Stateflow® and Stateflow® Coder Release Notes

Contents

Summary by Version

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

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Latest Version V6.4.1 (R2006a+)	No	No	Bug Reports at Web site	Printable Release Notes:PDF V6.4.1 product documentation
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 Release Notes: PDF V5.0 product documentation
V4.1 (R12.1)	Yes Details	Yes Summary	"Fixed Bugs" on page 43	No

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
V4.0 (R12)	Yes Details	Yes Summary	No	No
V3.0 (R11)	Yes Details	No	No	No

About Release Notes

Use release notes when upgrading to a newer version to learn about new features and changes, and the potential impact on your existing files and practices. Release notes are also beneficial if you use or support multiple versions.

If you are not upgrading from the most recent previous version, review release notes for all interim versions, not just for the version you are installing. For example, when upgrading from V1.0 to V1.2, review the New Features and Changes, Version Compatibility Considerations, and Bug Reports for V1.1 and V1.2.

New Features and Changes

These include

- New functionality
- Changes to existing functionality
- Changes to system requirements (complete system requirements for the current version are at the MathWorks Web site)
- Any version compatibility considerations associated with each new feature or change

Version Compatibility Considerations

When a new feature or change introduces a known incompatibility between versions for new or existing features, its description includes a **Compatibility Considerations** subsection that details the impact. For a list of all new

features and changes that have compatibility impact or limitations, see the "Compatibility Summary for Stateflow and Stateflow Coder" on page 51.

Compatibility issues that become known after the product has been released are added to Bug Reports at the MathWorks Web site. Because bug fixes can sometimes result in incompatibilities, also review fixed bugs in Bug Reports for any compatibility impact.

Fixed Bugs and Known Problems

MathWorks Bug Reports is a user-searchable database of known problems, workarounds, and fixes. The MathWorks updates the Bug Reports database as new problems and resolutions become known, so check it as needed for the latest information.

Access Bug Reports at the MathWorks Web site using your MathWorks Account. If you are not logged in to your MathWorks Account when you link to Bug Reports, you are prompted to log in or create an account. You then can view bug fixes and known problems for R14SP2 and more recent releases.

The Bug Reports database was introduced for R14SP2 and does not include information for prior releases. You can access a list of bug fixes made in prior versions via the links in the summary table.

Related Documentation at Web Site

Printable Release Notes (PDF). You can print release notes from the PDF version, located at the MathWorks Web site. The PDF version does not support links to other documents or to the Web site, such as to Bug Reports. Use the browser-based version of release notes for access to all information.

Product Documentation. At the MathWorks Web site, you can access complete product documentation for the current version and some previous versions, as noted in the summary table.

Version 6.4.1 (R2006a+) Stateflow and Stateflow Coder

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

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
No	No	Bug Reports at Web site	Printable Release Notes:PDF V6.4.1 product documentation

Version 6.4 (R2006a) Stateflow and Stateflow Coder

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

Stateflow provides a new chart option **Initialize Outputs Every Time Chart Wakes Up**. When you enable this option, Stateflow initializes the value of outputs every time a chart wakes up, not only at time 0 (see "Setting Properties for Individual Charts" in the online Stateflow 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

This release allows you to use M-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 Stateflow documentation.

Using the MATLAB Workspace Browser for Debugging Stateflow

The MATLAB® Workspace Browser is no longer available for debugging Stateflow. 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 Windows 64

No C compiler ships with Stateflow on Windows 64. Because Stateflow 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 Windows 64 include Microsoft Visual Studio 2005 and the Microsoft Platform SDK.

Version 6.3 (R14SP3) Stateflow and Stateflow Coder

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

Sharing Global Data Between Simulink and Stateflow

This release provides an interface that gives Stateflow charts access to global variables in Simulink® models. Simulink 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 by reading and writing data store memory symbolically using the Stateflow action language. See "Sharing Global Data with Simulink" in the Stateflow and Stateflow Coder User's Guide documentation.

Enhancements to Stateflow Data Dialog

The Stateflow data dialog has been enhanced to

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

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

- 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, Stateflow generates a compile-time error.

See “Defining Events and Data” in the Stateflow and Stateflow Coder User’s Guide documentation.

Truth Table Enhancements

Using Embedded MATLAB Action Language in Truth Tables

You now have an option to use the embedded MATLAB action language in Stateflow truth tables. Previously, you were restricted to the Stateflow action language. The 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 documentation.

Embedded MATLAB Truth Table Block in Simulink

A truth table function block is now available as a Stateflow 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 action language only. You must have a Stateflow license to use the truth table block in Simulink.

See Truth Table in the Stateflow and Stateflow Coder User's Guide documentation.

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 diagram.

Default Case Handling in Stateflow Generated Code

Stateflow 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. Re-entering 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

This release gives you the ability to 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 diagram, 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 documentation.

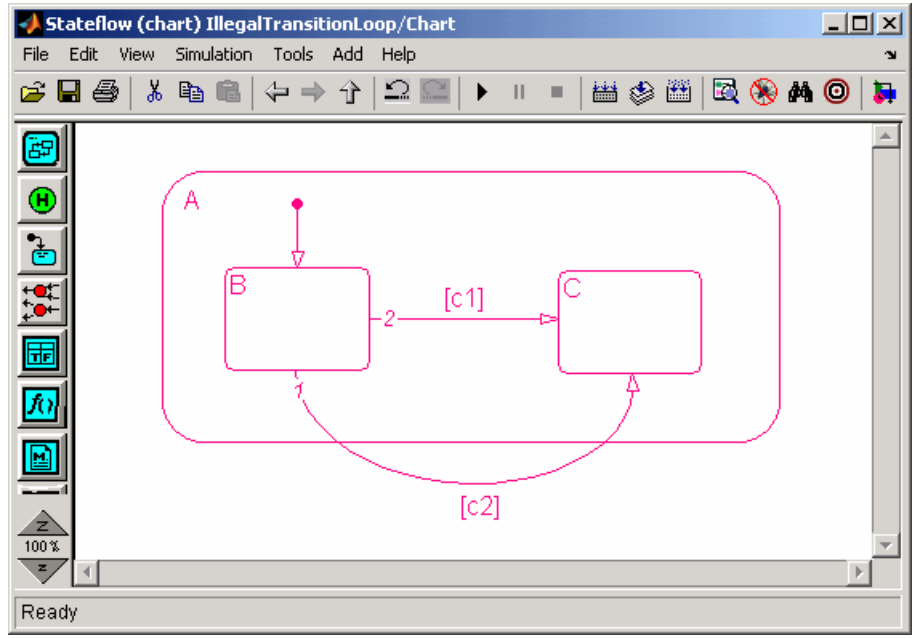
Hyperlinking Simulink Subsystems from Stateflow Events

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

Warnings for Transitions Looping Out of Logical Parent

A common modeling error in Stateflow 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 ancester.

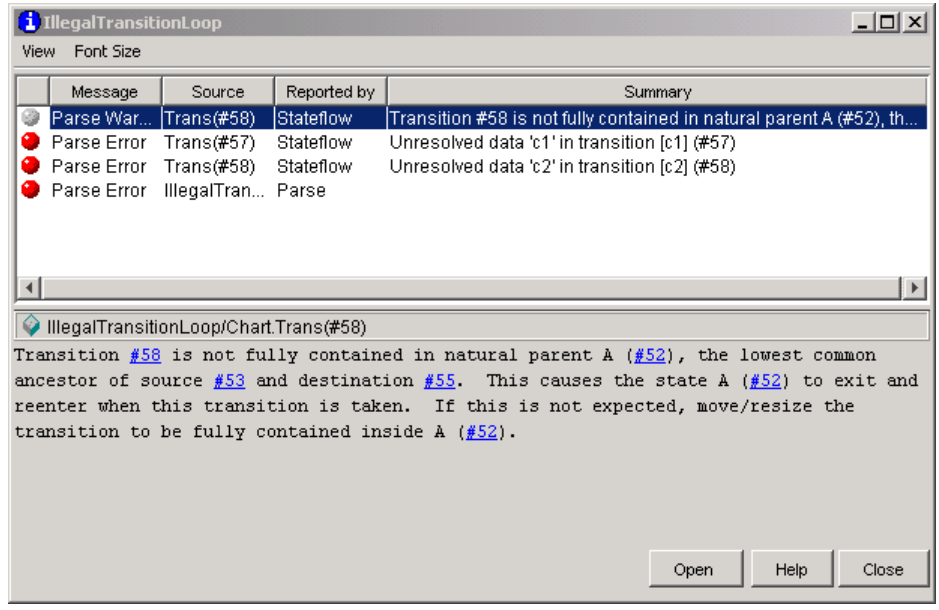
Consider the following example:



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, Stateflow executes the exit action of `A` and then executes the entry action of `A`. 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

This release adds a user preference that gives you the option of using 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 documentation.

Stateflow Chart Notes Click Function

This release introduces enhancements to Stateflow chart notes. The chart notes property dialog 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 Using Simulink documentation for a description of these new options.

Chart Viewing Enhancements

This release adds the following chart viewing enhancements:

View Command History. This release enhances the chart viewing commands. Stateflow 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 13).

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 13.

View Command Shortcut Keys. This release adds the following viewing command shortcut keys in Microsoft Windows and UNIX:

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

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

Version 6.2 (R14SP2) Stateflow and Stateflow Coder

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 chart 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.

Stateflow and Embedded MATLAB Support Simulink Datatype Aliases

Stateflow and Embedded MATLAB data may now be explicitly typed using the same aliased types that Simulink uses. Also, inherited and parameterized data types in Stateflow and Embedded MATLAB support propagation of aliased types. Code generated by Stateflow and Embedded MATLAB 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

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:

RTW Code Generation Fails for Charts with No Output Data

In this release, `RTW` 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

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 **Compatibility Considerations**, below. See also Summary.	Fixed Bugs See also "Additional Bug Fixes" on page 29	No

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

Code Generation

Multibyte Comment Characters in Generated Code

You can optionally include Stateflow action language comments in generated code for a Real-Time Workshop or Stateflow 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 online Stateflow User’s Guide documentation for more information on inserting comments into Stateflow action language and optionally into Stateflow generated code.

Target Configuration Integration with Simulink and RTW

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

For more information on configuring targets for models with Stateflow blocks in the Simulink **Configuration Parameters** dialog, see the following sections in Stateflow documentation:

- “Configuring a Simulation Target for Stateflow”
- “Configuring Real-Time Workshop for Stateflow”

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

- You need to update existing API scripts that modify properties on Stateflow RTW targets.
- Stateflow no longer copies custom code settings from simulation target to RTW target for nonlibrary models.

Updating Scripts that Modify Code on Stateflow RTW Targets. If you have existing M-file scripts that use the Stateflow programmatic API to set up custom-code settings on the Stateflow RTW target, you must modify these M-file scripts to use the Simulink/RTW configuration set. The following example illustrates how the properties on the old RTW target object map to the new configuration set object.

Here is a sample old script modifying custom code properties of an RTW target 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 M-file 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 Simulation Target to RTW Target. In prior versions of Stateflow, creating an RTW target copied custom code settings from the simulation target to the new RTW target. This is no longer true for nonlibrary models, which configure an RTW target in Simulink, not Stateflow. However, for library models you create an RTW target in Stateflow that still copies the custom code settings of the simulation target in Stateflow.

Custom C++ Code Included with Stateflow Generated Code

With the proper modifications, you can now include C++ custom code with Stateflow generated code. See “Including Custom C++ Code” in Stateflow online documentation.

Object Descriptions in Generated Code for RTW Targets

The object descriptions entered in the property dialogs for charts, states, transitions, and graphical functions are now optionally propagated into RTW generated code with RTW Embedded Coder license. You can enable this option in the Simulink **Configuration Parameters** dialog 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 and Stateflow

All Stateflow code generation settings for Real-Time Workshop are now consolidated into the new Simulink/RTW **Configuration Parameters** dialog. This enables users to specify code generation settings for Simulink and Stateflow 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 for Real-Time Workshop, although they can still be entered on the Real-Time Workshop target object for Stateflow. All custom code settings and Stateflow optimization settings from old models are grandfathered into the new **Configuration Parameters** dialog at load time.

Faster Real-Time Workshop Code Generation

Stateflow uses a two-phase code-generation scheme with Real-Time Workshop. 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 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 RTW code generation for models with a large number of Stateflow charts. If, for any reason, you desire to turn off the incremental TLC generation, use the following command, which lasts until the end of the current MATLAB session:

```
sf('Feature','RTW Incremental CodeGen',0);
```

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 from Simulink/Stateflow

model are now uniformly managed for a common look and feel, and to prevent name collisions.

Data Handling

Data Type and Size Inheritance from Simulink

Stateflow inherits the type and size for input and output data from the signals that connect to them in Simulink. 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 Simulink, enter `Inherited` for their **Type** property. To inherit input or output data size from their block connections in Simulink, enter `-1` for their **Size** property.

See "Inheriting Data Types from Simulink" and "Inheriting Input and Output Data Size from Simulink" in the Stateflow documentation 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 Stateflow documentation for more details.

Parameter Scope for Simulink Parameters in Stateflow

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 Stateflow documentation for more information.

In previous versions, the mechanism was to create data of type Constant and enable the Initialize From Workspace property. This 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, Stateflow threw an error instead of trying to resolve the parameter in the next parent's workspace.
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 in Stateflow. 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. For a procedure on how to autoscale fixed-point data in Simulink, see the topic Automatic Scaling in Fixed-Point Blockset documentation.

Compatibility Considerations. Stateflow implements the Simulink override of fixed-point numbers in Stateflow charts, with the following exceptions:

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

Stateflow Supports Directional Vectors

In Stateflow 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 Stateflow 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 Simulink. 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

Stateflow now supports the cast operator `cast` for all Stateflow data types except `ml` and fixed-point data. To make casting easier, Stateflow also provides 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 Type Cast Operations in Stateflow online documentation.

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

Stateflow now gives users access to a special subset of MATLAB programming 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.

Stateflow eML functions provide the following functionality to users:

- An editing environment for entering Embedded MATLAB function M-code
- A subset of MATLAB that you can use in Embedded MATLAB functions. See “Using the Embedded MATLAB Function Block” in online Simulink 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 Simulink uses the same underlying infrastructure as Stateflow for simulation (through code-generation), debugging, and integration with Real-Time Workshop. 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 Stateflow as the source of the warning or error.

3 Optimization settings that are specific to Stateflow in the RTW configuration object apply equally to Embedded MATLAB Function blocks. These optimization settings appear for the **Optimization** node in the Simulink **Configuration Parameters** dialog only when a Stateflow block or an Embedded MATLAB Function block is present in the model. They are identified and described in the topic Configuring Stateflow Blocks in Nonlibrary Models for Real-Time Workshop in Stateflow documentation.

4 Selecting the Rebuild All button in the **Embedded MATLAB Editor** or in a Stateflow diagram 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 with 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 infrastructure but does not require a Stateflow license.

See Using Embedded MATLAB Functions in Stateflow online documentation.

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 *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 Data Values and State Activity" in Stateflow documentation 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 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 Stateflow documentation for an example.

Command Line Debugger in MATLAB

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 in MATLAB, see the topic Using the Command Line Debugger to Watch Data in Stateflow documentation.

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 Added to the Model Explorer

The new Model Explorer tool in Simulink now lists Stateflow objects in its hierarchy of Simulink model objects. In place of the old Explorer tool in Stateflow, you can now use the Model Explorer tool to add and modify data and events to Stateflow objects. For complete details on using the Model Explorer from Stateflow, see Using the Model Explorer with Stateflow Objects in Stateflow documentation.

For details on using the Model Explorer in Simulink, see "The Model Explorer" in Simulink documentation.

Warnings for Missing and Conditional Default Paths

A common modeling error in Stateflow 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.

Stateflow now detects both of these cases during code generation and generates a warning to fix the model.

Automatic Upgrade of Stateflow Machines Developed in Previous Versions

When you open models containing Stateflow charts that were developed in earlier versions, Stateflow automatically upgrades the Stateflow machine to work in the current version. Stateflow no longer displays a warning or requires that you save the model in the current version of Stateflow.

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 by Stateflow. Here are some examples.

Size Specified by User	Size Specified by Stateflow
`[2, 3, 1]`	`[2, 3]`
`[3, 5, 1, 1, 1]`	`[3, 5]`
`[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 diagram 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 `ml` or the function form `ml()` with an argument whose data type was not `double` caused erroneous results.

Version 5.1.1 (R13SP1) Stateflow and Stateflow Coder

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 **Compatibility Considerations**, 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 Bind Actions and Using Bind Actions to Control Function Call Subsystems in Stateflow online documentation.

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

Stateflow now generates code faster than in the previous version, Stateflow version 5.1 (R12.1).

Compatibility Considerations

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

Version 5.1 (R13+) Stateflow and Stateflow Coder

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

Stateflow now provides a concise and powerful mechanism to represent Truth Tables as part of a state diagram. 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 to users:

- Truth tables are be called from anywhere on 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

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

Truth tables are documented in Truth Tables in Stateflow online documentation.

New RTW/Custom Target Coder Options

Stateflow provides users with the following four new options for generating code for Real-Time Workshop (RTW) 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 opens up opportunities for eliminating 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 RTW Target (rtw) Coder Options in Stateflow online documentation.

Version 5.0 (R13) Stateflow and Stateflow Coder

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 **Compatibility Considerations**, below. See also Summary.	Fixed Bugs	Printable Release Notes: PDF V5.0 product documentation

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

Data and Event Handling

Fixed-Point Data

Stateflow 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
- 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 "Using Fixed-Point Data in Stateflow" in the Stateflow documentation.

Matrix Support for Data Inputs and Outputs

Stateflow now supports 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 documentation.

ml Data Type

Stateflow supports a new data type called `ml`. 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 Stateflow to a data of type `ml`. This includes any type of data defined in Stateflow or returned from MATLAB with the `ml` namespace operator or `ml` function.

This feature is documented in ml Data Type in the Stateflow documentation.

Array and Matrix Support for ml Namespace Operator and Function Call

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

This feature is documented in MATLAB Functions and Variables in the Stateflow documentation.

Stateflow Allows Up to 254 Events

Stateflow now handles up to 254 events per chart. The previous maximum was 127. Stateflow now throws an error if your chart has more than 254 events.

This feature is documented in Adding Events to the Data Dictionary in the Stateflow documentation.

New Stateflow API

The new Stateflow API provides programmatic access to Stateflow. 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 diagrams 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 non-simulation targets
- Build, compile, or generate code for other targets including RTW targets

This feature is documented in Stateflow and Stateflow Coder API in the Stateflow 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 Exporting Graphical Functions in the Stateflow documentation.

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 Special Symbols in the Stateflow documentation.

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 diagram.

Compatibility Considerations. You can modify existing code using the workaround documented in Code Generation Error Messages in the Stateflow documentation.

Code Generation

User Comments Included in Generated Code

Stateflow provides the option for including comments you enter in the action language of Stateflow diagrams in generated code for `RTW` and custom targets. This option is enabled by default.

This feature is documented in Specifying Code Generation Options in the Stateflow documentation.

For-Loops Emitted in Generated Code

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

Enhanced Integration with Real-Time Workshop

The code generated for Stateflow blocks is seamlessly integrated with Simulink, 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 decides when it is advantageous to inline a graphical function.

This feature is documented in Specifying Graphical Function Properties in the Stateflow documentation.

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 Undoing and Redoing Editor Operations in the *Stateflow User's Guide* and Stateflow online help.

New Model Report

A new model report is available in Stateflow (also in Simulink) for making comprehensive reports of Stateflow objects. Make this report in the Stateflow diagram editor by selecting **Print Details** from the **File** menu.

This feature is documented in Generating a Model Report in Stateflow in the Stateflow documentation.

Stateflow Explorer Remembers Its Position and Size

Stateflow Explorer now remembers its position and size across sessions with Stateflow.

Version 4.1 (R12.1) Stateflow and Stateflow Coder

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

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	"Fixed Bugs" on page 43	No

New features and changes introduced in this version are

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 diagram.

This feature is fully documented in Setting Smart Behavior in Transitions in the Stateflow documentation and Stateflow online help.

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 the section Creating Self-Loop Transitions in online Stateflow User's Guide documentation 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 Searching and Replacing in Charts in the Stateflow documentation and Stateflow online help.

Stateflow Chart Notes

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

This feature is fully documented in Entering Chart Notes in the Stateflow documentation and Stateflow online help.

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 Stateflow Chart Model Coverage in the Stateflow documentation and Stateflow online help.

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 diagrams 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 Stateflow 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 Stateflow 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 Stateflow 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 Stateflow Versions 2.0 through 4.0.3.

MATLAB Variables Improperly Overwritten

Selecting **Save final value to base workspace** in the properties dialog for a data item caused unrelated MATLAB variables in the workspace to be overwritten in Stateflow 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 Stateflow 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 Stateflow 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 directories on the MATLAB path were added to the include directory list in Stateflow 4.0.3.

Code Generation for Default Transitions in Parallel States

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

Version 4.0 (R12) Stateflow and Stateflow Coder

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 **Compatibility Considerations**, below. See also Summary.	No	No

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

Improved Code Generation

The Stateflow Coder 4.0 code generation 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

Stateflow 4.0 automatically upgrades all features introduced in previous versions of Stateflow 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, 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, it is automatically upgraded.

Version 3.0 (R11) Stateflow and Stateflow Coder

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

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 model to express clearly and simply the time-dependent behavior of a system.

This feature is fully documented in Temporal Logic Operators in the Stateflow documentation and Stateflow online help.

Subcharts

Stateflow now allows you to 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 enable you to 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.

Stateflow ignores subchart boundaries when simulating and generating code from Stateflow models.

This feature is fully documented in "Using Subcharts to Extend Charts" in the Stateflow documentation and Stateflow online help.

Graphical Functions

A graphical function is a function defined by a flow graph. Graphical functions are similar to textual functions, such as MATLAB and C 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 MATLAB and C 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 Stateflow model along with the diagrams 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 Functions to Extend Actions" in the Stateflow documentation and Stateflow online help.

Symbol Autocreation Wizard

The Symbol Autocreation Wizard helps you to add missing data and events to your Stateflow charts. When you parse the diagram or run the simulation,

this wizard detects whenever data and events have not been previously defined in the Stateflow Explorer. The wizard then opens automatically and heuristically recommends attributes for the unresolved data or events to help you to quickly define these symbols.

This feature is fully documented in Symbol Autocreation Wizard in the Stateflow documentation and Stateflow online help.

Command Toolbar

The Stateflow editor now contains a toolbar containing buttons for the most commonly used editing and simulation commands in Stateflow. 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 Setting Data Properties in the Stateflow documentation and Stateflow online help.

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 diagrams that have objects with similar sets of properties.

This feature is fully documented in Transferring Object Properties in the Stateflow documentation and Stateflow online help.

Library Link Icons

An arrow distinguishes icons of library links from those of actual charts in the Stateflow Explorer. Clicking a library link icon opens the library chart in the Stateflow Editor. Stateflow 4.0 requires Release 12, including Simulink 4.0.

Compatibility Summary for Stateflow and Stateflow Coder

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
Latest Version V6.4.1 (R2006a+)	None
V6.4 (R2006a)	None
V6.3 (R14SP3)	None
V6.2 (R14SP2)	None
V6.1 (R14SP1)	None
V6.0 (R14)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "Target Configuration Integration with Simulink and RTW" on page 19 • "Parameter Scope for Simulink Parameters in Stateflow" on page 22 • "Defining Temporary Data for Charts" on page 23 • "Fixed-Point Autoscaling of Stateflow Data" on page 24 • "Stateflow Supports Directional Vectors" on page 24

Version (Release)	New Features and Changes with Version Compatibility Impact
	• "Trailing 1's Removed After Second Dimension of Array Size" on page 29
V5.1.1 (R13SP1)	See the **Compatibility Considerations** subheading for this new feature or change: • "Code Generation Speed Improved" on page 32
V5.1 (R13+)	None
V5.0 (R13)	See the **Compatibility Considerations** subheading for this new feature or change: • "Error on Transition Action Into Junction with Following Condition" on page 38
V4.1 (R12.1)	See the **Compatibility Considerations** subheading for this new feature or change: • "Transition Actions into Junctions Disallowed" on page 43
V4.0 (R12)	See the **Compatibility Considerations** subheading for this new feature or change: • "Automatic Upgrade to Release 12 of MATLAB" on page 46
V3.0 (R11)	None