspace must be changed to look for data with the scope Parameter, as shown in the following example:

Before R14:

```
d0 = sfm.find('-isa','Stateflow.Data','-and',
'Scope','CONSTANT_DATA','-and','InitFromWorkspace',1);
```

After R14:

```
d0 =
sfm.find('-isa','Stateflow.Data','-and','Scope','Parameter');
```

## Defining Temporary Data for Charts

You can no longer define temporary data at the chart level. Instead, an optimization in Stateflow generated code converts chart parented local data acting as temporary data to temporary data.

**Compatibility Considerations.** In existing models, temporary data defined for a chart is automatically converted to local data.

## Fixed-Point Autoscaling of Stateflow Data

Stateflow fixed-point data now participates in autoscaling for fixed-point numbers in Simulink models. For directions on how to autoscale fixed-point

data in Simulink models, see the Simulink® Fixed Point™ software documentation.

**Compatibility Considerations.** The Simulink model override of fixed-point numbers applies in Stateflow charts, with the following exceptions:

- The fixed-point override does not apply for scaled doubles. If you specify this option for an override, an error results.
- The fixed-point override is not supported for Stateflow library charts. Specifying fixed-point override for library charts results in a warning.

### Support for Directional Vectors

In Version 5.1.1 (R13SP1) and prior versions, data defined with the size of [1,3] was automatically converted to a nondirectional vector of size 3. In Version 6.0 (R14), the directionality is preserved.

**Compatibility Considerations.** If the vector data you define is a chart input or output, it is converted to a directional row vector to interface with the Simulink model. This data needs to be accessed as a two-dimensional matrix in Stateflow action language in state and transition labels, for example, as `x[0][1]`.

## API Enhancements

### Type Casting with `cast` and `type`

You can now use the cast operator `cast` for all Stateflow data types except `m1` and fixed-point data. To make casting easier, you can also use a `type` operator that provides the type of an existing data you can use as an argument in cast operations.

The `cast` and `type` operators are documented in the Stateflow and Stateflow Coder User's Guide.

### Writable Dirty Property for API Chart Object

The Dirty property for a Chart object in the Stateflow API is writable. The Dirty property for the Machine object is still read-only (RO).

## Enhanced Support for Functions

### Embedded MATLAB Functions

You can now use a special subset of the MATLAB programming language in the form of Embedded MATLAB functions. These functions give you the power of MATLAB data, function, and language features that are specially tailored to the tight memory and operating system requirements found in embedded target environments.

Embedded MATLAB functions provide:

- An editing environment for entering Embedded MATLAB function code
- A subset of the MATLAB programming language that you can use in Embedded MATLAB functions. See “Using the Embedded MATLAB Function Block” in the Simulink software documentation.
- Embedded MATLAB functions are callable from anywhere on the chart
- Embedded MATLAB functions can call other Embedded MATLAB functions, including truth tables and graphical functions
- Embedded MATLAB functions have access to all chart data (inputs, outputs, locals, and so on)

The new Embedded MATLAB Function block in the User Defined Functions library for a Simulink model uses the same underlying infrastructure as a Stateflow chart for simulation (through code generation), debugging, and integration with Real-Time Workshop code generation software. This dependency has the following ramifications:

- 1** The simulation settings for the Stateflow simulation target (sfun) apply equally to all Stateflow charts and Embedded MATLAB blocks in the model. For example, turning debugging on/off would globally affect the charts as well as Embedded MATLAB Function blocks.
- 2** Some of the compile-time warnings and run-time errors from the Embedded MATLAB Function block mention the Stateflow chart as the source of the warning or error.

- 3** Optimization settings for an embedded target apply equally to Embedded MATLAB Function blocks. These optimization settings appear for the **Optimization** node in the Configuration Parameters dialog box only when a Stateflow block or an Embedded MATLAB Function block is present in the model.
- 4** Selecting the **Rebuild All** button in the Embedded MATLAB Editor or in the Stateflow Editor regenerates the simulation code for all the Stateflow charts and all Embedded MATLAB Function blocks in the model.
- 5** In the Model Explorer, the icon for the library links of Embedded MATLAB Function blocks is identical to the icon used for Stateflow link charts.
- 6** In the Model Explorer, it is possible to copy data objects between Stateflow charts and Embedded MATLAB Function blocks.

---

**Note** An Embedded MATLAB Function block transparently shares Stateflow block infrastructure but does not require a Stateflow software license.

---

See “Using Embedded MATLAB Functions in Stateflow Charts” in the Stateflow and Stateflow Coder User’s Guide.

## **Matrix Inputs and Outputs for Graphical Functions**

Input and output data sizes for Stateflow graphical functions can now be vectors or two-dimensional matrices. The semantics of the input vector or matrix data are compliant with the MATLAB language semantics *pass-by-value* instead of *pass-by-reference* for the C language.

## **Greater Usability**

### **Data and State Activity Logging**

You can log the values of Stateflow data and the activity of Stateflow states against sampling time during simulation. After simulation, you access the data with a set of object-oriented commands. See “Logging Chart Signals



Using the Signal Logging Dialog Box” in the Stateflow and Stateflow Coder User’s Guide for a detailed example.

### **Data and State Monitoring with a Floating Scope**

You can now use Simulink floating scope blocks to monitor Stateflow data during simulation. Stateflow data and states now appear in the hierarchy of the Signal Selector dialog box of a floating scope. Selecting them causes them to be displayed in the floating scope at simulation time.

See “Using a Floating Scope to Monitor Data Values and State Activity” in the Stateflow and Stateflow Coder User’s Guide for an example.

### **Command Line Debugger in the MATLAB Command Window**

The Stateflow Debugger Browse Data option is limited to displaying all the elements of a large matrix. The Stateflow Debugger now supports a command-line mode at the MATLAB command prompt. Now when a Stateflow breakpoint is reached, you can type simple expressions at the Stateflow command-line debugger in the MATLAB Command Window to display their value. For example, the following entries compute and display values for parts of the matrix in the MATLAB Command Window:

```
x(2, :)
x(3,4)+3
```

For details on using the Command Line Debugger, see the Stateflow and Stateflow Coder User’s Guide.

### **Stateflow Boxes Parent Data**

Box objects in Stateflow charts can now parent data. This is useful for encapsulating the design better by limiting the visibility of certain data to a given Box object and its hierarchy. For example, a Box object that parents data along with graphical functions that manipulate this data can be thought of as an implementation of a singleton class with data and methods.

### **Stateflow Objects Added to the Model Explorer**

The Model Explorer now lists Stateflow objects in its hierarchy of Simulink model objects. In place of the old Stateflow Explorer tool, you can now use

the Model Explorer to add and modify data and events to Stateflow objects. For more information on using the Model Explorer with Stateflow objects, see the Stateflow and Stateflow Coder User's Guide.

## Warnings for Missing and Conditional Default Paths

A common modeling error in Stateflow charts is to create states with one of the following:

### 1 No default transitions

This can cause certain runtime state-inconsistency errors. For example, if a state has substates with no default transition, the state can be active, but no substates can.

### 2 Default transitions that are all conditional

In this case, if all default transition conditions evaluate to false at runtime, no state can become active.

Code generation during simulation detects both of these cases, and a warning to fix the model appears.

## Automatic Upgrade of Stateflow Machines Developed in Previous Versions

When you open models containing Stateflow charts that were developed in earlier versions, Stateflow software automatically upgrades the Stateflow machine to work in the current version. A warning message no longer appears, requiring that you save the model in the current version of Stateflow software.

## Trailing 1's Removed After Second Dimension of Array Size

Sizes specified for the third dimension or higher of an array that result in a trailing set of 1's are removed. Here are some examples.

Size Specified by User	Size Specified by Stateflow Software
[2, 3, 1]	[2, 3]
[3, 5, 1, 1, 1]	[3, 5]

Size Specified by User	Size Specified by Stateflow Software
[3, 4, 1, 1, 2]	No change
[4, 1]	No change

**Compatibility Considerations.** For size specifications that use trailing 1's after the second dimension, you must change array indexing in state and transition actions accordingly. For example, if you specify the size of the array `my_array` as [4, 5, 1, 1, 1], you can no longer specify the element as `my_array[2][3][0][0][0]` in an action. Instead, specify the element as `my_array[2][3]`.

## Additional Bug Fixes

The following bugs have also been fixed in V6.0. These bug fixes do not appear in Fixed Bugs on the MathWorks Web site.

### Multi-Instanced Stateflow Library Charts Generated Incorrect Code

If a Stateflow library chart was instanced more than once in a model, and if this chart had multiple function-call output events, sometimes the generated code did not compile.

### Reenabling a Link to a Subsystem Copied from a Library Corrupted a Model

Reenabling a disabled link to a subsystem copied from a library corrupted the model containing the linked subsystem. This always occurred if the library contained one or more link Stateflow charts or if a copy-and-paste (rather than a drag-and-drop) operation was used to create the library link, that is, if the subsystem was first copied from the library to the system clipboard (by typing **Ctrl+C** or selecting **Copy** from the **Simulink Edit** menu).

### **Clear Commands During Simulation Caused a Segmentation Violation**

Executing the MATLAB command `clear all` or `clear mex` during simulation of a model with a Stateflow chart caused a segmentation violation.

### **Calling MATLAB Functions from Stateflow Charts Caused Erroneous Results**

Calling MATLAB functions from Stateflow charts by using either the MATLAB namespace operator `m1` or the function form `m1()` with an argument whose data type was not `double` caused erroneous results.

## Version 5.1.1 (R13SP1) Stateflow and Stateflow Coder Software

This table summarizes what's new in V 5.1.1 (R13SP1):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary.	No	No

New features and changes introduced in this version are described here:

### Bind Actions

Bind actions bind specified data and events to a state. Data bound to a state can be changed by the actions of that state or its children. Other states and their children are free to read the bound data, but they cannot change it. Events bound to a state can be broadcast only by that state or its children. Other states and their children are free to listen for the bound event, but they cannot broadcast it.

Binding a function call event to a state also binds the function call subsystem that it calls. In this case, the function call subsystem becomes enabled when the binding state is entered and becomes disabled when the binding state is exited.

Bind actions are documented in the Stateflow and Stateflow Coder User's Guide.

### New API Method `idToHandle`

The Stateflow API method `idToHandle` is introduced to accommodate customers using old-style API scripts that use integer IDs for Stateflow

objects. `idToHandle` is a method of the Root object that converts the integer ID of a particular Stateflow object into an API handle to the object.

In the following example, the data `objectID` has the value of the ID for a particular Stateflow object, and the API handle `objectHandle` is used to change the name of the object.

```
rt = sfroot;  
objectHandle = rt.idToHandle(objectID);  
objectHandle.Name = 'ABC';
```

## **Code Generation Speed Improved**

Code generation now occurs faster than in the previous version, V5.1 (R13+).

### **Compatibility Considerations**

Part of the improvement in code generation speed comes from turning off the display of code generation messages in the MATLAB Command Window. Therefore, you will no longer see these messages in the MATLAB Command Window.

## Version 5.1 (R13+) Stateflow and Stateflow Coder Software

This table summarizes what's new in V5.1 (R13+):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Fixed Bugs	No

New features and changes introduced in this version are described here:

### Truth Tables

You can now represent truth tables in your Stateflow chart. Truth tables provide the convenience of specifying functions with logical behavior in a tabular form without having to draw flow graphs.

Because they are implemented on top of Stateflow graphical functions, Stateflow truth tables provide the following functionality:

- Truth tables can be called from anywhere in the chart
- Truth tables can call other truth tables
- Truth tables can access all of chart data (inputs, outputs, locals)
- Truth tables can broadcast events into Simulink models

Stateflow software provides its own analysis and diagnostics for underspecified and overspecified truth tables and fully integrates truth tables into the Stateflow Debugger and the Model Coverage tool.

Truth tables are documented in Truth Table Functions in the Stateflow and Stateflow Coder User's Guide.

## New Target Coder Options

You can use the following four new options for generating code for embedded and custom targets:

- **Compact nested if-else using logical AND/OR operators** — Improves readability of generated code by compacting nested if-else statements using logical AND (&&) and OR (|) operators.
- **Recognize if-elseif-else in nested if-else statements** — Improves readability of generated code by recognizing and emitting an if-elseif-else construct in place of deeply nested if-else statements.
- **Replace constant expressions by a single constant** — Improves readability by preevaluating constant expressions and emitting a single constant. Furthermore, this optimization enables elimination of dead code.
- **Minimize array reads using temporary variables** — In certain microprocessors, global array read operations are more expensive than accessing a temporary variable on the stack. Using this option minimizes array reads by using temporary variables when possible.

These coder options are documented in the Stateflow and Stateflow Coder User's Guide.



## Version 5.0 (R13) Stateflow and Stateflow Coder Software

This table summarizes what's new in V5.0 (R13):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary.	Fixed Bugs	Printable Release Notes: PDF V5.0 product documentation

New features and changes introduced in this version are:

- “Data and Event Handling” on page 117
- “New Stateflow API” on page 119
- “Functions and Actions” on page 120
- “Code Generation” on page 120
- “Greater Usability” on page 122

## Data and Event Handling

### Fixed-Point Data

Stateflow software now supports fixed-point data with the following features:

- Support for fixed-point data with both binary point scaling and slope and bias scaling
- Support for fixed-point operations included comparison, addition, subtraction, multiplication, and division
- Full coupling with input from and output to Simulink models
- Detection of overflow for fixed-point and integer types
- Convenient notation for expressing fixed-point literal constants in action language

- Automatic type promotion rules that select the default result type of an operation for maximum computational efficiency
- Full control for overflow prevention and precision retention using special `:=` assignment operator

This feature is documented in “How Fixed-Point Data Works in Stateflow Charts” in the Stateflow and Stateflow Coder User’s Guide.

### **Matrix Support for Data Inputs and Outputs**

You can now use two-dimensional matrices of any type for data with the scope **Data input from Simulink** or **Data output to Simulink**.

This feature is documented in “Adding Data” in the Stateflow and Stateflow Coder User’s Guide.

### **m1 Data Type**

You can use a new data type called `m1`. Data of this type is typed internally with the MATLAB type `mxArray`. This means that you can assign (store) any type of data available in a Stateflow chart to a data of type `m1`. This includes any type of data defined in a Stateflow chart or returned from the MATLAB workspace with the `m1` namespace operator or `m1` function.

This feature is documented in the Stateflow and Stateflow Coder User’s Guide.

### **Array and Matrix Support for m1 Namespace Operator and Function Call**

Stateflow software now supports vector arrays and n-dimensional matrices as arguments and return values for the `matlab` (`m1` for short) namespace operator and `matlab` (`m1` for short) function call. Before, only scalar data was supported.

This feature is documented in the Stateflow and Stateflow Coder User’s Guide.

### **A Stateflow Chart Allows Up to 254 Events**

You can now define up to 254 events per chart. The previous maximum was 127. If your chart has more than 254 events, an error message appears.

This feature is documented in the Stateflow and Stateflow Coder User's Guide.

## **New Stateflow API**

The new Stateflow API provides programmatic access to Stateflow objects. Through individual MATLAB commands or scripts of commands, you can manipulate Stateflow objects (machines, charts, states, boxes, functions, notes, transitions, junctions, data, events, and targets) to perform actions previously available only through Stateflow graphical interfaces. These actions include constructing new charts from scratch and modifying existing ones.

The Stateflow API provides control for the following Stateflow actions:

- Search for and find existing Stateflow objects, including charts, at any level of containment
- Set the triggering behavior of new or existing charts
- Create new Stateflow objects within charts with complete control over their positioning, scope, containment, and decomposition (states)
- Set the properties for all Stateflow objects
- Copy and paste Stateflow objects from one location to another
- Delete existing Stateflow objects
- Modify the graphical appearance of all Stateflow objects
- Parse Stateflow charts
- Change the debugging (simulation) behavior for simulation targets
- Build Stateflow simulation targets
- Change the deployment properties for Stateflow nonsimulation targets
- Build, compile, or generate code for other targets including embedded targets

This feature is documented in “Using the API” in the Stateflow API documentation.

## Functions and Actions

### Any Chart or Library Chart Can Export a Function

Any chart or library chart can now export a graphical function and any other chart or library chart can call it, as long as the caller and the called are both accessible through the same main model.

This feature is documented in the Stateflow and Stateflow Coder User's Guide.

### Trailing F Specifier for Single-Precision Floating-Point Numbers

Stateflow action language now recognizes a trailing F for specifying single-precision floating-point numbers, as in the action statement `x = 4.56F;`. In Stateflow action language, if a trailing F does not appear with a number, it is assumed to be double-precision. Specifying single-precision numbers allows you to save memory in generated code.

This feature is documented in the Stateflow and Stateflow Coder User's Guide.

### Error on Transition Action Into Junction with Following Condition

Transition actions for transitions into junctions with condition actions that follow as part of a state-to-state path are now flagged by an error. The error indicates that the execution of these actions is in reverse order to the apparent segment order in the chart.

**Compatibility Considerations.** You can modify existing code using the workaround documented in the Stateflow and Stateflow Coder User's Guide.

## Code Generation

### User Comments Included in Generated Code

You can include comments you enter in the action language of Stateflow charts in generated code for embedded and custom targets. This option is enabled by default.

This feature is documented in “Building Targets” in the Stateflow and Stateflow Coder User’s Guide.

### **For-Loops Emitted in Generated Code**

The Stateflow code generator detects and emits for-loops when applicable. In previous versions, the code generator always emitted while-loops.

### **Enhanced Integration with Real-Time Workshop Code Generation Software**

The code generated for Stateflow blocks is seamlessly integrated with code generated for other Simulink blocks, leading to more efficient and readable code.

### **Graphical Function Inlining in Generated Code**

Graphical functions with I/O can now be inlined in the generated code. You can specify inlining behavior (**auto**, **force inline**, **force no inline**) for every graphical function via its property dialog box. **auto** refers to a default strategy in which the Stateflow Coder code generation software decides when it is advantageous to inline a graphical function.

This feature is documented in the Stateflow and Stateflow Coder User’s Guide.

### **Performance Improvements**

When possible, Simulink input and output data to a Stateflow chart are made local, reducing RAM size. Whenever possible, these local inputs are conditionally evaluated (via *expression folding*) resulting in execution speed improvements.

### **Unnecessary Data Initialization Removed**

Unnecessary data initialization statements are now removed from the code generated for graphical functions.

### **Simple If Statements Optimized**

A simple Boolean expression evaluation scheme is used to optimize if statements such as `if(1)`, `if(0)`, `if(ON==OFF)`.

## Greater Usability

### Undo Operation in the Stateflow Editor

You can undo and redo operations you perform in the Stateflow Editor. When you undo an operation, you reverse the last edit operation you performed. After you undo operations, you can also redo them one-at-a-time. When you place your mouse cursor over the Undo or Redo buttons, the tooltip that appears indicates the nature of the operation to undo or redo.

This feature is documented in the Stateflow and Stateflow Coder User's Guide.

### New Model Report

A new model report is available for making comprehensive reports of Stateflow objects. Make this report in the Stateflow Editor by selecting **Print Details** from the **File** menu.

This feature is documented in the Stateflow and Stateflow Coder User's Guide.

### Stateflow Explorer Remembers Its Position and Size

Stateflow Explorer now remembers its position and size across sessions.

## Version 4.1 (R12.1) Stateflow and Stateflow Coder Software

This table summarizes what's new in V4.1 (R12.1):

<b>New Features and Changes</b>	<b>Version Compatibility Considerations</b>	<b>Fixed Bugs and Known Problems</b>	<b>Related Documentation at Web Site</b>
Yes Details below	Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary.	“Fixed Bugs” on page 125	No

New features and changes introduced in this version are:

- “Smart Transitions” on page 123
- “Search & Replace Tool Enhancements” on page 124
- “Stateflow Chart Notes” on page 124
- “Model Coverage Tool” on page 124
- “Single-Precision Constants in Code Generation” on page 125
- “Transition Actions into Junctions Disallowed” on page 125
- “Fixed Bugs” on page 125

### Smart Transitions

Stateflow charts now include the graphical innovation of smart transitions whose ends slide around the surfaces of states and junctions. When the source and/or destination objects are moved and resized in the Stateflow chart, these transitions use sliding and other behaviors to enable users to produce an aesthetically pleasing chart.

This feature is fully documented in “Extending Transitions with Smart Behavior” in the Stateflow and Stateflow Coder User’s Guide.

---

**Note** Transitions are automatically created with smart behavior on the assumption that this behavior is almost always desirable. However, *self-loop* transitions, must be created without smart transition behavior. See “Self-Loop Transitions” in the Stateflow and Stateflow Coder User’s Guide for instructions.

---

## Search & Replace Tool Enhancements

The Stateflow Search & Replace tool has been modified with the following enhancements:

- Regular expression tokenized replacements  
Allows you to dynamically choose replacement text based on matched text.
- Case insensitivity/case preservation  
Replaces text with different sensitivities to the use of upper or lower case characters in the found text.
- Addition of a **Search in:** field  
Now you can select any Stateflow chart in your Simulink model or select the entire Stateflow machine.

These enhancements are fully documented in the Stateflow and Stateflow Coder User’s Guide.

## Stateflow Chart Notes

You can now enter annotations to your Stateflow charts that are similar to annotations in Simulink models.

This feature is fully documented in “Using Notes to Extend Charts” in the Stateflow and Stateflow Coder User’s Guide.

## Model Coverage Tool

The Simulink Model Coverage tool has been modified to perform model coverage calculations for decisions and conditions of decision in the Stateflow



machine and its charts, states, and transitions. This includes Condition and MCDC coverage.

This enhancement is fully documented in “Understanding Model Coverage for Stateflow Charts” in the Stateflow and Stateflow Coder User’s Guide.

## Single-Precision Constants in Code Generation

Code generation now emits single-precision constants with a trailing F to the C-compiler. This results in lower ROM size. For example, the action language statement `x = y + single(1.375);` now generates the code `x = y + 1.375F;`

## Transition Actions into Junctions Disallowed

Transition actions are now permitted only on transitions that terminate on states. For the following reasons, transition actions are no longer permitted on transitions that terminate on junctions:

- The semantics of transition actions for transitions into junctions are complex and easily misunderstood and misused.
- The complex semantics of these transition actions also result in generated code that is inefficient. Eliminating these transition actions not only simplifies Stateflow charts but also results in generated code that is much more efficient.

## Compatibility Considerations

If your current model contains transition actions that terminate on junctions, they are flagged with an error. In most cases, replacing these transition actions with condition actions gives identical chart behavior.

## Fixed Bugs

### Editing Crossing Transitions out of Grouped Subcharts

Editing crossing transitions out of grouped subcharts caused model corruptions in Versions 3.0 through 4.0.2.

### **Disabled and Restored Library Chart Links**

Stateflow library chart links that are disabled and then restored caused model corruptions in Versions 3.0 through 4.0.2.

### **Too Many Action Statements During Simulation**

Stateflow Debugger caused an error during simulation when a state has more than 255 action statements in Versions 1.0 through 4.0.2.

### **False State Inconsistency Run-Time Error**

Charts within a Simulink enabled subsystem produced a false state inconsistency run-time error when the subsystem was disabled in Versions 2.0 through 4.0.3.

### **MATLAB Variables Improperly Overwritten**

Selecting **Save final value to base workspace** in the properties dialog box for a data item caused unrelated MATLAB variables in the workspace to be overwritten in Versions 4.0 through 4.0.3.

### **Target Options Fields Overwritten**

The **Custom Initialization** and **Custom Termination** fields in the Target Options dialog box were overwritten by empty strings in the data dictionary in Version 4.0.3.

### **Transitions Assertions**

Transitions containing a temporal trigger combined with other event triggers, such as `after(3E) | E2 | E3`, caused assertion errors during parsing in Version 4.0.3.

### **Build Failures with Custom Code**

When including custom code, build failures occurred due to DOS shell command-line length limitations because all user-defined folders on the MATLAB path were added to the include directory list in Version 4.0.3.

### **Code Generation for Default Transitions in Parallel States**

States with parallel decomposition with default transition paths generated incorrect code in Versions 3.0 through 4.0.2.

### **Code Generation for Double-Precision Whole Numbers**

Double-precision whole numbers were incorrectly emitted as integers (e.g., 5 instead of 5.0) in code generation in Versions 3.0 through 4.0.2.

## Version 4.0 (R12) Stateflow and Stateflow Coder Software

This table summarizes what's new in V4.0 (R12):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary.	No	No

New features and changes introduced in this version are described here:

### Improved Code Generation

Code generation from Stateflow Coder software has been significantly improved:

- The code looks hand written.
- ROM and RAM size rivals hand-written code.
- Code generation is faster.

### Automatic Upgrade to Release 12 of MATLAB Product Family

This release automatically upgrades all features introduced in previous versions of Stateflow software to work with Release 12 of the MATLAB product family.

### Compatibility Considerations

If you open a machine (model) made with a previous version of Stateflow software, you will see a warning message similar to the following:

```
Warning: An old Stateflow machine 'sf_car' is loaded.
This machine was saved with an older Stateflow 3.0311061000001.
```

Please save this machine again!

By saving the machine in the most recent version of Stateflow software, it is automatically upgraded.

## Version 3.0 (R11) Stateflow and Stateflow Coder Software

This table summarizes what's new in V 3.0 (R11):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	No	No

New features and changes introduced in this version are:

- “Temporal Logic” on page 130
- “Subcharts” on page 131
- “Graphical Functions” on page 131
- “Symbol Autocreation Wizard” on page 131
- “Command Toolbar” on page 132
- “Navigation Toolbar” on page 132
- “Straight Line Transitions” on page 132
- “Workspace-Based Data” on page 132
- “Explorer Copy Properties” on page 132
- “Library Link Icons” on page 133

### Temporal Logic

You can now use temporal conditions (before, after, at, every time) to determine the activation of transitions and duration of state activation. Temporal logic provides a simple paradigm for event scheduling and allows your Stateflow chart to express clearly and simply the time-dependent behavior of a system.

This feature is fully documented in “Using Temporal Logic in State Actions and Transitions” in the Stateflow and Stateflow Coder User’s Guide.

## Subcharts

You can now create charts within charts. A chart that is embedded in another chart is called a subchart. A subchart can contain anything a top-level chart can, including other subcharts. In fact, you can nest subcharts to any level. A subchart appears as a labeled block in the chart that contains it. You can create transitions among objects residing in different subcharts existing at the same level or at different levels. A transition that crosses subchart boundaries in this fashion is called a supertransition.

Subcharts help you reduce a complex chart to a set of simpler, hierarchically organized diagrams. This makes the chart easier to understand and maintain, without changing the semantics of the chart in any way.

Subchart boundaries are ignored when you simulate and generate code from Stateflow charts.

This feature is fully documented in “Using Subcharts to Extend Charts” in the Stateflow and Stateflow Coder User’s Guide.

## Graphical Functions

A graphical function is a function defined by a flow graph. Graphical functions are similar to textual functions, such as C and MATLAB functions. Like textual functions, graphical functions can accept arguments and return results. You invoke graphical functions in transition and state actions in the same way you invoke C and MATLAB functions. Unlike C and MATLAB functions, however, graphical functions are full-fledged Stateflow objects. You use the Stateflow Editor to create them, and they reside in your model along with the charts that invoke them. This makes graphical functions easier to create, access, and manage than textual functions, whose creation requires external tools and whose definitions reside separately from the model.

This feature is fully documented in “Using Graphical Functions to Extend Actions” in the Stateflow and Stateflow Coder User’s Guide.

## Symbol Autocreation Wizard

The Symbol Autocreation Wizard helps you add missing data and events to your Stateflow charts. When you parse the chart or run the simulation, this wizard detects data and events that you have not previously defined. The

wizard then opens automatically and heuristically recommends attributes for the unresolved data or events to help you define these symbols.

This feature is fully documented in “Symbol Autocreation Wizard” in the Stateflow and Stateflow Coder User’s Guide.

## **Command Toolbar**

The Stateflow Editor now contains a toolbar containing buttons for the most commonly used editing and simulation commands. The toolbar saves searching through menus for these commands.

## **Navigation Toolbar**

This toolbar contains buttons for navigating a chart hierarchy.

## **Straight Line Transitions**

You can now create straight lines between junctions. Transitions that are almost straight are automatically snapped straight during edit time. The snap-to-grid functionality helps align connected junctions vertically and horizontally.

## **Workspace-Based Data**

Data items can now be initialized from identically named variables in the MATLAB workspace and/or copied back to the workspace at the end of a simulation. Workspace-initialized constants consume no memory in generated code.

This feature is fully documented in the Stateflow and Stateflow Coder User’s Guide.

## **Explorer Copy Properties**

Stateflow Explorer now allows you to pick up properties from one data/event/target item and apply them to another data/event/target item or a group of items. This speeds up the process of creating charts that have objects with similar sets of properties.



This feature is fully documented in the Stateflow and Stateflow Coder User's Guide.

## **Library Link Icons**

In the Stateflow Explorer, an arrow distinguishes icons of library links from those of actual charts. Clicking a library link icon opens the library chart in the Stateflow Editor.

## Compatibility Summary for Stateflow and Stateflow Coder Software

This table summarizes new features and changes that might cause incompatibilities when you upgrade from an earlier version, or when you use files on multiple versions. Details are provided in the description of the new feature or change.

Version (Release)	New Features and Changes with Version Compatibility Impact
<b>Latest Version V7.5 (R2010a)</b>	See the <b>Compatibility Considerations</b> subheading for each of these new features or changes: <ul style="list-style-type: none"> <li>• “Data Change Implicit Event No Longer Supports Machine-Parented Data” on page 10</li> <li>• “Support for Machine-Parented Events Completely Removed” on page 10</li> </ul>
<b>V7.4 (R2009b)</b>	See the <b>Compatibility Considerations</b> subheading for each of these new features or changes: <ul style="list-style-type: none"> <li>• “New Compilation Report for Embedded MATLAB Functions in Stateflow Charts” on page 17</li> </ul>

<b>Version (Release)</b>	<b>New Features and Changes with Version Compatibility Impact</b>
V7.3 (R2009a)	<p>See the <b>Compatibility Considerations</b> subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> <li>• “Use of en, du, ex, entry, during, and exit for Data and Event Names Being Disallowed in a Future Version” on page 25</li> <li>• “Support for Machine-Parented Events Being Removed in a Future Version” on page 25</li> </ul>
V7.2 (R2008b)	<p>See the <b>Compatibility Considerations</b> subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> <li>• “Unified Simulation and Embeddable Code Generation Options for Stateflow Charts and Truth Table Blocks” on page 29</li> <li>• “Change in Casting Behavior When Calling MATLAB Functions in a Chart” on page 64</li> <li>• “Use of Output Data with Change Detection Operators Disallowed for Initialize-Outputs-at-Wakeup Mode” on page 65</li> <li>• “New Configuration Set Created When Loading Nonlibrary Models with an Active Configuration Reference” on page 66</li> </ul>
V7.1.1 (R2008a+)	None

<b>Version (Release)</b>	<b>New Features and Changes with Version Compatibility Impact</b>
V7.1 (R2008a)	See the <b>Compatibility Considerations</b> subheading for each of these new features or changes: <ul style="list-style-type: none"> <li>• “Using a Specific Path to a State for the in Operator” on page 71</li> <li>• “Specifying Scaling Explicitly for Fixed-Point Data” on page 74</li> </ul>
V7.0.1 (R2007b+)	None
V7.0 (R2007b)	See the <b>Compatibility Considerations</b> subheading for each of these new features or changes: <ul style="list-style-type: none"> <li>• “Enhanced Continuous-Time Support with Zero-Crossing Detection” on page 78</li> <li>• “Support for Inheriting Data Properties from Simulink Signal Objects Via Explicit Resolution” on page 78</li> </ul>
V6.6.1 (R2007a+)	None
V6.6 (R2007a)	None
V6.5 (R2006b)	None
V6.4.1 (R2006a+)	None
V6.4 (R2006a)	None
V6.3 (R14SP3)	None
V6.2 (R14SP2)	None
V6.1 (R14SP1)	None

<b>Version (Release)</b>	<b>New Features and Changes with Version Compatibility Impact</b>
V6.0 (R14)	<p>See the <b>Compatibility Considerations</b> subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> <li>• “Target Configuration Integrated with Simulink Targets” on page 101</li> <li>• “Parameter Scope for Simulink Parameters in Stateflow Charts” on page 104</li> <li>• “Defining Temporary Data for Charts” on page 105</li> <li>• “Fixed-Point Autoscaling of Stateflow Data” on page 105</li> <li>• “Support for Directional Vectors” on page 106</li> <li>• “Trailing 1’s Removed After Second Dimension of Array Size” on page 110</li> </ul>
V5.1.1 (R13SP1)	<p>See the <b>Compatibility Considerations</b> subheading for this new feature or change:</p> <ul style="list-style-type: none"> <li>• “Code Generation Speed Improved” on page 114</li> </ul>
V5.1 (R13+)	None

<b>Version (Release)</b>	<b>New Features and Changes with Version Compatibility Impact</b>
V5.0 (R13)	See the <b>Compatibility Considerations</b> subheading for this new feature or change: <ul style="list-style-type: none"><li>• “Error on Transition Action Into Junction with Following Condition” on page 120</li></ul>
V4.1 (R12.1)	See the <b>Compatibility Considerations</b> subheading for this new feature or change: <ul style="list-style-type: none"><li>• “Transition Actions into Junctions Disallowed” on page 125</li></ul>
V4.0 (R12)	See the <b>Compatibility Considerations</b> subheading for this new feature or change: <ul style="list-style-type: none"><li>• “Automatic Upgrade to Release 12 of MATLAB Product Family” on page 128</li></ul>
V3.0 (R11)	None