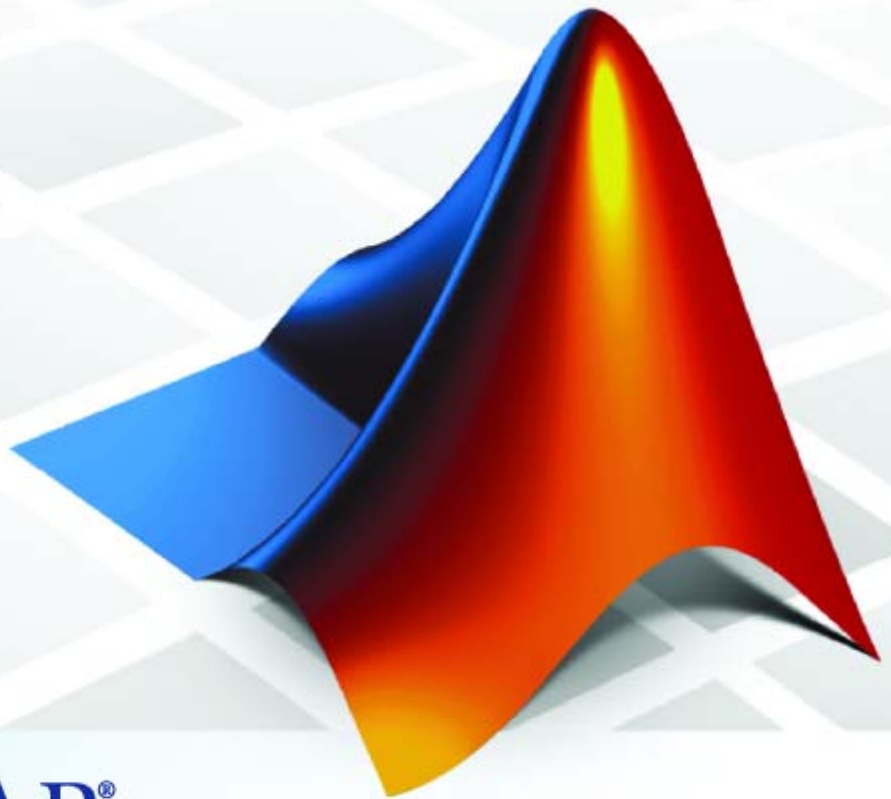


Stateflow[®] and Stateflow[®] Coder 6 API



MATLAB[®]
& **SIMULINK[®]**

How to Contact The MathWorks



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

Web
Newsgroup
Technical Support



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

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



508-647-7000 (Phone)



508-647-7001 (Fax)



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

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

Stateflow and Stateflow Coder API

© COPYRIGHT 2004–2006 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, Simulink, Stateflow, Handle Graphics, Real-Time Workshop, and xPC TargetBox are registered trademarks, and SimBiology, SimEvents, and SimHydraulics are trademarks of The MathWorks, Inc.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Patents

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

Revision History

June 2004	Online only	Revised for Stateflow 6.0 (Release 14)
October 2004	Online only	Revised for Stateflow 6.1 (Release 14SP1)
March 2005	Online only	Revised for Stateflow 6.21 (Release 14SP2)
September 2005	Online only	Revised for Stateflow 6.3 (Release 14SP3)
March 2006	Online only	Revised for Stateflow 6.4 (Release R2006a)
September 2006	Online only	Revised for Stateflow 6.5 (Release R2006b)

Using the API

1

Overview of the Stateflow API	1-3
What Is the Stateflow API?	1-3
Stateflow API Object Hierarchy	1-4
Getting a Handle on Stateflow API Objects	1-6
Using API Object Properties and Methods	1-7
API References to Properties and Methods	1-8
Quick Start for the Stateflow API	1-9
Create a New Model and Chart	1-9
Access the Model Object	1-9
Access the Chart Object	1-10
Create New Objects in the Chart	1-11
Accessing the Properties and Methods of Objects	1-17
Naming Conventions for Properties and Methods	1-17
Using Dot Notation with Properties and Methods	1-17
Using Function Notation with Methods	1-18
Displaying Properties and Methods	1-20
Displaying Properties	1-20
Displaying the Names of Methods	1-20
Displaying Property Subproperties	1-21
Displaying Enumerated Values for Properties	1-22
Creating and Destroying API Objects	1-23
Creating Stateflow Objects	1-23
Establishing an Object's Parent (Container)	1-25
Destroying Stateflow Objects	1-26
Accessing Existing Stateflow Objects	1-27
Finding Objects	1-27
Finding Objects at Different Levels of Containment	1-28
Retrieving Recently Selected Objects	1-30

Getting and Setting the Properties of Objects	1-31
Copying Objects	1-33
Accessing the Clipboard Object	1-33
copy Method Limitations	1-33
Copying by Grouping (Recommended)	1-34
Copying Objects Individually	1-35
Using the Editor Object	1-37
Accessing the Editor Object	1-37
Changing the Stateflow Display	1-37
Entering Multiline Labels	1-39
Creating Default Transitions	1-40
Making Supertransitions	1-41
Creating a MATLAB Script of API Commands	1-43

API Properties and Methods by Use

2

Reference Table Column Descriptions	2-3
Access Methods	2-4
Code Generation and Target Building	2-5
Code Generation and Build Methods	2-5
Code Generation Properties	2-6
Custom Code Properties	2-8
Display Control	2-10
Display Methods	2-10
Display Properties	2-10

Graphical Appearance	2-11
Color Properties	2-11
Drawing Properties	2-12
Font Properties	2-13
Position Properties	2-16
Text Properties	2-19
Creating and Deleting Objects	2-20
Containment	2-21
Data Definition Properties	2-22
Debugging Properties	2-25
Identifiers	2-28
Interface to Simulink	2-30
Machine (Model) Identifier Properties	2-34
Truth Table Construction Properties	2-35

API Properties and Methods — By Category

3

Reference Table Columns	3-6
Constructor Methods	3-7
Editor Properties	3-8
Editor Methods	3-9
Clipboard Methods	3-10

All Object Methods	3-11
Root Methods	3-12
Machine Properties	3-13
Machine Methods	3-17
Chart Properties	3-18
Chart Methods	3-26
State Properties	3-27
State Methods	3-31
Box Properties	3-33
Box Methods	3-35
Graphical Function Properties	3-36
Graphical Function Methods	3-39
Truth Table Properties	3-40
Truth Table Methods	3-43
Truth Table Chart Properties	3-44
Truth Table Chart Methods	3-47
Embedded MATLAB Function Properties	3-48
Embedded MATLAB Function Methods	3-51

Note Properties	3-52
Note Methods	3-54
Transition Properties	3-55
Transition Methods	3-59
Junction Properties	3-60
Junction Methods	3-61
Data Properties	3-62
Data Methods	3-70
Event Properties	3-71
Event Methods	3-74
Target Properties	3-75
CodeFlagsInfo Property of Targets	3-77
Target Methods	3-80

API Methods — Alphabetical List

4 |

Index

Using the API

The procedures and conceptual information in this chapter explain the basic operations of the Stateflow® Application Program Interface (API). It includes the following sections:

Overview of the Stateflow API
(p. 1-3)

Introduces you to concepts you need to know to understand the Stateflow API and how to use it to create and edit Stateflow diagrams.

Quick Start for the Stateflow API
(p. 1-9)

Step-by-step instructions for constructing a Stateflow diagram with the Stateflow API.

Accessing the Properties and Methods of Objects (p. 1-17)

Describes the conventions used in naming the properties and methods of Stateflow API objects and the rules for using them in commands.

Displaying Properties and Methods
(p. 1-20)

Information on calling built-in methods for listing properties and methods for each object type.

Creating and Destroying API Objects
(p. 1-23)

Information on creating and destroying any Stateflow object with the Stateflow API, and how to connect one object with another.

Accessing Existing Stateflow Objects
(p. 1-27)

Create handles to any object in an existing Stateflow diagram and use them to manipulate actual Stateflow objects in a Stateflow diagram.

Copying Objects (p. 1-33)	Learn the copy and paste procedure for copying Stateflow objects from one environment to another.
Using the Editor Object (p. 1-37)	Access the Editor object for a Stateflow diagram to perform operations that are graphical only, such as changing fonts and colors.
Entering Multiline Labels (p. 1-39)	The Stateflow API provides two techniques to enter text with more than one line for the labels of states and transitions.
Creating Default Transitions (p. 1-40)	The Stateflow API provides two means for making default transitions.
Making Supertransitions (p. 1-41)	Describes how to create supertransitions.
Creating a MATLAB Script of API Commands (p. 1-43)	You can execute your API commands in a single MATLAB® script.

Overview of the Stateflow API

The Stateflow API is a textual programming interface with the Stateflow diagram editor from the MATLAB Command Window. Before you get started with the “Quick Start for the Stateflow API” on page 1-9, read the topics in this section for an introduction to some new concepts that are part of the Stateflow API:

- “What Is the Stateflow API?” on page 1-3 — Defines and describes the nature of the Stateflow API.
- “Stateflow API Object Hierarchy” on page 1-4 — Introduces you to the hierarchy of objects in the Stateflow API, which mimics the hierarchy of objects in Stateflow.
- “Getting a Handle on Stateflow API Objects” on page 1-6 — Introduces you to the concept of a handle that is used to represent a Stateflow API object in MATLAB.
- “Using API Object Properties and Methods” on page 1-7 — Introduces you to the properties and methods of each API object. Properties and methods do the work of the API to create and change Stateflow diagrams.
- “API References to Properties and Methods” on page 1-8 — Introduces you to the extensive reference information available in this guide on each individual property and method.

Caution You cannot undo any operation to the Stateflow diagram editor performed through the Stateflow API. If you do perform an editing operation through the API, the undo and redo buttons are disabled from undoing and redoing any prior operations.

What Is the Stateflow API?

The Stateflow Application Programming Interface (API) is a tool of convenience that lets you create or change Stateflow diagrams with MATLAB commands. By placing Stateflow API commands in a MATLAB script, you can automate Stateflow diagram editing processes in a single command.

There are many possible applications for the Stateflow API. Here are some:

- Create a script that performs common graphical edits that makes editing of Stateflow diagrams easier.
- Create a script that immediately creates a repetitive "base" Stateflow diagram.
- Create a script that produces a specialized report of your model.

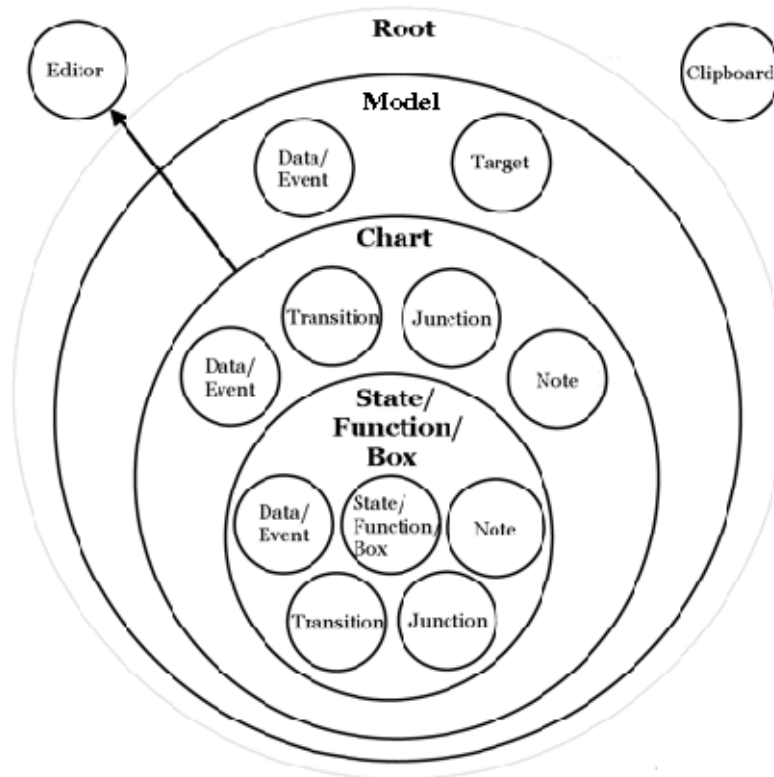
The Stateflow API consists of objects that represent actual Stateflow objects. For example, an API object of type `State` represents a Stateflow state, an API object of type `Junction` represents a Stateflow junction, and so on.

Each API object has methods and properties you use to perform editing operations on it. The correspondence between API object and Stateflow object is so close that what you do to a Stateflow API object affects the object it represents in the Stateflow diagram editor, and what you do to a graphical object in the Stateflow diagram editor affects the Stateflow API object that represents it.

The following topics introduce you to the objects, properties, and methods of the Stateflow API.

Stateflow API Object Hierarchy

Stateflow API objects represent actual Stateflow objects in a Stateflow diagram. Like Stateflow objects, API objects contain or are contained by other Stateflow objects. For example, if state A contains state B in the Stateflow Editor, then the API object for state A contains the API object for state B. The following diagram depicts the Stateflow API hierarchy of objects:



Rules of containment define the Stateflow and Stateflow API object hierarchy. For example, charts can contain states but states cannot contain charts. The hierarchy of Stateflow objects, also known as the Stateflow data dictionary, is depicted in the section “Stateflow Hierarchy of Objects” in the Stateflow and Stateflow Coder User’s Guide documentation. The Stateflow API hierarchy is very similar to the hierarchy of the Stateflow data dictionary and consists of the following layers of containment:

- **Root** — The Root object (there is only one) serves as the parent of all Stateflow API objects. It is a placeholder at the top of the Stateflow API hierarchy to distinguish Stateflow tool objects from the objects of other tools such as Simulink® and Handle Graphics®. The Root object is automatically created when you load a model containing a Stateflow chart or call the function `sfnew` to create a new model with a Stateflow chart.

- **Model** — Objects of type Model are accessed through the Stateflow Root object. Model objects are equivalent to Simulink models from a Stateflow perspective. They can hold objects of type Chart, Data/Event, and Target.
- **Chart** — Within any Model object (model) there can be any number of chart objects. Within each object of type Chart, there can be objects of type State, Function, Box, Note, Data, Event, Transition, and Junction. These objects represent the components of a Stateflow chart.
- **State/Function/Box** — Nested within objects of type State, Function, and Box, there can be further objects of type State, Function, Box, Note, Junction, Transition, Data, and Event. Levels of nesting can continue indefinitely.

The preceding figure also shows two object types that exist outside the Stateflow containment hierarchy, which are as follows:

- **Editor** — Though not a part of the Stateflow containment hierarchy, an object of type Editor provides access to the purely graphical aspects of objects of type Chart. For each Chart object there is an Editor object that provides API access to the Chart object's diagram editor.
- **Clipboard** — The Clipboard object has two methods, copy and pasteTo, that use the clipboard as a convenient staging area to implement the operation of copy and paste functionality in the Stateflow API.

Getting a Handle on Stateflow API Objects

You manipulate Stateflow objects by manipulating the Stateflow API objects that represent them. You manipulate Stateflow API objects through a MATLAB variable called a *handle*.

The first handle that you require in the Stateflow API is a handle to the Root object, the parent object of all objects in the Stateflow API. In the following command, the function `sfrroot` returns a handle to the Root object:

```
rt = sfrroot
```

Once you have a handle to the Root object, you can find a handle to the Model object corresponding to the Simulink model you want to work with. Once you have a handle to a Model object, you can find a handle to a Chart object for the chart you want to edit. Later on, when you create objects or find existing

objects in a Stateflow chart, you receive a handle to the object that allows you to manipulate the actual object in Stateflow.

You are introduced to obtaining handles to Stateflow API objects and using them to create and alter Stateflow diagrams in “Quick Start for the Stateflow API” on page 1-9.

Using API Object Properties and Methods

Once you obtain handles to Stateflow API objects, you can manipulate the Stateflow objects that they represent through the properties and methods that each Stateflow API object possesses. You access the properties and methods of an object through a handle to the object.

API properties correspond to values that you normally set for an object through the user interface of the Stateflow diagram editor. For example, you can change the position of a transition by changing the `Position` property of the `Transition` object that represents the transition. In the Stateflow diagram editor you can click-drag the source, end, or midpoint of a transition to change its position.

API methods are similar to functions for creating, finding, changing, or deleting the objects they belong to. They provide services that are normally provided by the Stateflow diagram editor. For example, you can delete a transition in the Stateflow diagram editor by calling the `delete` method of the `Transition` object that represents the transition. Deleting a transition in the diagram editor is normally done by selecting a transition and pressing the **Delete** key.

Stateflow API objects have some common properties and methods. For example, all API objects have an `Id` and a `Description` property. All API objects have a `get` and a `set` method for viewing or changing the properties of an object, respectively. Most API objects also have a `delete` method. Methods held in common among all Stateflow objects are listed in the reference section “All Object Methods” on page 3-11.

Each API object also has properties and methods unique to its type. For example, a `State` object has a `Position` property containing the spatial coordinates for the state it represents in the chart editor. A `Data` object, however, has no `Position` property.

API References to Properties and Methods

When you need to know what property's value to change or what method to call to effect a change in a Stateflow chart, you can consult the following references for specific information about an individual Stateflow API property or method:

- **API Properties and Methods by Use** — This reference section lists the properties and methods of the Stateflow API organized according to their type of use in Stateflow.

For example, if you want to use the API to change the font color or style for a state, see the section “Drawing Properties” on page 2-12.

- **API Properties and Methods by Object** — This reference section lists the properties and methods of the Stateflow API by their owning objects.

For example, if you want to change a transition with a transition property in the API, see the section “Transition Properties” on page 3-55.

- **API Methods Reference** — This reference section contains individual references for each method in the Stateflow API.

These references are ordered alphabetically and provide information on the objects that they belong to, the syntax for calling them, a description of what they do, and information on their argument and return values, along with examples.

Quick Start for the Stateflow API

This section helps you create a single Stateflow chart and its member objects using the Stateflow API. It reflects the major steps in using the Stateflow API to create a Stateflow chart:

- 1 “Create a New Model and Chart” on page 1-9 — Teaches you by example to create a new empty Stateflow diagram in its own new model.
- 2 “Access the Model Object” on page 1-9 — Tells you how to access the Stateflow Model object, which you need to access before you can access a Stateflow Chart object.
- 3 “Access the Chart Object” on page 1-10 — Tells you how to access the Chart object so that you can begin creating Stateflow objects in the chart you created.
- 4 “Create New Objects in the Chart” on page 1-11 — Gives many example commands for creating new objects in a new Stateflow chart.

Create a New Model and Chart

Create a new model by itself in MATLAB with the following steps:

- 1 Close down all models in Simulink.
- 2 Use the function `sfnew` to create a new chart.

The `sfnew` function creates a new untitled Simulink model with a new Stateflow chart in it. Do not open the Stateflow chart.

You now have only one Simulink model in memory. You are now ready to access the API Model object that represents the model itself.

Access the Model Object

In the Stateflow API, each model you create or load into memory is represented by an object of type `Model`. Before accessing the Stateflow chart you created in the previous section, you must first connect to its `Model` object. However, in the Stateflow API, all `Model` objects are contained by the

Stateflow API Root object, so you must use the Root object returned by the function `sfroot` to access a Model object:

- 1 Use the following command to obtain a handle to the Root object:

```
rt = sfroot
```

- 2 Use the handle to the Root object, `rt`, to find the Model object representing your new untitled Simulink model and assign it a handle, `m` in the following command:

```
m = rt.find('-isa','Simulink.BlockDiagram')
```

If, instead of one model, there are several models open, this command returns an array of different Model objects that you could access through indexing (`m(1)`,`m(2)`,...). You can identify a specific Model object using the properties of each model, particularly the `Name` property, which is the name of the model. For example, you can use the `Name` property to find a Model object named **myModel** with the following command:

```
m = rt.find('-isa', 'Simulink.BlockDiagram', '-and',  
           'Name', 'myModel')
```

However, since you now have only one model loaded, the object handle `m` in the command for step 2 returns the Model object for the model that you just created. You are now ready to use `m` to access the empty Stateflow chart so that you can start filling it with Stateflow objects.

Access the Chart Object

In “Access the Model Object” on page 1-9, you accessed the Model object containing your new chart to return a handle to the Model object for your new model, `m`. Perform the following steps to access the new Stateflow chart:

- 1 Access the new Chart object and assign it to the workspace variable `chart` as follows:

```
chart = m.find('-isa','Stateflow.Chart')
```

In the preceding command, the `find` method of the Model object `m` returns an array of all charts belonging to that model. Because you created only one

chart, the result of this command is the chart you created. If you created several charts, the `find` method returns an array of charts that you could access through indexing (for example, `chart(1)`, `chart(2)`, and so on).

You can also use standard function notation instead of dot notation for the preceding command. In this case, the first argument is the Model object handle, `m`.

```
chart = find(m, '-isa','Chart')
```

- 2 Open the Stateflow chart with the following API command:

```
chart.view
```

The preceding command calls the `view` method of the Chart object whose handle is `chart`. This displays the specified chart in the Stateflow diagram editor. You should now have an empty Stateflow chart in front of you. Other Stateflow API objects have `view` methods as well.

Create New Objects in the Chart

In the previous section, you created a handle to the new Chart object, `chart`. Continue by creating new objects for your chart using the following steps:

- 1 Create a new state in the Chart object `chart` with the following command:

```
sA = Stateflow.State(chart)
```

This command is a Stateflow API constructor for a new state in which `Stateflow.State` is the object type for a state, `chart` is a workspace variable containing a handle to the parent chart of the new state, and `sA` is a workspace variable to receive the returned handle to the new state.

An empty state now appears in the upper left-hand corner of the diagram editor.

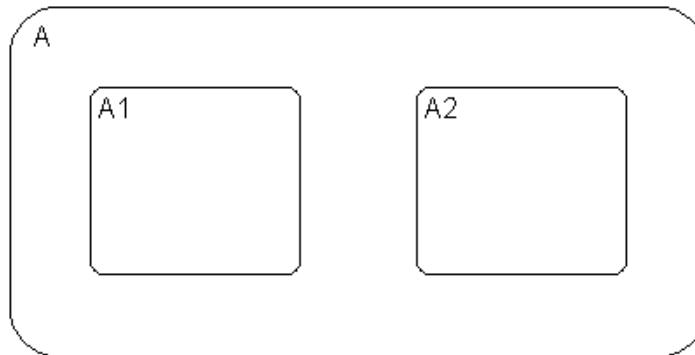
- 2 Use the `chart.view` command to bring the chart diagram editor to the foreground for viewing.
- 3 Assign a name and position to the new state by assigning values to the new State object's properties as follows:

```
sA.Name = 'A'  
sA.Position = [50 50 310 200]
```

- 4** Create new states A1 and A2 inside state A and assign them properties with the following commands:

```
sA1 = Stateflow.State(chart)  
sA1.Name = 'A1'  
sA1.Position = [80 120 90 60]  
sA2 = Stateflow.State(chart)  
sA2.Name = 'A2'  
sA2.Position = [240 120 90 60]
```

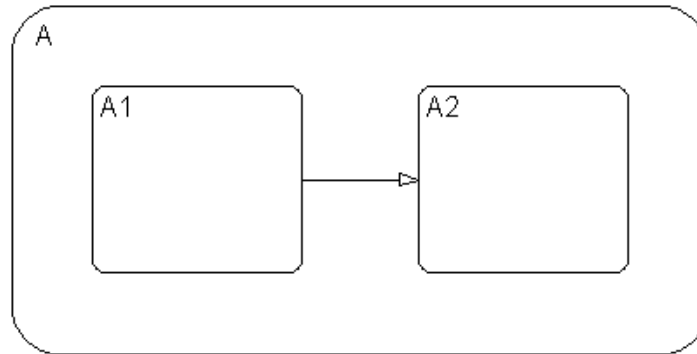
These commands create and use the workspace variables sA, sA1, and sA2 as handles to the new states, which now have the following appearance:



- 5** Create a transition from the 3 o'clock position (right side) of state A1 to the 9 o'clock position (left side) of state A2 with the following commands:

```
tA1A2 = Stateflow.Transition(chart)  
tA1A2.Source = sA1  
tA1A2.Destination = sA2  
tA1A2.SourceOClock = 3.  
tA1A2.DestinationOClock = 9.
```

A transition now appears as shown:



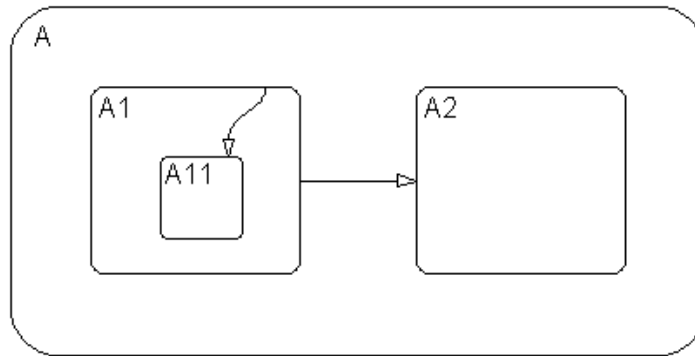
- 6** Draw, name, and position a new state A11 inside A1 with the following commands:

```
sA11 = Stateflow.State(chart)
sA11.Name = 'A11'
sA11.Position = [90 130 35 35]
```

- 7** Draw an inner transition from the 1 o'clock position of state A1 to the 1 o'clock position of state A11 with the following commands:

```
tA1A11 = Stateflow.Transition(chart)
tA1A11.Source = sA1
tA1A11.Destination = sA11
tA1A11.SourceOClock = 1.
tA1A11.DestinationOClock = 1.
```

Your Stateflow diagram now has the following appearance:



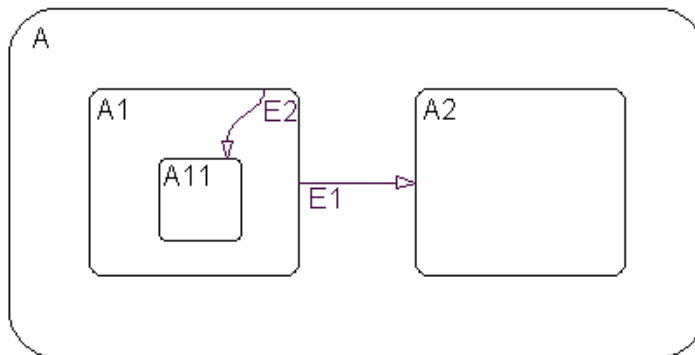
- 8 Add the label E1 to the transition from state A1 to state A2 with the following command:

```
tA1A2.LabelString = 'E1'
```

- 9 Add the label E2 to the transition from state A1 to state A11 with the following command:

```
tA1A11.LabelString = 'E2'
```

The Stateflow diagram now has the following appearance:



Both the state and transition labels in our example are simple one-line labels. To enter more complex multiline labels, see “Entering Multiline Labels” on page 1-39. Labels for transitions also have a `LabelPosition` property that you can use to move the labels to better locations.

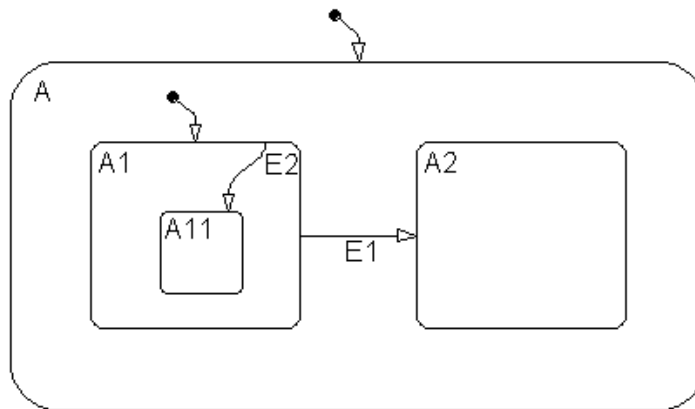
- 10** Use the following commands to move the label for the transition from A1 to A2 to the right by 15 pixels:

```
pos = tA1A2.LabelPosition
pos(1) = pos(1)+15
tA1A2.LabelPosition = pos
```

- 11** Use the following commands to finish your new chart diagram by adding default transitions to states A and A1 with source points 20 pixels above and 10 pixels to the left of the top midpoint of each state:

```
dtA = Stateflow.Transition(chart)
dtA.Destination = sA
dtA.DestinationOClock = 0
xsource = sA.Position(1)+sA.Position(3)/2-10
ysource = sA.Position(2)-20
dtA.SourceEndPoint = [xsource ysource]
dtA1 = Stateflow.Transition(chart)
dtA1.Destination = sA1
dtA1.DestinationOClock = 0
xsource = sA1.Position(1)+sA1.Position(3)/2-10
ysource = sA1.Position(2)-20
dtA1.SourceEndPoint = [xsource ysource]
```

You now have the following finished Stateflow diagram:



- 12** Save the Simulink model with its new Stateflow chart to the working directory as `myModel.mdl` with the following command:

```
sfsave(m.Name, 'myModel')
```

Notice that the preceding command uses the `Name` property of the `Model` object `m` for saving the model under a new name.

You are now finished with “Quick Start for the Stateflow API” on page 1-9. You can continue with “Accessing the Properties and Methods of Objects” on page 1-17, or you can go to “Creating a MATLAB Script of API Commands” on page 1-43 to see how to create a script of the API commands you used in this Quick Start section.

Accessing the Properties and Methods of Objects

All Stateflow API commands access the properties and methods of Stateflow objects. Before you start creating a new Stateflow diagram or changing an existing one, you must learn how to access the properties and methods of objects.

This section describes the conventions used in naming the properties and methods of Stateflow API objects and the rules for using them in commands:

- “Naming Conventions for Properties and Methods” on page 1-17 — Gives you a look at the conventions followed in naming properties and methods.
- “Using Dot Notation with Properties and Methods” on page 1-17 — Shows you how to use dot notation to access the value of an object’s property or call the method of an object.
- “Using Function Notation with Methods” on page 1-18 — Shows you how to use standard function notation to call the methods of objects.

Naming Conventions for Properties and Methods

By convention, all properties begin with a capital letter, for example, the property `Name`. However, if a property consists of concatenated words, the words following the first word are capitalized, for example, the property `LabelString`. The same naming convention applies to methods, with the exception that a method name must begin with a letter in lowercase; for example, the method `find`.

Using Dot Notation with Properties and Methods

You can access the properties and methods of an object by adding a period (.) and the name of the property or method to the end of an object’s handle variable. For example, the following command returns the `Type` property for a State object represented by the handle `s`:

```
stype = s.Type
```

The following command calls the `dialog` method of the State object `s` to open a properties dialog for that state:

```
s.dialog
```

Nesting Dot Notation

You can nest smaller dot expressions in larger dot expressions of properties. For example, the `Chart` property of a `State` object returns the `Chart` object of the containing chart. Therefore, the expression `s.Chart.Name` returns the name of the chart containing the `State` whose object is `s`.

Methods can also be nested in dot expressions. For example, if the `State` object `sA1` represents state `A1` in the final `Stateflow` chart at the end of “Create New Objects in the Chart” on page 1-11, the following command returns the string label for state `A1`’s inner transition to its state `A11`.

```
label = sA1.innerTransitionsOf.LabelString
```

The preceding command uses the `LabelString` property of a `Transition` object and the `innerTransitions` method for a `State` object. It works as shown only because state `A1` has one inner transition. If state `A1` has more than one transition, you must first find all the inner transitions and then use an array index to access each one, as shown below:

```
innerTransitions = sA1.innerTransitionsOf
label1 = innerTransitions(1).LabelString
label2 = innerTransitions(2).LabelString
and so on...
```

Using Function Notation with Methods

As an alternative to dot notation, you can access object methods with standard function call notation. For example, you can use the `get` method to access the `Name` property of a `Chart` object, `ch`, through one of the following commands:

```
name = ch.get('Name')
name = get(ch, 'Name')
```

If you have array arguments to methods you call, use function notation. The following example returns a vector of strings with the names of each chart in the array of `Chart` objects `chartArray`:

```
names = get(chartArray, 'Name')
```

If, instead, you attempt to use the `get` command with the following dot notation, an error results:

```
names = chartArray.get('Name')
```

Displaying Properties and Methods

The Stateflow API provides a few methods that help you see the properties and methods available for each object. These are described in the following sections:

- “Displaying Properties” on page 1-20 — Shows you how to display a list of the properties for a given object.
- “Displaying the Names of Methods” on page 1-20 — Shows you how to display a list of the methods for a given object.
- “Displaying Property Subproperties” on page 1-21 — Shows you how to display the subproperties of some properties.
- “Displaying Enumerated Values for Properties” on page 1-22 — Shows you how to display lists of acceptable values for properties that require enumerated values.

Displaying Properties

To access the names of all properties for any particular object, use the `get` method. For example, if the object `s` is a State object, enter the following command to list the properties and current values for any State object:

```
get(s)
```

To get a quick description for each property, use the `help` method. For example, if `s` is a State object, the following command returns a list of State object properties, each with a small accompanying description:

```
s.help
```

Note Some properties do not have a description, because their names are considered descriptive enough.

Displaying the Names of Methods

Use the `methods` method to list the methods for any object. For example, if the object `t` is a handle to a Transition object, use the following command to list the methods for any Transition object:

```
t.methods
```

Note The following internal methods might be displayed by the methods method for an object, but is not applicable to Stateflow use, and is not documented: areChildrenOrdered, getChildren, getDialogInterface, getDialogSchema, getDisplayClass, getDisplayIcon, getDisplayLabel, getFullName, getHierarchicalChildren, getPreferredProperties, isHierarchical, isLibrary, isLinked, isMasked.

Use a combination of the `get` method and the `classhandle` method to list only the names of the methods for an object. For example, list the names of the methods for the Transition object `t` with the following command:

```
get(t.classhandle.Methods, 'Name')
```

Displaying Property Subproperties

Some properties are objects that have properties referred to as subproperties. For example, when you invoke the command `get(ch)` on a chart object, `ch`, the output displays the following for the `StateFont` property:

```
StateFont: [1x1 Font]
```

This value indicates that the `StateFont` property of a state has subproperties. To view the subproperties of `StateFont`, enter the command `get(ch.StateFont.get)` to receive something like the following:

```
Name: Helvetica '  
Size: 12  
Weight: 'NORMAL '  
Angle: 'NORMAL '
```

From this list it is clearly seen that `Name`, `Size`, `Weight`, and `Angle` are subproperties of the property `StateFont`. In the API property references for this guide (see “API References to Properties and Methods” on page 1-8), these properties are listed by their full names: `Statefont.Name`, `Statefont.Size`, and so on.

Displaying Enumerated Values for Properties

Many of the properties for API objects can only be set to one of a group of enumerated strings. You can identify these properties from the API references for properties and methods (see “API References to Properties and Methods” on page 1-8). Generally, in the display for properties generated by the `get` command (see “Displaying Properties” on page 1-20) the values for these properties appear as strings of capital letters.

You display a list of acceptable strings for a property requiring enumerated values using the `set` method. For example, if `ch` is a handle to a `Chart` object, you can display the allowed enumerated values for the `Decomposition` property of that chart with the following command:

```
set (ch, 'Decomposition')
```


Creating and Destroying API Objects

You create (construct), parent (contain), and delete (destroy) objects in Stateflow through constructor methods in the Stateflow API. For all but the Editor and Clipboard objects, creating objects establishes a handle to them that you can use for accessing their properties and methods to make modifications to Stateflow diagrams. See the following sections:

- “Creating Stateflow Objects” on page 1-23 — Shows you how to create and connect to Stateflow objects in the Stateflow API.
- “Establishing an Object’s Parent (Container)” on page 1-25 — Shows you how to control the containment of graphical objects in the Stateflow diagram editor.

Stateflow objects are contained (parented) by other objects as defined in the Stateflow hierarchy of objects (see “Stateflow API Object Hierarchy” on page 1-4). You control containment of nongraphical objects in the Stateflow Explorer.

- “Destroying Stateflow Objects” on page 1-26 — Shows you how to delete objects in the Stateflow API.

Creating Stateflow Objects

You create a Stateflow object as the child of a parent object through API constructor methods. Each Stateflow object type has its own constructor method. See “Constructor Methods” on page 3-7 for a list of the valid constructor methods.

Use the following process to create Stateflow objects with the Stateflow API:

- 1 Access the parent object to obtain a handle to it.

When you first begin populating a model or chart, this means that you must get a handle to the Stateflow Model object or a particular Chart object. See “Access the Model Object” on page 1-9 and “Access the Chart Object” on page 1-10.

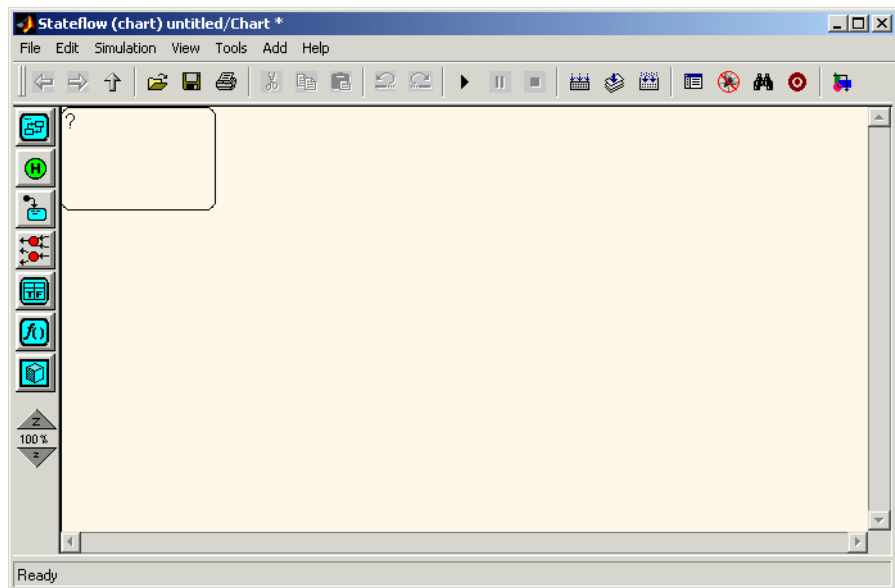
See also “Accessing Existing Stateflow Objects” on page 1-27 for a more general means of accessing (getting an object handle to) an existing Stateflow object.

- 2 Call the appropriate constructor method for the creation of the object using the parent (containing) object as an argument.

For example, the following command creates and returns a handle *s* to a new state object in the chart object with the handle *ch*:

```
s = Stateflow.State(ch)
```

By default, the newly created state from the preceding command appears in the upper-left corner of the Stateflow chart (at *x-y* coordinates 0,0).



The constructor returns a handle to an API object for the newly created Stateflow object. Use this handle to display or change the object through its properties and methods.

- 3 Use the object handle returned by the constructor to make changes to the object in Stateflow.

For example, you can now use the handle *s* to set its name (Name property) and position (Position property). You can also connect it to other states or junctions by creating a Transition object and setting its Source or

Destination property to `s`. See “Create New Objects in the Chart” on page 1-11 for examples.

Use the preceding process to create all Stateflow objects in your chart. “Create New Objects in the Chart” on page 1-11 gives examples for creating states and transitions. Objects of other types are created just as easily. For example, the following command creates and returns a handle (`d1`) for a new Data object belonging to the state A (handle `sA`):

```
d1 = Stateflow.Data(sA)
```

Note Currently, there is no constructor for a Stateflow chart. To create a chart with the Stateflow API you must use the `sfnew` function.

Establishing an Object’s Parent (Container)

As discussed in the previous section, “Creating Stateflow Objects” on page 1-23, the Stateflow API constructor establishes the parent for a newly created object by taking a handle for the parent object as an argument to the constructor.

Graphical Object Parentage

When graphical objects (states, boxes, notes, functions, transitions, junctions) are created, they appear completely inside their containing parent object. In the diagram editor, graphical containment is a necessary and sufficient condition for establishing the containing parent.

Repositioning a graphical object through its `Position` property can change an object’s parent or cause an undefined parent error condition. Parsing a chart in which the edges of one object overlap with another produces an undefined parent error condition that cannot be resolved by the Stateflow parser. You can check for this condition by examining the value of the `BadIntersection` property of a `Chart` object, which equals 1 if the edges of a graphical object overlap with other objects. You need to set the size and position of objects so that they are clearly positioned and separate from other objects.

Nongraphical Object Parentage

When nongraphical objects (data, events, and targets) are created, they appear in the Stateflow Explorer at the hierarchical level of their owning object. Containment for nongraphical objects is established through the Stateflow Explorer only. See the section “Using the Model Explorer with Stateflow Objects” in the Stateflow and Stateflow Coder User’s Guide documentation.

Destroying Stateflow Objects

Each Stateflow object of type State, Box, Function, Note, Transition, Junction, Event, Data, or Target has a destructor method named `delete`. In the following example, a State object, `s`, is deleted:

```
s.delete
```

The preceding command is equivalent to performing a mouse select and keyboard delete operation in the Stateflow diagram editor. Upon deletion, graphical Stateflow objects are sent to the clipboard; nongraphical objects, such as data and events, are completely deleted. The workspace variable `s` still exists but is no longer a handle to the deleted state.

Accessing Existing Stateflow Objects

Creating Stateflow objects through the Stateflow API gives you an immediate handle to the newly created objects (see “Creating Stateflow Objects” on page 1-23). You can also connect to Stateflow objects that already exist for which you have no current API handle.

The following sections describe how you use the Stateflow API to find and access existing objects in Stateflow charts:

- “Finding Objects” on page 1-27 — Shows you how to use the `find` method, which is the most powerful and versatile method for locating objects in the Stateflow API.
- “Finding Objects at Different Levels of Containment” on page 1-28 — Shows you how to use the `find` method for finding objects at different levels of containment.
- “Retrieving Recently Selected Objects” on page 1-30 — Shows you how to use the `sfco` function to retrieve handles to the most recently selected objects in a Stateflow chart.
- “Getting and Setting the Properties of Objects” on page 1-31 — Shows you how to use the Stateflow API to access the properties of the existing objects you find in Stateflow charts.

Finding Objects

There are several object methods that you use to traverse the Stateflow hierarchy to locate existing objects. Chief among these is the versatile `find` method.

With the `find` method, you specify what to search for by specifying combinations of the following types of information:

- The type of object to find
- A property name for the object to find and its value

The following example searches through Model object `m` to return every State object with the name `'On'`.

```
onState = m.find('-isa','Stateflow.State','-and','Name','On')
```

If a `find` command finds more than one object that meets its specifications, it returns an array of qualifying objects. The following example returns an array of all charts in your model:

```
chartArray = m.find('-isa','Stateflow.Chart')
```

Use array indexing to access individual properties and methods for a chart. For example, if the preceding command returns three Stateflow charts, the following command returns the Name property of the second chart found:

```
name2 = chartArray(2).Name
```

By default, the `find` command finds objects at all depths of containment within an object. This includes the zeroth level of containment, which is the searched object itself. For example, if state A, which is represented by State object `sA`, contains two states, A1 and A2, and you specify a `find` command that finds all the states in A as follows,

```
states= sA.find( '-isa','Stateflow.State')
```

The preceding command finds three states: A, A1, and A2.

Note Be careful when specifying the objects you want to find with the `find` method for a Root or Model object. Using the `find` method for these objects can return Simulink objects matching the arguments you specify. For example, if `rt` is a handle to the Root object, the command `find('Name','ABC')` might return a Simulink subsystem or block named ABC. See the reference for the `find` method for a full description of the method and its parameters.

Finding Objects at Different Levels of Containment

Once you find a particular object in a Stateflow diagram by its name or another property, you might want to find the objects that it contains (children), or the object that contains it (parent). To find child objects, use the `find` method. To find a parent object, use the `method up`.

Finding Child Objects

The `find` method finds objects at the depth of containment within an object that you specify. If you want to limit the containment search depth with the `find` command, use the `depth` switch. For example, to find all the objects in State object `sA` at the first level of containment, use the following command:

```
objArray = sA.find('-depth', 1)
```

Don't forget, however, that the `find` command always includes the zeroth level of containment, which is the object itself. So, the preceding command also includes state `A` in the list of objects found. However, you can exclude state `A` from the vector of objects in `objArray` with the MATLAB function `setdiff` as follows:

```
objArray = setdiff(objArray, sA)
```

The following command returns a collection of all junctions at the first level of containment inside the state `A` that is represented by State object `sA`:

```
juncArray = sA.find('-isa','Stateflow.Junction','-depth',1)
```

The following command returns an array of all transitions inside state `A` at all levels of containment:

```
transArray = sA.find('-isa','Stateflow.Transition')
```

Finding a Parent Object

The `up` method finds the parent container object of any given object. In the example Stateflow diagram in “Create New Objects in the Chart” on page 1-11, state `A` contains states `A1` and `A2`. Also, state `A1` contains state `A11`. In the example, `sA11` is a handle to the state `A11`. This means that

```
>> pA11 = sA11.up;  
>> pA11.Name
```

```
ans =
```

```
A1
```

returns a handle pA11 to the state A1, the parent of state A11, and

```
>> ppA11 = pA11.up;
>> ppA11.Name
```

```
ans =
```

```
A
```

returns a handle ppA11 to the state A, the parent of state A1.

Retrieving Recently Selected Objects

You can retrieve the most recently selected objects in a Stateflow diagram by using the `sfgco` function. This function returns object handles or a vector of handles depending on the following conditions:

If	Then <code>sfgco</code> returns
There are no open diagrams	An empty matrix
There is no selection list	Handle of the diagram most recently clicked
You select one object in a diagram	Handle to the selected object
You select multiple objects in a diagram	Vector of handles for the selected objects
You select objects in multiple diagrams	Handles of the most recently selected objects in the most recently selected diagram

For example, suppose you run the `sf_boiler` demo and open the Stateflow chart called `Bang Bang Controller`. If you select the `Off` state in the chart, `sfgco` returns

```
ans =
```

```
Path: [1x37 char]
Id: 31
Machine: [1x1 Stateflow.Machine]
```



```

        Name: 'Off'
    Description: ''
    LabelString: [1x56 char]
        FontSize: 12
        ArrowSize: 8
        TestPoint: 0
        Chart: [1x1 Stateflow.Chart]
    BadIntersection: 0
        Subviewer: [1x1 Stateflow.Chart]
        Document: ''
        Tag: []
    RequirementInfo: ''
    ExecutionOrder: 0
    HasOutputData: 0
        Position: [31.7440 40.9730 214.1620 59.5660]
    Decomposition: 'EXCLUSIVE_OR'
        Type: 'OR'
    IsSubchart: 0
    IsGrouped: 0
    Debug: [1x1 Stateflow.StateDebug]

```

If you then hold down the Shift key to select the three transitions in state Off, `sfgco` returns

```

ans =

    Stateflow.Transition: 3-by-1

```

Getting and Setting the Properties of Objects

Once you obtain a particular object, you can access its properties directly or through the `get` method. For example, you obtain the description for a State object `s` with one of the following commands:

- `od = s.Description`
- `od = s.get ('Description')`
- `od = get (s, 'Description')`

You change the properties of an object directly or through the `set` method. For example, you change the description of the `State` object `s` with one of the following commands:

- `s.Description = 'This is the On state.'`
- `s.set ('Description', 'This is the On state.')`
- `set (s, 'Description', 'This is the On state.')`

Copying Objects

You can use the clipboard (accessed through the Stateflow API Clipboard object) to copy one object to another. See the following topics to learn how to copy objects from one object to another using the Clipboard object and its methods:

- “Accessing the Clipboard Object” on page 1-33 — Shows you how to use the Clipboard object and its two methods, `copy` and `pasteTo`, to copy objects from one object to another.
- “copy Method Limitations” on page 1-33 — Describes the features and limitations of the `copy` method that determine how you can copy objects in the Stateflow API.
- “Copying by Grouping (Recommended)” on page 1-34 — Gives you the most powerful method for copying using the `IsGrouped` property of State objects.
- “Copying Objects Individually” on page 1-35 — Tells you how you can also copy objects individually from one object to another.

Accessing the Clipboard Object

The Clipboard object (there is only one) provides an interface to the clipboard used in copying Stateflow objects. You cannot directly create or destroy the Clipboard object as you do other Stateflow API objects. However, you can attach a handle to it to use its properties and methods to copy Stateflow objects.

You create a handle to the Clipboard object by using the `sfclipboard` function as follows:

```
cb = sfclipboard
```

Clipboard objects have two methods, `copy` and `pasteTo`, that together provide the functionality to copy objects from one object to another. The `copy` method copies the specified objects to the Clipboard object, and the `pasteTo` method pastes the contents of the clipboard to a new container.

copy Method Limitations

The `copy` method is subject to the following limitations for all objects:

- The objects copied must be either *all* graphical (states, boxes, functions, transitions, junctions) or *all* nongraphical (data, events, targets).

You cannot copy a mixture of graphical and nongraphical objects to the clipboard in the same copy operation.

- To maintain the transition connections and containment relationships between copied objects, you must copy the entire array of related objects.

All related objects must be part of the array of objects copied to the clipboard. For example, if you attempt to copy two states connected by a transition to another container, you can only accomplish this by copying both the states and the transition at the same time. That is, you must do a single copy of a single array containing both the states and the transition that connects them.

If you copy a grouped state to the clipboard, not only are all the objects contained in the state copied, but all the relations among the objects in the grouped state are copied as well. Thus, copying by grouping is a recommended procedure. See “Copying by Grouping (Recommended)” on page 1-34.

Copying Graphical Objects

The copy method is subject to the following limitations for all graphical objects:

- Copying graphical objects also copies the Data, Event, and Target objects that the graphical objects contain.
- If all copied objects are graphical, they must all be seen in the same subviewer.

This means that all graphical objects copied in a single copy command must reside in the same chart or subchart.

Copying by Grouping (Recommended)

Copying a grouped state in Stateflow copies not only the state but all of its contents. By grouping a state before you copy it, you can copy it and all of its contained objects at all levels of containment with the Stateflow API. This is the simplest way of copying objects and should be used whenever possible.

You use the Boolean `IsGrouped` property for a state to group that state. If you set the `IsGrouped` property for a state to a value of true (=1), it is grouped. If you set `IsGrouped` to a value of false (=0), the state is not grouped.

The following example procedure copies state A to the chart X through grouping. In this example, assume that you already have a handle to state A and chart X through the MATLAB variables `sA` and `chX`, respectively:

- 1 If the state to copy is not already grouped, group it along with all its contents by setting the `IsGrouped` property for that state to true (=1).

```
prevGrouping = sA.IsGrouped
if (prevGrouping == 0)
    sA.IsGrouped = 1
end
```

- 2 Get a handle to the Clipboard object.

```
cb = sfclipboard
```

- 3 Copy the grouped state to the clipboard using the Clipboard object.

```
cb.copy(sA)
```

- 4 Paste the grouped object to its new container.

```
cb.pasteTo(chX)
```

- 5 Set the copied state and its source state to its previous `IsGrouped` property value.

```
sA.IsGrouped = prevGrouping
sNew = chX.find('-isa',Stateflow.State','-and','Name',sA.Name)
sNew.IsGrouped = prevGrouping
```

Copying Objects Individually

You can copy specific objects from one object to another. However, in order to preserve transition connections and containment relations between objects, you must copy all the connected objects at once. To accomplish this, use the general technique of appending objects from successive finds in MATLAB

to a growing array of objects before copying the finished object array to the clipboard.

Using the example of the Stateflow chart at the end of “Create New Objects in the Chart” on page 1-11, you can copy states A1, A2, and the transition connecting them to another state, B, with the following API commands, where, sA and sB are workspace handles to states A and B, respectively.

```
objArrayS = sA.find('-isa','Stateflow.State','-depth',1)
objArrayT = sA.find('-isa','Stateflow.Transition','-depth',1)
sourceObjs = {objArrayS ; objArrayT}
cb = sfclipboard
cb.copy(sourceObjs)
destObjs = cb.pasteTo(sB)
```

You can also accomplish the job of constructing the copy array through a complex `find` command. This might be adequate in certain situations. However, this approach might require you to formulate a very complex command. By contrast, the technique of appending found objects to an array relies on simpler `find` commands.

You can also copy nongraphical data, events, and target objects individually. However, since there is no way for these objects to find their new owners, you must ensure that each of these objects is separately copied to its appropriate owner object.

Note Copying objects individually is more difficult than copying grouped objects. This is why copying objects by grouping is recommended. See “Copying by Grouping (Recommended)” on page 1-34.

Using the Editor Object

The Editor object provides access to the purely graphical properties and methods of Chart objects. Each Chart object has its own Editor object. See the following topics to learn how to use the Editor object and its methods to change the display of the Stateflow diagram editor for a chart:

- “Accessing the Editor Object” on page 1-37 — Tells you how to connect to the Editor object.
- “Changing the Stateflow Display” on page 1-37 — Teaches you how to use the methods of the Editor object to change the display of the Stateflow diagram editor.

Accessing the Editor Object

You cannot directly create or destroy the Editor and Clipboard objects as you do other Stateflow API objects. However, you can attach a handle to them to use their properties and methods for modifications to Stateflow diagrams.

When you create a chart, an Editor object is automatically created for it. If `ch` is a workspace handle to a chart, you create a handle to the Editor object for that chart with the following command:

```
ed = ch.Editor
```

Changing the Stateflow Display

Use the handle `ed` from the preceding example to access the Editor object properties and methods. For example, the following command calls the `zoomIn` method to zoom in the chart by a factor of 20%:

```
ed.zoomIn
```

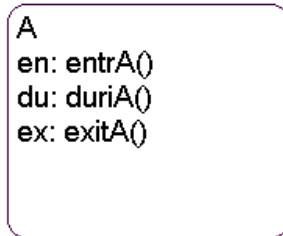
Or, you can simply set the `ZoomFactor` property of this chart’s editor to an absolute zoom factor of 150%:

```
ed.ZoomFactor = 1.5
```

You can also use a chart's Editor object to change the window position of the diagram editor. For a reference to all the Editor object's properties and methods, see "Editor Properties" on page 3-8 and "Editor Methods" on page 3-9.

Entering Multiline Labels

In the examples shown thus far of entering labels for states and transitions, only a simple one-line expression has been used. The following figure shows state A with a multiline label.



There are two ways to enter multiline labels for both states and transitions. In the following examples, `sA` is a workspace variable handle to the State object in the Stateflow API for state A:

- Use the MATLAB function `sprintf`:

```
str = sprintf('A\nen: entrA()\ndu: durIA()\nex: exitA()')
sA.LabelString = str
```

In this example, carriage returns are inserted into a string expression with the escape sequence `\n`.

- Use a concatenated string expression:

```
str = ['A',10,'entr: entrA()',10,'du: durIA()',
      10,'ex: exitA()']
sA.LabelString = str
```

In this example, carriage returns are inserted into a concatenated string expression with the ASCII equivalent of a carriage return, the integer 10.

Creating Default Transitions

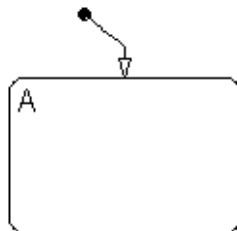
Default transitions differ from normal transitions in not having a source object. You can create a default transition with the following process:

- 1 Create a transition.
- 2 Attach the destination end of the transition to an object.
- 3 Position the source endpoint for the transition.

If you assume that the workspace variable `sA` is a handle to state A, the following commands create a default transition and position its source 25 pixels above and 15 pixels to the left of the top midpoint of state A:

```
dt = Stateflow.Transition(sA)
dt.Destination = sA
dt.DestinationOClock = 0
xsource = sA.Position(1)+sA.Position(3)/2-15
ysource = sA.Position(2)-25
dt.SourceEndPoint = [xsource ysource]
```

The created default transition has the following appearance:

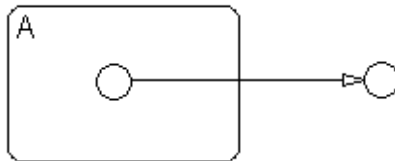


This method is also used for adding the default transitions toward the end of the example Stateflow diagram constructed in “Create New Objects in the Chart” on page 1-11.

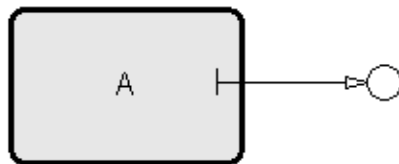
Making Supertransitions

The Stateflow API does not currently support the direct creation of supertransitions. Supertransitions are transitions between a state or junction in a top-level chart and a state or junction in one of its subcharts, or between states residing in different subcharts at the same or different levels in a diagram. For a better understanding of supertransitions, see “What Is a Supertransition?” in the Stateflow and Stateflow Coder User’s Guide documentation.

Stateflow does provide a workaround for indirectly creating supertransitions. In the following example, a supertransition is desired from a junction inside a subchart to a junction outside the subchart. In order to use the Stateflow API to create the supertransition in this example, first use the API to create the superstate as an ordinary state with a transition between its contained junction and a junction outside it.



Now set the `IsSubchart` property of the state A to true (=1).



This makes state A a subchart, and the transition between the junctions is now a supertransition.

You can also connect supertransitions to and from objects in an existing subchart (state A, for example) with the following procedure:

- 1** Save the original position of subchart A to a temporary workspace variable.

For example, if the subchart A has the API handle `sA`, store its position with the following command:

```
sA_pos = sA.Position
```

- 2** Convert subchart A to a state by setting its `IsSubchart` property to false (=0).

```
sA.IsSubchart = 0
```

- 3** Ungroup state A by setting its `IsGrouped` property to false (=0).

```
sA.IsGrouped = 0
```

When convert a subchart a normal state, it stays grouped to hide the contents of the subchart. When you ungroup the subchart, it might resize to display its contents.

- 4** Make the necessary transition connections.

See “Create New Objects in the Chart” on page 1-11 for an example of creating a transition.

- 5** Set the `IsSubchart` property of state A back to true (=1).

For example, `sA.IsSubchart = 1`

- 6** Assign subchart A its original position.

```
sA.Position = sA_pos
```

When you convert a subchart to a normal state and ungroup it, it might resize to accommodate the size of its contents. The first step of this procedure stores the original position of the subchart so that this can be restored after the transition connection is made.

Creating a MATLAB Script of API Commands

In “Quick Start for the Stateflow API” on page 1-9, you created and saved a new model through a series of Stateflow API commands. You can include the same API commands in the following MATLAB script. This script allows you to quickly recreate the same model with the single command `makeMyModel`.

```
function makeMyModel
% Get all previous models loaded

rt = sfroot;
prev_models = rt.find('-isa','Simulink.BlockDiagram');

% Create new model, and get current models

sfnew;
curr_models = rt.find('-isa','Simulink.BlockDiagram');

% New model is current models - previous models

m = setdiff(curr_models, prev_models);

% Get chart in new model

chart = m.find('-isa', 'Stateflow.Chart');

% Create state A in chart

sA = Stateflow.State(chart);
sA.Name = 'A';
sA.Position = [45 45 300 150];

% Create state A1 inside of state A

sA1 = Stateflow.State(chart);
sA1.Name = 'A1';
sA1.Position = [80 80 90 80];

% Create state A2 inside of state A
```

```
sA2 = Stateflow.State(chart);
sA2.Name = 'A2';
sA2.Position = [220 80 90 80];

% Create a transition from A1 to A2

tA1A2 = Stateflow.Transition(chart);
tA1A2.Source = sA1;
tA1A2.Destination = sA2;
tA1A2.SourceOClock = 3.;
tA1A2.DestinationOClock = 9.;

% Create state A11 inside of state A1

sA11 = Stateflow.State(chart);
sA11.Name = 'A11';
sA11.Position = [110 110 35 35];

% Create a transition from A1 to A11

tA1A11 = Stateflow.Transition(chart);
tA1A11.Source = sA1;
tA1A11.Destination = sA11;
tA1A11.SourceOClock = 1.;
tA1A11.DestinationOClock = 1.;

% Label transitions A1-A11 and A1-A2
%   to listen for events E1 and E2

tA1A2.LabelString = 'E1';
tA1A11.LabelString = 'E2';

% Create the Events E1 and E2

E1 = Stateflow.Event(chart);
E1.Name = 'E1';

% Move label for transition A1-A1 to the right a bit

pos = tA1A2.LabelPosition;
```

```
pos(1) = pos(1)+15;
tA1A2.LabelPosition = pos;

% Create a default transition to state A

dtA = Stateflow.Transition(chart);
dtA.Destination = sA;
dtA.DestinationOClock = 0;
xsource = sA.Position(1)+sA.Position(3)/2-10;
ysource = sA.Position(2)-20;
dtA.SourceEndPoint = [xsource ysource];

% Create a default transition to state A1

dtA1 = Stateflow.Transition(chart);
dtA1.Destination = sA1;
dtA1.DestinationOClock = 0;
xsource = sA1.Position(1)+sA1.Position(3)/2-10;
ysource = sA1.Position(2)-20;
dtA1.SourceEndPoint = [xsource ysource];
```


API Properties and Methods by Use

This reference section lists and categorizes the properties and methods of the Stateflow Application Programming Interface (API) by different types of use in Stateflow.

Reference Table Column Descriptions (p. 2-3)	Identifies the columns of the tables, in the sections that follow, that list and describe the properties and methods of the Stateflow API
Access Methods (p. 2-4)	Methods for finding and getting objects, and setting properties.
Code Generation and Target Building (p. 2-5)	Properties and methods that control the generated code for a Stateflow chart. These include
Display Control (p. 2-10)	The following properties and methods control the display of a diagram or dialog.
Graphical Appearance (p. 2-11)	Properties and methods that control the graphical appearance of a Stateflow object in its Stateflow diagram. These include
Creating and Deleting Objects (p. 2-20)	Methods that create new Stateflow objects
Containment (p. 2-21)	Properties that control how one Stateflow object contains another Stateflow object.

Data Definition Properties (p. 2-22)	Properties that control the type and size of data.
Debugging Properties (p. 2-25)	Properties that control debugging during simulation.
Identifiers (p. 2-28)	Properties that identify and describe an object.
Interface to Simulink (p. 2-30)	Properties that determine how a Stateflow block interfaces with its Simulink model.
Machine (Model) Identifier Properties (p. 2-34)	Properties that identify parts of a Simulink.
Truth Table Construction Properties (p. 2-35)	Properties relevant only to truth tables.

Reference Table Column Descriptions

Reference tables for Stateflow API properties and methods have the following columns:

- **Name** — The name for the property or method. Each property or method has a name that you use in dot notation along with a Stateflow object to set or obtain the property's value or call the method.
- **Type** — A data type for the property. Some types are other Stateflow API objects, such as the Machine property, which is the Machine object that contains this object.
- **Access** — An access type for the property. Properties that are listed as RW (read/write) can be read and changed. For example, the Name and Description properties of particular objects are RW. However, some properties are RO (read-only) because they are set by MATLAB itself.
- **Description** — A description for the property or method. For some properties, the equivalent GUI operations for setting it in Stateflow are also given.
- **Objects** — The types of objects that have this property or method. The object types are listed by a single letter corresponding to the beginning character of each object type (except for the Target object), which are as follows: Root (R), Machine (M), Chart (C), State (S), Box (B), Function (F), Truth Table (TT), Note (N), Transition (T), Junction (J), Event (E), Data (D), Target (X), Editor (ED), and Clipboard (CB).

Access Methods

The following methods find, get, and set objects and their properties.

Method	Description	Objects
defaultTransitions	Return the default transitions in this chart at the top level of containment.	C S B F
find	Return objects of this object that meet the criteria specified by the arguments.	All
get	Display the property settings of this object.	All
help	Display a list of properties for this object with short descriptions. Used with all objects except the Root and Machine object.	All
innerTransitions	Return the inner transitions that originate with this object and terminate on a contained object.	S B
methods	Return the methods of this object.	All
outerTransitions	Return an array of transitions that exit the outer edge of this object and terminate on an object outside the containment of this object.	S B
set	Set the specified property of this object with a specified value. Used with all objects except the Root object.	All
sourcedTransitions	Return all inner and outer transitions whose source is this object.	S B F J
struct	Return a MATLAB structure containing the property settings of this object.	C S B F N T J D E X

Code Generation and Target Building

This section lists and describes properties and methods that control parsing, code generation, and building of an simulation application in the following topics:

- “Code Generation and Build Methods” on page 2-5
- “Code Generation Properties” on page 2-6
- “Custom Code Properties” on page 2-8

Code Generation and Build Methods

The following methods control parsing, code generation, and building of an simulation application.

Method	Description	Objects
build	Build this target only for those portions of the target's charts that have changed since the last build (i.e., incrementally). See also the methods <code>rebuildAll</code> , <code>generate</code> , <code>rebuildAll</code> , and <code>make</code> .	X
generate	Generate code for this target only for those portions of this target's charts that have changed since the last code generation (i.e., incrementally). See also the methods <code>build</code> , <code>rebuildAll</code> , <code>regenerateAll</code> , and <code>make</code> .	X
getCodeFlag	Return the value of the specified code flag for this target.	X
make	Compile this target incrementally, for only those portions of this target's charts that have changed since the last compilation. For a simulation target (<code>sfun</code>), compile a shared library (S-function MEX-file) from the generated code. See also the methods <code>build</code> , <code>rebuildAll</code> , <code>generate</code> , and <code>regenerateAll</code> .	X

Method	Description	Objects
parse	Parses all the charts in this machine (model) or just this chart.	M C
rebuildAll	Completely rebuild this target. See also the methods build, generate, regenerateAll, and make.	X
regenerateAll	Completely regenerate code for this target. See also the methods build, rebuildAll, generate, and make.	X
setCodeFlag	Set the value of the specified code flag for this target.	X

Code Generation Properties

The following properties control the code generated from the Stateflow charts in a model.

Property	Type	Access	Description	Objects
ApplyToAllLibs	Boolean	RW	If set to true, use settings in this target for all libraries. Equivalent to selecting the Use settings for all libraries check box in this target's Target Builder dialog.	X

Property	Type	Access	Description	Objects
CodeFlagsInfo	Array	RO	<p>A MATLAB vector of structures containing information on code flag settings for this target. Each element in the vector is a MATLAB structure with information about a particular code flag. Each flag corresponds to a selection in the Coder Options dialog for this target. If you want to see information about the first flag for a Target object <code>t</code>, use the following commands:</p> <pre>cfi = t.CodeFlagsInfo disp(cfi(1))</pre> <p>If you want to quickly see the names of all the flags, do the following:</p> <pre>cfi.name</pre> <p>The <code>Name</code> member of the <code>CodeFlagsInfo</code> structure is shorthand for a longer expression in the Coder Options dialog. For example, the name 'comments' actually refers to the dialog setting Comments in generated code.</p> <p>You use the name of a code flag to get and set the code flag value with the methods <code>getCodeFlag</code> and <code>setCodeFlag</code>. Changing the vector returned by <code>CodeFlagsInfo</code> does not change an actual flag.</p>	X
EnableBitOps	Boolean	RW	<p>If set to true, enable C-like bit operations in generated code for this chart. Equivalent to selecting the Enable C-like bit operations check box in the chart properties dialog.</p>	C

Custom Code Properties

The following properties control the custom code that you include with the Stateflow model.

Property	Type	Access	Description	Objects
CodegenDirectory	String	RW	Directory to receive generated code. Applies only to targets other than sfun and rtw targets.	X
CustomCode	String	RW	Custom code included at the top of the generated code. Equivalent to the entry for the Custom code included at the top of generated code selection of the Target Options dialog for this target.	X
CustomInitializer	String	RW	Custom initialization code. Equivalent to the entry for the Custom initialization code (called from mdlInitialize) selection of the Target Options dialog for this target. Applies only to sfun and rtw targets.	X
CustomTerminator	String	RW	Custom termination code. Equivalent to the entry for the Custom termination code (called from mdlTerminate) selection of the Target Options dialog for this target. Applies only to sfun and rtw targets.	X
UserIncludeDirs	Boolean	RW	Custom include directory paths. Equivalent to the entry for the Custom include directory paths selection of the Target Options dialog for this target.	X

Property	Type	Access	Description	Objects
UserLibraries	String	RW	Custom libraries. Equivalent to the entry for the Custom libraries selection of the Target Options dialog for this target.	X
UserSources	String	RW	Custom source files. Equivalent to the entry for the Custom source files selection of the Target Options dialog for this target.	X
ReservedNames	String	RW	Comma- or space-separated list of names to not use in Stateflow generated code. Equivalent to the entry in the Reserved Names panel of the Target Options dialog.	X

Display Control

Lists properties and methods that control the current display in the following topics:

- “Display Methods” on page 2-10
- “Display Properties” on page 2-10

Display Methods

The following methods control the current display.

Method	Description	Objects
dialog	Display the Properties dialog of this object.	M C S B F N T J D E X
view	Display this object in a Stateflow diagram editor.	C S B F N T J D E X
zoomIn and zoomOut	Causes the Stateflow chart editor to zoom in or zoom out on this chart.	ED

Display Properties

The following properties affect the display of the current Stateflow diagram.

Property	Type	Access	Description	Objects
Visible	Boolean	RO	If true, indicates that this object is currently displayed in a Stateflow diagram editor window.	C
Subviewer	Chart or State	RO	State or chart in which this object can be graphically viewed.	S B F N T J TT
ZoomFactor	Double	RW	View magnification level (zoom factor) of this chart in the chart diagram editor.	ED

Graphical Appearance

The following properties and methods control the graphical appearance of objects in Stateflow diagrams in the following topics:

- “Color Properties” on page 2-11
- “Drawing Properties” on page 2-12
- “Font Properties” on page 2-13
- “Position Properties” on page 2-16
- “Text Properties” on page 2-19

Color Properties

The following properties set colors for the graphical objects in Stateflow charts.

Property	Type	Access	Description	Objects
ChartColor	[R,G,B]	RW	Background color of this chart in a 1-by-3 RGB array with each value normalized on a scale of 0 to 1.	C
ErrorColor	[R,G,B]	RW	Set the RGB color for errors in the Stateflow Diagram Editor in a 1-by-3 RGB array with each value normalized on a scale of 0 to 1. Equivalent to changing Error color in the Colors & Fonts dialog under Edit > Style .	C
JunctionColor	[R,G,B]	RW	Set the RGB color for junctions in the Stateflow Diagram Editor in a 1-by-3 RGB array with each value normalized on a scale of 0 to 1. Equivalent to changing the Junction color in the Colors & Fonts dialog under Edit > Style .	C

Property	Type	Access	Description	Objects
SelectionColor	[R,G,B]	RW	Color of selected items for this chart in a 1-by-3 RGB array with each value normalized on a scale of 0 to 1. Equivalent to changing the Selection color in the Colors & Fonts dialog under Edit > Style .	C
StateColor	[R,G,B]	RW	Color of the state box in a 1-by-3 RGB array with each value normalized on a scale of 0 to 1. Equivalent to changing the State/Frame color in the Colors & Fonts dialog under Edit > Style .	C
StateLabelColor	[R,G,B]	RW	Color of the state labels for this chart in 1-by-3 RGB array with each value normalized on a scale of 0 to 1. Equivalent to changing the label color of StateLabel in the Colors & Fonts dialog under Edit > Style .	C
TransitionColor	[R,G,B]	RW	Set the RGB color for transitions in the Stateflow Diagram Editor in a 1-by-3 RGB array with each value normalized on a scale of 0 to 1. Equivalent to changing the Transition color in the Colors & Fonts dialog under Edit > Style .	C
Transition LabelColor	[R,G,B]	RW	Color of the transition labels for this chart in 1-by-3 RGB array with each value normalized on a scale of 0 to 1. Equivalent to changing the label color of TransitionLabel in the Colors & Fonts dialog under Edit > Style .	C

Drawing Properties

The following properties control how Stateflow objects are drawn in their diagrams.

Property	Type	Access	Description	Objects
ArrowSize	Double	RW	Size of transition arrows coming into this object. Equivalent to selecting Arrowhead Size from the context menu for this state.	S B F T J
DrawStyle	String	RW	Drawing style for this transition. Set to 'SMART' (default) for smart transitions or 'STATIC' for static transitions. Equivalent to selecting Smart from the context menu for this transition to toggle between settings. <hr/> Note Transition must be connected to effect a change in the DrawStyle property. Otherwise, an error occurs. <hr/>	T
Editor	Editor	RO	Editor object for this chart.	C

Font Properties

The following properties change the font used for text in a Stateflow chart.

Property	Type	Access	Description	Objects
Font. Angle	Enum	RW	Style of the font for the text in this note. Can be 'ITALIC' or 'NORMAL'. This property overrides the default style for this note, which is set by the StateFont.Angle property of the Chart object containing this note.	N
Font. Name	String	RO	Name of the font for the text in this note. This property is read-only (RO) and set by the StateFont.Name property of the Chart object containing this note.	N

Property	Type	Access	Description	Objects
Font. Size	Double	RW	Size of the font for the label text for this note. This property overrides the default size for this note, which is set by the StateFont.Size property of the Chart object containing this note. Equivalent to selecting Font Size > in the context menu for this note.	N
Font. Weight	Enum	RW	Weight of the font for the label text for this note. Can be 'BOLD' or 'NORMAL'. This property overrides the default weight for the text in this note, which is set by the StateFont.Weight property of the Chart object containing this note.	N
FontSize	Double	RW	Size of the font for the label text for this object. This property overrides the default size for this object, which is set by the StateFont.Size property (TransitionFont.Size for transitions) of the Chart object containing this object. Equivalent to selecting Font Size > in the context menu for this object.	S B F T TT
StateFont. Angle	Enum	RW	Font angle for the labels of State, Box, Function, and Note objects. Can be 'ITALIC' or 'NORMAL'. Equivalent to Italic and Regular settings when changing the font style of the StateLabel in the Colors & Fonts dialog under Edit > Style . Use with property StateFont.Weight to achieve Bold Italic style. You can individually override this property with the Font.Angle property for Note objects.	C

Property	Type	Access	Description	Objects
StateFont. Name	String	RW	Font style used for the labels of State, Box, Function, and Note objects. Enter a string for the font name -- no selectable values. Font remains set to previous font for unrecognized font strings. Equivalent to changing the font of StateLabel in the Colors & Fonts dialog under Edit > Style .	C
StateFont. Size	Integer	RW	Font size for the labels of State, Box, Function, and Note objects. Equivalent to changing the font size of StateLabel in the Colors & Fonts dialog under Edit > Style . You can individually override this property with the <code>FontSize</code> property for State, Box, and Function objects and with the <code>Font.Size</code> property for Note objects.	C
StateFont. Weight	Enum	RW	Font weight for state labels. Can be 'BOLD' or 'NORMAL'. Equivalent to Bold and Regular settings of StateLabel in the Colors & Fonts dialog under Edit > Style . Use with the property <code>StateFont.Angle</code> to achieve Bold Italic style. You can individually override this property with the <code>Font.Weight</code> property for Note objects.	C
TransitionFont. Angle	Enum	RW	Font angle for state labels. Can be 'ITALIC' or 'NORMAL'. Equivalent to Italic and Regular settings when changing font style of TransitionLabel in the Colors & Fonts dialog under Edit > Style . Use with property <code>StateFont.Weight</code> to achieve Bold Italic style.	C

Property	Type	Access	Description	Objects
TransitionFont. Name	String	RW	Font used for transition labels. Enter string for font name (no selectable values). Font remains set to previous font for unrecognized font strings. Equivalent to changing the font of TransitionLabel in the Colors & Fonts dialog under Edit > Style .	C
TransitionFont. Size	Integer	RW	Default font size for transition labels. Truncated to closest whole number less than or equal to entered value. Equivalent to changing font size of TransitionLabel in the Colors & Fonts dialog under Edit > Style .	C
TransitionFont. Weight	Enum	RW	Font weight for transition labels. Can be 'BOLD' or 'NORMAL'. Equivalent to Bold and Regular settings when changing font style of TransitionLabel in the Colors & Fonts dialog under Edit > Style . Use with property StateFont.Angle to achieve Bold Italic style.	C

Position Properties

The following properties control the position of Stateflow objects in a Stateflow diagram.

Property	Type	Access	Description	Objects
BadIntersection	Boolean	RO	If true, this object graphically intersects another state, box, or function in an invalid way.	S B F TT

Property	Type	Access	Description	Objects
Destination	State or Junction	RW	<p>Destination state or junction of this transition. Assign Destination the destination object for this transition.</p> <p>You can also use the property Destination to detach the destination end of a transition through the command <code>t.Destination = []</code> where <code>t</code> is the Transition object.</p>	T
Destination OClock	Double	RW	<p>Location of transition destination connection on state. Varies from 0 to 12 for full clock cycle location. Value taken as modulus 12 of entered value.</p>	T
LabelPosition	Rect	RW	<p>Position and size of this transition's label in the Stateflow chart, given in the form of a 1-by-4 array consisting of the following:</p> <ul style="list-style-type: none"> • (x,y) coordinates for the label's left upper vertex relative to the upper left vertex of the Stateflow diagram editor workspace • Width and height of the label 	T
MidPoint	Rect	RW	<p>Position of the midpoint of this transition relative to the upper left corner of the Stateflow diagram editor workspace, in a 1-by-2 (x,y) point array.</p>	T

Property	Type	Access	Description	Objects
Position	Rect	RW	<p>Position and size of box-like objects in the Stateflow chart, given in the form of a 1-by-4 array consisting of the following:</p> <ul style="list-style-type: none"> • (x,y) coordinates for the object's left upper vertex relative to the upper left vertex of the Stateflow diagram editor workspace • Width and height of the box 	S B F TT EML N
Position.Center	Rect	RW	(x,y) position of junction relative to the upper left vertex of the parent chart or state.	J
Position.Radius	Double	RW	Radius of this junction.	J
Source	State or Junction	RW	<p>Source state or junction of this transition. Assign Source the source object for this transition.</p> <p>You can also use the property Source to detach the source end of a transition with the command <code>t.Source = []</code> where t is the Transition object.</p>	T
SourceEndPoint	Rect	RO*	<p>x,y spatial coordinates for the endpoint of a transition.</p> <p>*This property can be changed only for default transitions. For all other transitions it is RO (read only).</p>	T

Property	Type	Access	Description	Objects
SourceOClock	Double	RW	Location of transition source connection on state. Varies from 0 to 12 for full clock cycle location. Value taken as modulus 12 of entered value.	T
WindowPosition	Rect	RW	Position and size of this chart given in the form of a 1-by-4 array consisting of the following: <ul style="list-style-type: none"> (x,y) coordinates for the window's left bottom vertex relative to the lower left corner of the screen Width and height of the box 	ED

Text Properties

The following properties control the text and text appearance apart from font and color in Stateflow diagrams.

Property	Type	Access	Description	Objects
Alignment	Enum	RW	Alignment of text in note box. Can be 'LEFT', 'CENTER', or 'RIGHT'.	N
Interpretation	Enum	RW	How the text in this note is interpreted for text processing. Can be 'NORMAL' or 'TEX'.	N
LabelString	String	RW	Label for this object. Equivalent to typing the label for this object in its label text field in the diagram editor.	S B F T TT EML
Text	String	RW	Label for this note. The text content for this note that you enter directly into the note in the diagram editor or in the Label field of the Properties dialog for this note.	N

Creating and Deleting Objects

Use the following methods to create and delete Stateflow objects.

Method	Description	Objects
copy	Copy the specified array of objects to the clipboard for pasting. See also the pasteTo method.	CB
delete	Delete this object.	All but R M C CB ED
pasteTo	Paste the objects in the Clipboard to the specified container object. See also copy method.	CB
Stateflow.Box	Create a box for a parent chart, state, box, or function.	NA
Stateflow.Data	Create a data for a parent machine, chart, state, box, or function.	NA
Stateflow.Event	Create an event for a parent machine, chart, state, box, or function.	NA
Stateflow.Function	Create a graphical function for a parent chart, state, box, or function.	NA
Stateflow.Junction	Create a junction for a parent chart, state, box, or function.	NA
Stateflow.Note	Create a note for a parent chart or state.	NA
Stateflow.State	Create a state for a parent chart, state, box, or function.	NA
Stateflow.Target	Create a target for a parent machine.	NA
Stateflow.Transition	Create a transition for a parent chart, state, box, or function.	NA
Stateflow.TruthTable	Create a truth table for a parent state or chart.	NA

Containment

The following properties control how one Stateflow object contains another Stateflow object.

Property	Type	Access	Description	Objects
Chart	Chart	RO	Chart object containing this object.	S B F N T J TT
Decomposition	Enum	RW	Set this property to 'EXCLUSIVE_OR' to specify exclusive (OR) decomposition for the states at the first level of containment in this chart or state. Set to 'PARALLEL_AND' to specify parallel (AND) decomposition for these states. Equivalent to selecting the Decomposition in the context menu for the chart or state.	C S
IsGrouped	Boolean	RW	If set to true, group this object. Nothing is allowed to change inside a grouped object. You must first ungroup the object before you can change its contents. This property is also useful for copying states and their contents to a new location. See “Copying by Grouping (Recommended)” on page 1-34.	S B F
IsSubchart	Boolean	RW	If set to true, makes this state, box, or graphical function a subchart.	S B F
Machine	Machine	RO	Machine that contains this object. A machine object contains all of the Chart objects in a Model.	C S B F N T J D E X TT

Data Definition Properties

The following properties control the type, size, and value of data in Stateflow diagrams.

Property	Type	Access	Description	Objects
DataType	Enum	RW	Data type of this data. Can have one of the following possible values: 'boolean', 'uint8', 'int8', 'uint16', 'int16', 'uint32', 'int32', 'single', 'double' and 'fixpt'. Equivalent to an entry in the Type column for this data in Explorer or the Type field in the properties dialog for this data.	D
FixptType. Bias	Double	RW	The Bias value for this fixed-point type.	D
FixptType. FractionalSlope	Double	RW	The Fractional Slope value for this fixed-point type.	D
FixptType. RadixPoint	Integer	RW	The power of two specifying the binary point location for this fixed-point type.	D
FixptType. BaseType	Enum	RW	The size and sign of the base for the quantized integer, Q, of this fixed-point type.	D
ParsedInfo. Array. Size	Integer	RO	Numeric equivalent of string Data property Props.Array.Size.	D
ParsedInfo. Array. FirstIndex	Integer	RO	Numeric equivalent of string Data property Props.Range.FirstIndex.	D
ParsedInfo. InitialValue	Double	RO	Numeric equivalent of string Data property Props.InitialValue.	D
ParsedInfo. Range. Maximum	Double	RO	Numeric equivalent of string Data property Props.Range.Maximum.	D

Property	Type	Access	Description	Objects
ParsedInfo. Range. Minimum	Double	RO	Numeric equivalent of string Data property Props.Range.Minimum.	D
Port	Integer	RW	Port index number for this input or output data or event (default = 1).	D E
Props. Array. Size	String	RW	Specifying a positive value for this property specifies that this data is an array of specified size. Equivalent to entering a positive value in the Size column for this data in the Explorer or in the Sizes field of the properties dialog for this data.	D
Props. Array. FirstIndex	String	RW	Index of the first element of this data if it is an array (Props.Array.Size >= 1). Equivalent to entering a value of zero or greater in the First Index field of the Properties dialog for this data.	D
Props. InitialValue	String	RW	If the source of the initial value for this data is the Stateflow data dictionary, this is the value used. Equivalent to entering this value in the InitVal column for this data in the Explorer or similar field in the Properties dialog for this data.	D
Props. Range. Maximum	String	RW	Maximum value that this data can have during execution or simulation of the state machine. Equivalent to entering value in Max column for this data in Explorer or the Max field in the Properties dialog for this data.	D

Property	Type	Access	Description	Objects
Props. Range. Minimum	String	RW	Minimum value that this data can have during execution or simulation of the state machine. Equivalent to entering value in the Min column for this data in Explorer or in the Min field in the properties dialog for this data.	D
SaveToWorkspace	Boolean	RW	If set to true, this data is saved to the MATLAB workspace. Equivalent to selecting the ToWS column entry for this data in the Explorer or selecting the Save final value to base workspace field in the properties dialog for this data.	D
Units	String	RW	Physical units corresponding to the value of this data object.	D

Debugging Properties

The following properties control values used in debugging Stateflow applications with the Stateflow Debugger.

Property	Type	Access	Description	Objects
Debug. Animation. Delay	Double	RW	Specify a delay (slow down) value for animation. Equivalent to setting the Delay (sec) field in the Animation section of the Debugger window.	M
Debug. Animation. Enabled	Boolean	RW	If true, animation (simulation) is enabled. If false (=0), disabled. Equivalent to selecting the Enabled or Disabled radio button of the Animation section of the Debugger window.	M
Debug. BreakOn. ChartEntry	Boolean	RW	If true, sets the chart entry breakpoint for all charts in this machine. Equivalent to selecting the Chart Entry check box on the Debugger window.	M
Debug. BreakOn. EventBroadcast	Boolean	RW	If true, sets the event broadcast breakpoint for all charts in this machine. Equivalent to selecting the Event Broadcast check box on the Debugger window.	M
Debug. BreakOn. StateEntry	Boolean	RW	If true, sets the state entry breakpoint for all states in this machine. Equivalent to selecting the State Entry check box on the Debugger window.	M
Debug. Breakpoints. EndBroadcast	Boolean	RW	If true, sets a debugger breakpoint for the end of the broadcast of this event. Equivalent to selecting the End of broadcast check box in the Properties dialog for this event.	E

Property	Type	Access	Description	Objects
Debug. Breakpoints. StartBroadcast	Boolean	RW	If true, sets a debugger breakpoint for the start of the broadcast of this event. Equivalent to selecting the Start of broadcast check box in the properties dialog for this event.	E
Debug. Breakpoints. onDuring	Boolean	RW	If true, sets the state during breakpoint for this chart. Equivalent to selecting the State During check box in the properties dialog for this state.	S
Debug. Breakpoints. OnEntry	Boolean	RW	If true, sets the entry breakpoint for this object. Equivalent to selecting the Chart Entry or State Entry check box in the properties dialog for this chart or state, respectively.	C S
Debug. Breakpoints. OnExit	Boolean	RW	If true, sets the state entry breakpoint for this object. Equivalent to selecting the State Exit check box in the properties dialog for this object.	S
Debug. Breakpoints. WhenTested	Boolean	RW	If true, sets a debugging breakpoint to occur when this transition is tested to see if it is a valid transition. Equivalent to selecting the When Tested check box in the properties dialog of this transition.	T
Debug. Breakpoints. WhenValid	Boolean	RW	If true, sets a debugging breakpoint to occur when this transition has tested as valid. Equivalent to selecting the When Valid check box in the Properties dialog of this transition.	T
Debug. DisableAll Breakpoints	Boolean	RW	If true, disables the use of all breakpoints in this machine. Equivalent to selecting the Disable all check box in the Debugger window.	M

Property	Type	Access	Description	Objects
Debug. State RunTimeCheck. Inconsistencies	Boolean	RW	If true, checks for state inconsistencies during a debug session. Equivalent to selecting the State Inconsistency check box in the Debugger window.	M
Debug. RunTimeCheck. TransitionConflicts	Boolean	RW	If true, checks for transition conflicts during a debug session. Equivalent to selecting the Transition Conflict check box in the Debugger window.	M
Debug. RunTimeCheck. CycleDetection	Boolean	RW	If true, checks for cyclical behavior errors during a debug session. Equivalent to selecting the Detect Cycles check box in the Debugger window.	M
Debug. RunTimeCheck. DataRangeChecks	Boolean	RW	If true, checks for data range violations during a debug session. Equivalent to selecting the Data Range check box in the Debugger window.	M
Debug. Watch	Boolean	RW	If true, causes the debugger to halt execution if this data is modified. Equivalent to selecting the Watch column entry for this data in the Explorer or selecting the Watch in debug check box in the Properties dialog for this data.	D
IsTestPoint	Boolean	RW	If true (default = false), sets this data or state as a Stateflow test point. You can monitor Stateflow test points with a floating scope during simulation. You can also log test point values into MATLAB workspace objects. See “Monitoring Stateflow Test Points” in the Stateflow and Stateflow Coder User’s Guide documentation.	S D

Identifiers

The following properties identify objects or the Stateflow version.

Property	Type	Access	Description	Objects
Description	String	RW	Description of this object. Equivalent to entering a description in the Description field of the Properties dialog for this object.	M C S B F N T J D E X T T
Document	String	RW	Document link to this note. Equivalent to entering the Document Link field of the Properties dialog for this object.	M C S B F N T J D E X T T
Id	Integer	RO	Unique identifier assigned to this object to distinguish it from other objects loaded in memory.	M C S B F N T J D E X T T
Name	String	RW	Name of this object. This property is RW except for the name of Machine object, which is RO.	M C S B F D E X T T
SfVersion	Double	RO	Full version number for current Stateflow. For example, the string '41112101' appears for Stateflow version 4.1.1 and MATLAB version 12.1. The remaining '01' is for internal use.	M

Property	Type	Access	Description	Objects
Tag	Any Type	RW	A field you can use to hold data of any type for this object.	M C S B F T J D E X TT
Type	Enum	RO	Type of this state or junction. For states, can be one of the following: <ul style="list-style-type: none"> • 'OR' (inclusive) • 'AND' (parallel) The type of a state is determined by the parent's Decomposition property. For junctions, can be one of the following: <ul style="list-style-type: none"> • 'CONNECTIVE' • 'HISTORY' 	S J

Interface to Simulink

The following properties and (methods) control how data and events are input from and output to the Simulink model for a Stateflow chart.

Property (Method)	Type	Access	Description	Objects
ChartUpdate	Enum	RW	<p>Activation method of this chart. Can be one of the following:</p> <ul style="list-style-type: none"> • 'INHERITED' (Triggered or Inherited) • 'DISCRETE' (Sampled) • 'CONTINUOUS' (Continuous) <p>These preceding entries are equivalent to the parenthetical entries for the Update method field in the Properties dialog for this chart.</p>	C
ExecuteAtInitialization	Boolean	RW	<p>If set to true, initialize this chart's state configuration at time zero instead of at first input event. Equivalent to selecting the Execute (enter) Chart at Initialization check box in the chart properties dialog.</p>	C
ExportChartFunctions	Boolean	RW	<p>If set to true (default = false), graphical functions at chart level are made global. Equivalent to selecting the Export Chart Level Graphical Functions (Make Global) check box in the chart properties dialog.</p>	C

Property (Method)	Type	Access	Description	Objects
InitializeOutput	Boolean	RW	Applies the initial value of outputs every time a chart wakes up, not only at time 0. See “Setting Properties for Individual Charts”.	C
(outputData)	No Return	NA	Output the activity status of this state to Simulink via a data output port on the Chart block of this state.	S
Port	Integer	RW	Port index number for this input or output data or event (default = 1).	D E
SampleTime	String	RW	Sample time for activating this chart. Applies only when the ChartUpdate property for this chart is set to 'DISCRETE' (= Sampled in the Update method field in the Properties dialog for this chart).	C
SaveToWorkspace	Boolean	RW	If set to true, this data is saved to the MATLAB workspace. Equivalent to selecting the SaveToWorkspace column entry for this data in the Model Explorer or selecting the Save final value to base workspace field in the properties dialog for this data.	D

Property (Method)	Type	Access	Description	Objects
Scope	Enum	RW	<p>Scope of this data. Allowed values vary with the object containing this data, which are as follows:</p> <ul style="list-style-type: none"> • 'Local' • 'Constant' • 'Imported' (machine objects only) • 'Exported' (machine objects only) • 'Input' (chart objects only) • 'Output' (chart objects only) • 'Temporary' (function objects only) • 'Function input' (function objects only) • 'Function output' (function objects only) <p>Above values correspond to entries in the Scope field of the Data or Event dialog.</p>	D E

Property (Method)	Type	Access	Description	Objects
StrongDataTyping WithSimulink	Boolean	RW	If set to true, set strong data typing with Simulink I/O. Equivalent to the Use Strong Data Typing with Simulink I/O check box in the chart properties dialog.	C
Trigger	Enum	RW	<p>Type of signal that triggers this chart input event. Also the type of trigger associated with this chart output event.</p> <p>The following triggers apply to both chart input and output events:</p> <ul style="list-style-type: none"> • 'Either' (Either Edge) • 'Function call' (Function Call) <p>The following triggers apply only to chart input events:</p> <ul style="list-style-type: none"> • 'Rising' (Rising Edge) • 'Falling' (Falling Edge) <p>The preceding entries are equivalent to the entries in parentheses for the Trigger field in the Event dialog for this event.</p>	E

Machine (Model) Identifier Properties

The following properties identify parts of the Simulink model containing a Stateflow chart.

Property	Type	Access	Description	Objects
Created	String	RO	Date of creation of this machine.	M
Creator	String	RW	Creator of this machine.	M
Dirty	Boolean	RW	If true, this object has changed since it was opened or saved.	M C
FullFileName	String	RO	Full path name of file under which this machine (model) is stored.	M
Iced	Boolean	RO	Equivalent to property Locked except that this property is used internally to lock this object from being changed during activities such as simulation.	M C
IsLibrary	Boolean	RO	If true, specifies that the current model builds a library and not an application.	M
Locked	Boolean	RW	If set to true, prevents user from changing any Stateflow chart in this machine or chart.	M C
Modified	String	RW	Comment area for entering date and name of modification to this machine (model).	M
Version	String	RW	Comment string for recording version of this model.	M

Truth Table Construction Properties

The following properties control the definition of a truth table.

Property	Type	Access	Description	Objects
ActionTable	Cell Array	RW	A cell array of strings containing the contents of the Action Table for this truth table.	TT
ConditionTable	Cell Array	RW	A cell array of strings containing the contents of the Action Table for this truth table.	TT
OverSpec Diagnostic	String	RW	Interprets the error diagnosis of this truth table as overspecified according to the possible values 'Error', 'Warning', or 'None'. In the truth table editor, the value of this property is assigned by selecting Overspecified from the Diagnostics menu item and then selecting one of the three values.	TT
UnderSpec Diagnostic	String	RW	Interprets the error diagnosis of this truth table as underspecified according to the possible values 'Error', 'Warning', or 'None'. In the truth table editor, the value of this property is assigned by selecting Underspecified from the Diagnostics menu item and then selecting one of the three values.	TT

API Properties and Methods

— By Category

Reference Table Columns (p. 3-6)	Describes the columns appearing in the tables listing the properties and methods by object in the Stateflow API.
Constructor Methods (p. 3-7)	Construct Stateflow objects
Editor Properties (p. 3-8)	Descriptions of properties for the Editor (diagram editor) in the Stateflow API.
Editor Methods (p. 3-9)	Descriptions of methods for the Editor (diagram editor) in the Stateflow API.
Clipboard Methods (p. 3-10)	Descriptions of the methods of the Clipboard used for copying and pasting objects from chart to chart.
All Object Methods (p. 3-11)	Describes methods that belong to all Stateflow objects.
Root Methods (p. 3-12)	Description of the methods of the Root object that contains all other Stateflow objects
Machine Properties (p. 3-13)	Descriptions of properties for the Machine object (model) in the Stateflow API.

Machine Methods (p. 3-17)	Descriptions of methods for the Machine object (model) in the Stateflow API.
Chart Properties (p. 3-18)	Descriptions of properties for Chart objects (charts) in the Stateflow API.
Chart Methods (p. 3-26)	Descriptions of methods for Chart objects (charts) in the Stateflow API.
State Properties (p. 3-27)	Descriptions of properties for State objects (states) in the Stateflow API.
State Methods (p. 3-31)	Descriptions of methods for State objects (states) in the Stateflow API.
Box Properties (p. 3-33)	Descriptions of properties for Box objects (boxes) in the Stateflow API.
Box Methods (p. 3-35)	Descriptions of methods for Box objects (boxes) in the Stateflow API.
Graphical Function Properties (p. 3-36)	Descriptions of properties for Function objects (graphical functions) in the Stateflow API.
Graphical Function Methods (p. 3-39)	Descriptions of methods for Function objects (graphical functions) in the Stateflow API.
Truth Table Properties (p. 3-40)	Descriptions of properties for Truth Table objects in Stateflow API.
Truth Table Methods (p. 3-43)	Descriptions of methods for Truth Table objects in Stateflow API.
Truth Table Chart Properties (p. 3-44)	Descriptions of properties for Truth Table Chart objects in the Stateflow API.
Truth Table Chart Methods (p. 3-47)	Descriptions of methods for Truth Table Chart objects in Stateflow API.
Embedded MATLAB Function Properties (p. 3-48)	Descriptions of properties for Embedded MATLAB Function objects in the Stateflow API.

Embedded MATLAB Function Methods (p. 3-51)	Descriptions of methods for Embedded MATLAB Function objects in the Stateflow API.
Note Properties (p. 3-52)	Descriptions of properties for Note objects (notes) in the Stateflow API.
Note Methods (p. 3-54)	Descriptions of methods for Note objects (notes) in the Stateflow API.
Transition Properties (p. 3-55)	Descriptions of properties for Transition objects (transitions) in the Stateflow API.
Transition Methods (p. 3-59)	Descriptions of properties and methods for Transition objects (transitions) in the Stateflow API.
Junction Properties (p. 3-60)	Descriptions of properties for Junction objects (junctions) in the Stateflow API.
Junction Methods (p. 3-61)	Descriptions of methods for Junction objects (junctions) in the Stateflow API.
Data Properties (p. 3-62)	Description of properties for Data objects (data) in the Stateflow API.
Data Methods (p. 3-70)	Description of methods for Data objects (data) in the Stateflow API.
Event Properties (p. 3-71)	Descriptions of properties for Target objects (targets) in the Stateflow API.
Event Methods (p. 3-74)	Descriptions of methods for Target objects (targets) in the Stateflow API.

Target Properties (p. 3-75)

Descriptions of properties for Target objects (targets) in the Stateflow API.

Target Methods (p. 3-80)

Descriptions of methods for Target objects (targets) in the Stateflow API.

Reference Table Columns

Reference tables for Stateflow API properties and methods have the following columns:

- **Name** — The name for the property or method. Each property or method has a name that you use in dot notation along with a Stateflow object to set or obtain the property's value or call the method.
- **Type** — A data type for the property. Some types are other Stateflow API objects, such as the `Machine` property, which is the `Machine` object that contains this object.
- **Access** — An access type for the property. Properties that are listed as RW (read/write) can be read and changed. For example, the `Name` and `Description` properties of particular objects are RW. However, some properties are RO (read-only) because they are set by MATLAB itself.
- **Description** — A description for the property or method. For some properties, the equivalent GUI operations in Stateflow for setting it are also given.

Constructor Methods

The following methods create a new Stateflow object for a parent object specified as an argument in the general expression `o = Stateflow.Object(p)`, where `o` is a handle to an API object for the new Stateflow object, `p` is a handle to the parent object, and *Object* is the type of the object:

Method	Description
<code>Stateflow.Box</code>	Create a box for a parent chart, state, box, or function.
<code>Stateflow.Data</code>	Create a data for a parent machine, chart, state, box, or function.
<code>Stateflow.EMFunction</code>	Create an Embedded MATLAB function for a parent chart or state.
<code>Stateflow.Event</code>	Create an event for a parent machine, chart, state, box, or function.
<code>Stateflow.Function</code>	Create a graphical function for a parent chart, state, box, or function.
<code>Stateflow.Junction</code>	Create a junction for a parent chart, state, box, or function.
<code>Stateflow.Note</code>	Create a note for a parent chart, state, box, or function.
<code>Stateflow.State</code>	Create a state for a parent chart, state, box, or function.
<code>Stateflow.Target</code>	Create a target for a parent machine.
<code>Stateflow.Transition</code>	Create a transition for a parent chart, state, box, or function.
<code>Stateflow.TruthTable</code>	Create a truth table for a parent chart or state.

Editor Properties

The Editor object has the properties in the table below. See also “Editor Methods” on page 3-9.

Property	Type	Access	Description
WindowPosition	Rect	RW	<p>Position and size of this chart given in the form of a 1-by-4 array consisting of the following:</p> <ul style="list-style-type: none">• (x,y) coordinates for the window’s left bottom vertex relative to the lower left corner of the screen• Width and height of the box <p>Default value = [124.3125 182.8125 417 348.75]</p>
ZoomFactor	Double	RW	<p>View magnification level (zoom factor) of this chart in the chart diagram editor. A value of 1 corresponds to a zoom factor of 100%, 2 to a value of 200%, and so on. Default value = 1.</p>

Editor Methods

The Editor object has the methods displayed in the table below. For details on each method, see the reference pages.

See also “Editor Properties” on page 3-8.

Method	Description
disp	Display the property names and their settings for this Editor object.
get	Return the specified property settings for the Editor object.
help	Display a list of properties for this Editor object with short descriptions.
methods	Display all nonglobal methods of this Editor object.
set	Set the specified property of this Editor object with the specified value.
struct	Return and display a MATLAB structure containing the property settings of this Editor object.
zoomIn and zoomOut	Cause the Stateflow chart editor to zoom in or zoom out on this chart.

Clipboard Methods

The Clipboard object has the methods displayed in the table below. For details on each method, see the reference pages.

Method	Description
copy	Copy the objects specified to this Clipboard object.
get	Return the specified property settings for this Clipboard object.
help	Display a list of properties for this Clipboard object with short descriptions.
methods	Display all nonglobal methods of this Clipboard object.
pasteTo	Paste the contents of this clipboard to the specified container object.
set	Set the specified property of this Clipboard object with the specified value.
struct	Return and display a MATLAB structure containing the property settings of this Clipboard object.

All Object Methods

The following methods apply to all API objects including those of Stateflow. Only object-exclusive methods appear when you use the `methods` method to display methods for an object. However, the tables of methods for each API object that follow do list these methods as if they were their own.

See the reference pages for details on each method.

Method	Description
<code>delete</code>	Delete this object. Used with all objects except the Root, Machine, Chart, Clipboard, and Editor objects.
<code>disp</code>	Display the property names and their settings for this object.
<code>find</code>	Find all objects of this object that meet the specified criteria.
<code>get</code>	Return the specified property settings for this object.
<code>methods</code>	Display all nonglobal methods of this object.
<code>set</code>	Set the specified property of this object with a specified value.
<code>struct</code>	Return and display a MATLAB structure containing the property settings of this object.
<code>up</code>	Return the parent (container) object of this object.

Root Methods

The Root object has the methods displayed in the table below. For details on each method, see the reference pages.

Method	Description
find	Find all objects that this Root object contains that meet the specified criteria.
get	Return the specified property settings for the Root object.
help	Display a list of properties for the Root object with short descriptions.
methods	Display all nonglobal methods of this Root object.
set	Set the specified property of this Root object with the specified value.
struct	Return and display a MATLAB structure containing the property settings of this Root object.

Machine Properties

Stateflow API objects of type Machine have the properties shown in the table below. See also “Machine Methods” on page 3-17.

Property	Type	Access	Description
Created	String	RO	Date of creation of this machine.
Creator	String	RW	Creator (default = 'Unknown') of this machine.
Debug. Animation. Enabled	Boolean	RW	If set to true (default), animation (simulation) is enabled. If false, disabled. Equivalent to the Enabled or Disabled radio button of the Animation section of the Debugger window.
Debug. Animation. Delay	Double	RW	Specify a value to delay (slow down) animation (default value = 0). Equivalent to the Delay (sec) field in the Animation section of the Debugger window.
Debug. BreakOn. ChartEntry	Boolean	RW	If set to true (default = false), set the chart entry breakpoint for all charts in this machine. Equivalent to the Chart Entry check box in the Debugger window.
Debug. BreakOn. EventBroadcast	Boolean	RW	If set to true (default = false), set the event broadcast breakpoint for all charts in this machine. Equivalent to the Event Broadcast check box in the Debugger window.
Debug. BreakOn. StateEntry	Boolean	RW	If set to true (default = false), set the state entry breakpoint for all charts in this machine. Equivalent to the State Entry check box in the Debugger window.

Property	Type	Access	Description
Debug. DisableAllBreakpoints	Boolean	RW	If set to true (default = false), disable the use of all breakpoints in this machine. Equivalent to the Disable all check box in the Debugger window.
Debug. RunTimeCheck. CycleDetection	Boolean	RW	If set to true, check for cyclical behavior errors during a debug session. Equivalent to the Detect Cycles check box in the Debugger window.
Debug. RunTimeCheck. DataRangeChecks	Boolean	RW	If set to true (default), check for data range violations during a debug session. Equivalent to the Data Range check box in the Debugger window.
Debug. RunTimeCheck. StateInconsistencies	Boolean	RW	If set to true (default), check for state inconsistencies during a debug session. Equivalent to the State Inconsistency check box in the Debugger window.
Debug. RunTimeCheck. TransitionConflicts	Boolean	RW	If set to true (default), check for transition conflicts during a debug session. Equivalent to the Transition Conflict check box in the Debugger window.
Description	String	RW	Description of this state (default = ''). Equivalent to entering a description in the Description field of the properties dialog for this machine.
Dirty	Boolean	RW	If true (default), this model has changed since it was opened or saved.

Property	Type	Access	Description
Document	String	RW	Document link to this machine (default = ''). Equivalent to entering the Document Link field of the properties dialog for this machine.
EnableBitOps	Boolean	RW	If true, recognize C bitwise operators (~, &, , ^, >>, and so on) in action language statements for all Stateflow charts in the model and encode them as C bitwise operations.
FullFileName	String	RO	Full path name of file (default value = '') under which this machine (model) is stored.
Iced	Boolean	RO	Equivalent to property Locked (default = false) except that this property is used internally to lock this model from being changed during activities such as simulation.
Id	Integer	RO	Unique identifier assigned to this machine to distinguish it from other objects loaded in memory.
isLibrary	Boolean	RO	If true (default = false), specifies that the current model builds a library and not an application.
Locked	Boolean	RW	If set to true (default = false), prevents user from changing any Stateflow chart in this model.
Machine	Machine	RO	A handle to the Machine object for this Machine object, that is, this Machine object.

Property	Type	Access	Description
Modified	String	RW	Comment area (default = '') for entering date and name of modification to this model.
Name	String	RO	Name of this model (default = 'untitled') set when saved to disk.
SfVersion	Double	RO	Full version number for current Stateflow. For example, the string '41112101' appears for Stateflow version 4.1.1 and MATLAB version 12.1. The remaining '01' is for internal use.
Tag	Any Type	RW	A field you can use to hold data of any type for this machine (default = []).
Version	String	RW	Comment string (default = 'none') for recording the version of this model.

Machine Methods

Machine objects have the methods displayed in the table below. For details on each method, see the reference pages.

See also “Machine Properties” on page 3-13.

Method	Description
dialog	Display the properties dialog of this machine.
disp	Display the property names and their settings for this Machine object.
find	Find all objects that this machine contains that meet the specified criteria. Note Do not use the <code>-depth</code> switch with the <code>find</code> method for a machine object.
get	Return the specified property settings for this machine.
help	Display a list of properties for this Machine object with short descriptions.
methods	Display all nonglobal methods of this Machine object.
parse	Parse all the charts in this machine.
set	Set the specified property of this Machine object with the specified value.
struct	Return and display a MATLAB structure containing the property settings of this Machine object.

Chart Properties

Stateflow API objects of type Chart have the properties shown below. See also “Chart Methods” on page 3-26.

Property	Type	Access	Description
ChartColor	[R,G,B]	RW	Background color of this chart in a 1-by-3 RGB array (default = [1 0.9608 0.8824]) with each value normalized on a scale of 0 to 1.
ChartUpdate	Enum	RW	Activation method of this chart. Can be 'INHERITED' (default), 'DISCRETE', or 'CONTINUOUS'. Equivalent to the Update method field in the properties dialog for this chart, which takes one of the following corresponding selections: Triggered or Inherited, Sampled, Continuous .
Debug. Breakpoints. OnEntry	Boolean	RW	If set to true (default = false), set the chart entry breakpoint for this chart. Equivalent to selecting the Chart Entry check box in the properties dialog for this chart.
Decomposition	Enum	RW	Set this property to 'EXCLUSIVE_OR' (default) to specify exclusive (OR) decomposition for the states at the first level of containment in this chart. Set to 'PARALLEL_AND' to specify parallel (AND) decomposition for these states. Equivalent to the Decomposition selection in the context menu for the Stateflow diagram editor.

Property	Type	Access	Description
Description	String	RW	Description (default = '') of this state. Equivalent to entering a description in the Description field of the properties dialog for this chart.
Dirty	Boolean	RW	If set to true (default = false), this chart has changed since being opened or saved.
Document	String	RW	Document link (default = '') to this chart. Equivalent to entering the Document Link field of the properties dialog for this chart.
Editor	Editor	RO	Editor object for this chart.
EnableBitOps	Boolean	RW	If set to true (default = false), enables C-like bit operations in generated code for this chart. Equivalent to the Enable C-like bit operations check box in the chart properties dialog.
ErrorColor	[R,G,B]	RW	Set the RGB color for errors in the Stateflow Diagram Editor in a 1-by-3 RGB array (default value [1 0 0]) with each value normalized on a scale of 0 to 1. Equivalent to changing the Error color in the Colors & Fonts dialog under Edit > Style .
ExecuteAtInitialization	Boolean	RW	If set to true (default = false), this chart's state configuration is initialized at time zero instead of at the first input event. Equivalent to selecting the Execute (enter) Chart at Initialization check box in chart properties dialog

Property	Type	Access	Description
ExportChartFunctions	Boolean	RW	If set to true (default = false), graphical functions at chart level are made global. Equivalent to selecting the Export Chart Level Graphical Functions (Make Global) check box in chart properties dialog.
Iced	Boolean	RO	Equivalent to property Locked (default = false) except that this property is used internally to lock this chart from change during activities such as simulation.
Id	Integer	RO	Unique identifier assigned to this chart to distinguish it from other objects loaded in memory.
InitializeOutput	Boolean	RW	Applies the initial value of outputs every time a chart wakes up, not only at time 0. See “Setting Properties for Individual Charts”.
JunctionColor	[R,G,B]	RW	Set the RGB color for junctions in the Stateflow Diagram Editor in a 1-by-3 RGB array (default value [0.6824 0.3294 0]) with each value normalized on a scale of 0 to 1. Equivalent to changing the Junction color in the Colors & Fonts dialog under Edit > Style .
Locked	Boolean	RW	If set to true (default = false), mark this chart as read-only and prohibit any write operations on it. Equivalent to selecting the Locked check box in the Editor section of the properties dialog for this chart.
Machine	Machine	RO	Machine that contains this chart.

Property	Type	Access	Description
Name	String	RW	Name of this chart (default = 'Chart'). Equivalent to changing the name of this chart's Stateflow block in Simulink.
SampleTime	String	RW	Sample time for activating this chart (default = ''). Applies only when the UpdateMethod property for this chart is set to 'DISCRETE' (= Sampled in the Update method field in the properties dialog for this chart).
SelectionColor	[R,G,B]	RW	Color of selected items for this chart in a 1-by-3 RGB array (default value [1 0 0.5176]) with each value normalized on a scale of 0 to 1. Equivalent to changing the Selection color in the Colors & Fonts dialog under Edit > Style .
StateColor	[R,G,B]	RW	Color of the state box in a 1-by-3 RGB array (default value [0 0 0]) with each value normalized on a scale of 0 to 1. Equivalent to changing the State/Frame color in the Colors & Fonts dialog under Edit > Style .
StateFont. Angle	Enum	RW	Font angle for the labels of State, Box, Function, and Note objects. Can be 'ITALIC' or 'NORMAL' (default). Equivalent to Italic and Regular settings when changing the font style of StateLabel in the Colors & Fonts dialog under Edit > Style . Use with property StateFont.Weight to achieve Bold Italic style. You can individually override this property with the Font.Angle property for Note objects.

Property	Type	Access	Description
StateFont. Name	String	RW	Font style (default = 'Helvetica') used for the labels of State, Box, Function, and Note objects. Enter a string for the font name (there are no selectable values). Font remains set to previous font for unrecognized font strings. Equivalent to changing the font of StateLabel in the Colors & Fonts dialog under Edit > Style .
StateFont. Size	Integer	RW	<p>Default font size for the labels of a new State, Box, Function, or Note object. Equivalent to changing the font size of StateLabel in the Colors & Fonts dialog under Edit > Style.</p> <p>You can change the font size for an existing State, Box, or Function object with the <code>FontSize</code> property of that object. You can change the font size for an existing Note object with its <code>Font.Size</code> property.</p>
StateFont. Weight	Enum	RW	<p>Font weight for state labels. Can be 'BOLD' or 'NORMAL' (default). Equivalent to the Bold and Regular settings of StateLabel in the Colors & Fonts dialog under Edit > Style. Use with the property <code>StateFont.Angle</code> to achieve Bold Italic style.</p> <p>You can individually override this property with the <code>Font.Weight</code> property for Note objects.</p>

Property	Type	Access	Description
StateLabelColor	[R,G,B]	RW	Color of the state labels for this chart in a 1-by-3 RGB array (default = [0 0 0]) with each value normalized on a scale of 0 to 1. Equivalent to changing the label color of StateLabel in the Colors & Fonts dialog under Edit > Style .
StateMachineType	Enum	RW	Type of state chart to create. Default is <i>Classic</i> , which provides the full set of Stateflow semantics. You can also create Mealy and Moore charts, which use a subset of Stateflow semantics (see “Building Mealy and Moore Charts in Stateflow”).
StrongDataTyping WithSimulink	Boolean	RW	If set to true (default = false), set strong data typing with Simulink I/O. Equivalent to selecting the Use Strong Data Typing with Simulink I/O check box in the chart properties dialog.
Tag	Any Type	RW	A field you can use to hold data of any type for this chart (default = []).
TransitionColor	[R,G,B]	RW	Set the RGB color for transitions in the Stateflow Diagram Editor in a 1-by-3 RGB array (default = [0.2902 0.3294 0.6039]) with each value normalized on a scale of 0 to 1. Equivalent to changing the Transition color in the Colors & Fonts dialog under Edit > Style .

Property	Type	Access	Description
TransitionFont. Angle	Enum	RW	Font angle for state labels. Can be 'ITALIC' or 'NORMAL' (default). Equivalent to Italic and Regular settings when you change the font style of TransitionLabel in the Colors & Fonts dialog under Edit > Style . Use with property StateFont.Weight to achieve Bold Italic style.
TransitionFont. Name	String	RW	Font style (default = 'Helvetica') used for transition labels. Enter a string for font name (there are no selectable values). Font remains set to previous font for unrecognized font strings. Equivalent to changing the font of TransitionLabel in the Colors & Fonts dialog under Edit > Style .
TransitionFont. Size	Integer	RW	Default font size (default = 12) for transition labels. Truncated to closest whole number less than or equal to entered value. Equivalent to changing the font size of TransitionLabel in the Colors & Fonts dialog under Edit > Style .
TransitionFont. Weight	Enum	RW	Font weight for transition labels. Can be 'BOLD' or 'NORMAL' (default). Equivalent to Bold and Regular settings when you change the font style of TransitionLabel in the Colors & Fonts dialog under Edit > Style . Use with property StateFont.Angle to achieve Bold Italic style.

Property	Type	Access	Description
TransitionLabel Color	[R,G,B]	RW	Color of the transition labels for this chart in a 1-by-3 RGB array (default = [0.2902 0.3294 0.6039]) with each value normalized on a scale of 0 to 1. Equivalent to changing the label color of TransitionLabel in the Colors & Fonts dialog under Edit > Style .
UserSpecifiedState TransitionExecutionOrder	Boolean	RW	If set to true (default = false), you have complete control of the order in which transitions originating from a source are tested for execution. Equivalent to selecting the User specified transition execution order check box in the chart properties dialog.
Visible	Boolean	RW	If set to true (default), display this chart in the chart diagram editor.

Chart Methods

Chart objects have the methods displayed in the table below. For details on each method, see the reference pages.

See also “Chart Properties” on page 3-18.

Method	Description
defaultTransitions	Return the default transitions in this chart at the top level of containment.
dialog	Display the properties dialog of this chart.
disp	Display the property names and their settings for this Chart object.
find	Find all objects that this chart contains that meet the specified criteria.
get	Return the specified property settings for this chart.
help	Display a list of properties for this Chart object with short descriptions.
methods	Display all nonglobal methods of this Chart object.
parse	Parse this chart.
set	Set the specified property of this Chart object with the specified value.
struct	Return and display a MATLAB structure containing the property settings of this Chart object.
view	Display this chart in a Stateflow diagram editor.

State Properties

Stateflow API objects of type State have the properties listed in the table below. See also “State Methods” on page 3-31.

Property	Type	Access	Description
ArrowSize	Double	RW	Size of transition arrows coming into this state (default = 8). Equivalent to selecting Arrowhead Size from the context menu for this state.
BadIntersection	Boolean	RO	If true, this state graphically intersects a box, graphical function, or other state.
Chart	Chart	RO	Chart object containing this state.
Debug. Breakpoints. onDuring	Boolean	RW	If set to true (default = false), set the state entry breakpoint for this chart. Equivalent to selecting the State During check box in the properties dialog for this state.
Debug. Breakpoints. OnEntry	Boolean	RW	If set to true (default = false), set the state entry breakpoint for this chart. Equivalent to selecting the State Entry check box in the properties dialog for this state.
Debug. Breakpoints. onExit	Boolean	RW	If set to true (default = false), set the state entry breakpoint for this chart. Equivalent to selecting the State Exit check box in the properties dialog for this state.
Decomposition	Enum	RW	Set this property to 'EXCLUSIVE_OR' (default) to specify exclusive (OR) decomposition for the states at the first level of containment in this state. Set to 'PARALLEL_AND' to specify parallel (AND) decomposition for these states. Equivalent to the Decomposition selection in the context menu for the state.

Property	Type	Access	Description
Description	String	RW	Description of this state (default = ''). Equivalent to entering a description in the Description field of the properties dialog for this state.
Document	String	RW	Document link to this state (default = ''). Equivalent to entering the Document Link field of the properties dialog for this state.
FontSize	Double	RW	Size of the font (default = 12) for the label text for this state. This property overrides the font size set for this state at creation by the <code>StateFont.Size</code> property of the containing Chart's object. Equivalent to selecting Font Size > in the context menu for this state.
HasOutputData	Boolean	RW	If set to true (default = false), create a data output port on the Stateflow block for this state with its activity status. If the state is active, the output value is 1. If the state is inactive, the output is 0. This is equivalent to selecting the Output State Activity check box in the State properties dialog for this state.
Id	Integer	RO	Unique identifier assigned to this state to distinguish it from other objects loaded in memory.
IsGrouped	Boolean	RW	If set to true (default = false), group this state. Nothing is allowed to change inside a grouped state. This property is also useful for copying states to a new location. See “Copying by Grouping (Recommended)” on page 1-34.

Property	Type	Access	Description
IsSubchart	Boolean	RW	If set to true (default = false), make this state a subchart.
IsTestPoint	Boolean	RW	If set to true (default = false), sets this data or state as a Stateflow test point. You can monitor individual Stateflow test points with a floating scope during model simulation. You can also log test point values into MATLAB workspace objects. See “Monitoring Stateflow Test Points” in the Stateflow User’s Guide for details.
LabelString	String	RW	Label for this state (default = '?'). Equivalent to typing the label for this state in its label text field in the diagram editor.
Machine	Machine	RO	Machine containing this state.
Name	String	RW	Name of this state (default = ''). Equivalent to typing this state’s name into the beginning of the label text field for this state in the diagram editor. Name is separated from the remainder of this state’s label text by a forward slash (/) character.
Position	Rect	RW	Position and size of this state’s box in the Stateflow chart, given in the form of a 1-by-4 array (default is [0 0 90 60]) consisting of the following: <ul style="list-style-type: none"> • (x,y) coordinates for the box’s left upper vertex relative to the upper left vertex of the Stateflow diagram editor workspace • Width and height of the box
Subviewer	Chart or State	RO	State or chart in which this state can be graphically viewed.

Property	Type	Access	Description
Tag	Any Type	RW	Holds data of any type (default = []) for this state.
Type	Enum	RO	Type of this state (default = 'OR'). Can be 'OR' (exclusive) or 'AND' (parallel). The type of this state is determined by the parent's Decomposition property.

State Methods

State objects have the methods displayed in the table below. For details on each method, see the reference pages.

See also “State Properties” on page 3-27.

Method	Description
defaultTransitions	Return the default transitions in this state at the top level of containment.
delete	Delete this state.
dialog	Display the properties dialog of this state.
disp	Display the property names and their settings for this State object.
find	Find all objects that this state contains that meet the specified criteria.
get	Return the specified property settings for this state.
help	Display a list of properties for this State object with short descriptions.
innerTransitions	Return the inner transitions that originate with this state and terminate on a contained object.
methods	Display all nonglobal methods of this State object.
outerTransitions	Return an array of transitions that exit the outer edge of this state and terminate on an object outside the containment of this state.
outputData	Output the activity status of this state to Simulink via a data output port on the chart block of this state.
set	Set the specified property of this State object with the specified value.
sourcedTransitions	Return all inner and outer transitions whose source is this state.

Method	Description
struct	Return and display a MATLAB structure containing the property settings of this State object.
view	Display this state's chart in a diagram editor with this state highlighted.

Box Properties

The following are properties of Stateflow API objects of type Box. See also “Box Methods” on page 3-35.

Property	Type	Access	Description
ArrowSize	Double	RW	Size of transition arrows coming into this box (default = 8). Equivalent to selecting Arrowhead Size from the context menu for this box.
BadIntersection	Boolean	RO	If true, this box graphically intersects a state, graphical function, or other box.
Chart	Chart	RO	Chart object containing this box.
Description	String	RW	Description of this box (default = ''). Equivalent to entering a description in the Description field of the properties dialog for this box.
Document	String	RW	Document link to this box (default = ''). Equivalent to entering the Document Link field of the properties dialog for this box.
FontSize	Double	RW	Size of the font (default = 12) for the label text of this box. This property overrides the font size set for this box at creation by the StateFont.Size property of the containing Chart's object. Equivalent to selecting Font Size > in the context menu for this box.
Id	Integer	RW	Unique identifier assigned to this box to distinguish it from other objects loaded in memory.
IsGrouped	Boolean	RW	If set to true (default = false), group this box.
IsSubchart	Boolean	RW	If set to true (default = false), make this box a subchart.

Property	Type	Access	Description
LabelString	String	RW	Label for this box (default = '?'). Equivalent to typing the label for this box in its label text field in the diagram editor.
Machine	Machine	RO	Machine that contains this box.
Name	String	RW	Name of this box (default = ''). Equivalent to typing this box's name into the beginning of the label text field for this box in the diagram editor.
Position	Rect	RW	Position and size of this box in the Stateflow chart, given in the form of a 1-by-4 array (default is [0 0 90 60]) consisting of the following: <ul style="list-style-type: none"> • (x,y) coordinates for the box's left upper vertex relative to the upper left vertex of the Stateflow diagram editor workspace • Width and height of the box
Subviewer	Chart or State	RO	State or chart in which this box can be graphically viewed.
Tag	Any Type	RW	Holds data of any type (default = []) for this box.

Box Methods

Box objects have the methods displayed in the table below. For details on each method, see the reference pages.

See also “Box Properties” on page 3-33.

Method	Description
defaultTransitions	Return the default transitions in this box at the top level of containment.
delete	Delete this box.
dialog	Display the properties dialog of this box.
disp	Display the property names and their settings for this Box object.
find	Find all objects that this box contains that meet the specified criteria.
get	Return the specified property settings for this box.
help	Display a list of properties for this Box object with short descriptions.
innerTransitions	Return the inner transitions that originate with this box and terminate on a contained object.
methods	Display all nonglobal methods of this Box object.
outerTransitions	Return an array of transitions that exit the outer edge of this box and terminate on an object outside the containment of this box.
set	Set the specified property of this Box object with the specified value.
sourcedTransitions	Return all inner and outer transitions whose source is this box.
struct	Return and display a MATLAB structure containing the property settings of this Box object.
view	Display this box’s chart in a diagram editor with this box highlighted.

Graphical Function Properties

Stateflow API objects of type Function have the properties listed in the table below. See also “Graphical Function Methods” on page 3-39.

Property	Type	Access	Description
ArrowSize	Double	RW	Size of transition arrows coming into this graphical function (default = 8). Equivalent to selecting Arrowhead Size from the context menu for this function.
BadIntersection	Boolean	RO	If true, this state graphically intersects a state, box, or other graphical function.
Chart	Chart	RO	Chart object containing this function.
Description	String	RW	Description of this function (default = ''). Equivalent to entering a description in the Description field of the properties dialog for this function.
Document	String	RW	Document link to this function. Equivalent to entering the Document Link field of the properties dialog for this function.
FontSize	Double	RW	Size of the (default = 12) font of the label text for this function. This property overrides the font size set for this function at creation by the <code>StateFont.Size</code> property of the containing Chart's object. Equivalent to selecting Font Size > in the context menu for this function.
Id	Integer	RO	Unique identifier assigned to this function to distinguish it from other objects in the model.

Property	Type	Access	Description
InlineOption	Boolean	RW	Determine how generated code for this graphical function is implemented. Possible settings are as follows: 'Inline' — Call to function is replaced by code. 'Function' — Function becomes a C function. 'Auto' — Stateflow determines if the function is inlined or made a function through an internal calculation.
IsGrouped	Boolean	RW	If set to true (default = false), group this function.
IsSubchart	Boolean	RW	If set to true (default = false), make this function a subchart.
LabelString	String	RW	Label for this function (default = ' () '). Equivalent to typing the label for this function in its label text field in the diagram editor.
Machine	Machine	RO	Machine that contains this function.
Name	String	RW	Name of this function (default = ' '). Equivalent to typing this function's name into the beginning of the label text field after the word 'function' in the diagram editor.

Property	Type	Access	Description
Position	Rect	RW	Position and size of this function's box in the Stateflow chart, given in the form of a 1-by-4 array (default is [0 0 90 60]) consisting of the following: <ul style="list-style-type: none">• (x,y) coordinates for the box's left upper vertex relative to the upper left vertex of the Stateflow diagram editor workspace• Width and height of the box
Subviewer	Chart or State	RO	State or chart in which this function can be graphically viewed.
Tag	Any Type	RW	Holds data of any type (default = []) for this function.

Graphical Function Methods

Function objects have the methods displayed in the table below. For details on each method, see the reference pages.

See also “Graphical Function Properties” on page 3-36.

Method	Description
defaultTransitions	Return the default transitions in this function at the top level of containment.
delete	Delete this function.
dialog	Display the properties dialog of this function.
disp	Display the property names and their settings for this Function object.
find	Find all objects that this graphical function contains that meet the specified criteria.
get	Return the specified property settings for this function.
help	Display a list of properties for this Function object with short descriptions.
methods	Display all nonglobal methods of this Function object.
set	Set the specified property of this Function object with the specified value.
sourcedTransitions	Return all inner and outer transitions whose source is this function.
struct	Return and display a MATLAB structure containing the property settings of this Function object.
view	Display this function’s chart in a diagram editor with this state highlighted.

Truth Table Properties

Stateflow API objects of type TruthTable have the properties listed in the table below. See also “Truth Table Methods” on page 3-43.

Property	Type	Access	Description
ActionTable	Cell Array	RW	A cell array of strings containing the contents of the Action Table for this truth table.
ArrowSize	Double	RW	Size of transition arrows coming into the truth table function in the Stateflow diagram (default = 8). Equivalent to selecting Arrowhead Size from the context menu for this function.
BadIntersection	Boolean	RO	If true, this truth table graphically intersects a state, box, graphical function, or other truth table.
Chart	Chart	RO	Chart object containing this truth table.
ConditionTable	Cell Array	RW	A cell array of strings containing the contents of the Condition Table for this truth table, including the Actions row.
Description	String	RW	Description of this truth table (default = ' '). Equivalent to entering a description in the Description field of the properties dialog for this truth table.
Document	String	RW	Document link to this truth table. Equivalent to entering the Document Link field of the properties dialog for this truth table.

Property	Type	Access	Description
FontSize	Double	RW	Size of the (default = 12) font of the label text for this truth table. This property overrides the font size set for this truth table at creation by the <code>StateFont.Size</code> property of the containing <code>Chart</code> 's object. Equivalent to selecting Font Size > <i>font size</i> in the context menu for this truth table.
Id	Integer	RO	Unique identifier assigned to this truth table to distinguish it from other objects in the model.
LabelString	String	RW	Full label for this truth table (default = '()') including its return, name, and arguments. Equivalent to typing the label for this truth table in its label text field in the diagram editor.
Machine	Machine	RO	Machine that contains this truth table.
Name	String	RW	Name of this truth table (default = ''). Equivalent to typing a name for this truth table into the label text field of the truth table box in the diagram editor. Label syntax is <i>return = Name (arguments)</i> .
OverSpecDiagnostic	String	RW	Interprets the error diagnosis of this truth table as overspecified according to the possible values 'Error', 'Warning', or 'None'. In the truth table editor, the value of this property is assigned by selecting Overspecified from the Diagnostics menu item and then selecting one of the three values.

Property	Type	Access	Description
Position	Rect	RW	<p>Position and size of this truth table's box in the Stateflow chart, given in the form of a 1-by-4 array (default is [0 0 90 60]) consisting of the following:</p> <ul style="list-style-type: none"> • (x,y) coordinates for the box's left upper vertex relative to the upper left vertex of the Stateflow diagram editor workspace • Width and height of the box
Subviewer	Chart or State	RO	State or chart in which this truth table can be graphically viewed.
Tag	Any Type	RW	Holds data of any type (default = []) for this truth table.
UnderSpecDiagnostic	String	RW	<p>Interprets the error diagnosis of this truth table as underspecified according to the possible values 'Error', 'Warning', or 'None'. In the truth table editor, the value of this property is assigned by selecting Underspecified from the Diagnostics menu item and then selecting one of the three values.</p>

Truth Table Methods

Truth table objects have the methods displayed in the table below. For details on each method, see the reference pages.

See also “Truth Table Properties” on page 3-40.

Method	Description
<code>delete</code>	Delete this truth table.
<code>dialog</code>	Display the properties dialog of this truth table.
<code>disp</code>	Display the property names and their settings for this truth table object.
<code>find</code>	Find all objects that this graphical truth table contains that meet the specified criteria.
<code>get</code>	Return the specified property settings for this truth table.
<code>help</code>	Display a list of properties for this truth table object with short descriptions.
<code>methods</code>	Display all nonglobal methods of this truth table object.
<code>set</code>	Set the specified property of this truth table object with the specified value.
<code>struct</code>	Return and display a MATLAB structure containing the property settings of this truth table object.
<code>view</code>	Display this truth table’s chart in a diagram editor with this state highlighted.

Truth Table Chart Properties

Stateflow API objects of type TruthTableChart have the properties listed in the table below. See also “Truth Table Chart Methods” on page 3-47.

Property	Type	Access	Description
ActionTable	Cell Array	RW	A cell array of strings containing the contents of the Action Table for this truth table block.
ChartUpdate	Enum	RW	Activation method of this chart. Can be 'INHERITED' (default), 'DISCRETE', or 'CONTINUOUS'.
ConditionTable	Cell Array	RW	A cell array of strings containing the contents of the Condition Table for this truth table block, including the Actions row.
Description	String	RW	Description of this truth table block (default = ''). Equivalent to entering a description in the Description field of the properties dialog for this truth table block.
Dirty	Boolean	RW	If set to true (default = false), this chart has changed since being opened or saved.
Document	String	RW	Document link to this truth table block.
Iced	Boolean	RO	Equivalent to property Locked (default = false) except that this property is used internally to lock this block from change during activities such as simulation.
Id	Integer	RO	Unique identifier assigned to this truth table block to distinguish it from other objects in the model.

Property	Type	Access	Description
InputFimath	embedded.fimath object	RW	The embedded.fimath object that will be associated with inputs from Simulink.
LabelString	String	RW	Full label for this truth table (default = '()') including its return, name, and arguments. Equivalent to typing the label for this truth table in its label text field in the diagram editor.
Locked	Boolean	RW	If set to true (default = false), mark this block as read-only and prohibit any write operations on it.
Machine	Machine	RO	Machine that contains this truth table block.
Name	String	RW	Name of this truth table block. (default = ''). Equivalent to typing a name for this truth table into the label text field of the truth table box in the diagram editor. Label syntax is <i>return = Name (arguments)</i> .
OverSpecDiagnostic	String	RW	Interprets the error diagnosis of this truth table as overspecified according to the possible values 'Error', 'Warning', or 'None'. In the truth table editor, the value of this property is assigned by selecting Overspecified from the Settings menu item and then selecting one of the three values.
Path	String	RW	Path to the block.
SampleTime	String	RW	Sample time for activating this chart (default = '').

Property	Type	Access	Description
Tag	Any Type	RW	Holds data of any type (default = []) for this truth table block.
TreatInheritedIntegersAs	String	RW	<p>Determines how inherited integer signals are treated in Embedded MATLAB.</p> <p>The two choices are <code>Integers</code> and <code>Fixed-point</code>. Simulink does not distinguish between a fixed-point signal with zero fraction length and an integer signal. However, MATLAB has two different classes for these two kinds of data: <code>Integers</code> (<code>uint8</code>, <code>int16</code>, etc...) and <code>embedded.fi</code>.</p> <p>You can specify the type for any given input signal to be either <code>Integer</code> or <code>fixed-point</code> and override this default.</p>
UnderSpecDiagnostic	String	RW	<p>Interprets the error diagnosis of this truth table as underspecified according to the possible values <code>'Error'</code>, <code>'Warning'</code>, or <code>'None'</code>. In the truth table editor, the value of this property is assigned by selecting Underspecified from the Settings menu item and then selecting one of the three values.</p>

Truth Table Chart Methods

Truth Table Chart objects have the methods displayed in the table below. For details on each method, see the reference pages.

See also “Truth Table Chart Properties” on page 3-44.

Method	Description
defaultTransitions	Return the default transitions in this object at the top level of containment.
delete	Delete this truth table.
dialog	Display the properties dialog of this truth table.
disp	Display the property names and their settings for this truth table object.
find	Find all objects that this graphical truth table contains that meet the specified criteria.
get	Return the specified property settings for this truth table.
help	Display a list of properties for this truth table object with short descriptions.
methods	Display all nonglobal methods of this truth table object.
set	Set the specified property of this truth table object with the specified value.
struct	Return and display a MATLAB structure containing the property settings of this truth table object.
view	Display this truth table’s chart in a diagram editor with this state highlighted.

Embedded MATLAB Function Properties

Stateflow API objects for Embedded MATLAB Functions have the properties listed in the table below. See also “Embedded MATLAB Function Methods” on page 3-51.

Property	Type	Access	Description
ArrowSize	Double	RW	Size of transition arrows coming into the Embedded MATLAB function in the Stateflow diagram (default = 8). Equivalent to selecting Arrowhead Size from the context menu for this function.
BadIntersection	Boolean	RO	If true, this Embedded MATLAB function graphically intersects a state, box, graphical function, truth table, or other Embedded MATLAB function.
Chart	Chart	RO	Chart object containing this Embedded MATLAB function.
Description	String	RW	Description of this Embedded MATLAB function (default = ''). Equivalent to entering a description in the Description field of the properties dialog for this Embedded MATLAB function.
Document	String	RW	Document link to this Embedded MATLAB function. Equivalent to entering the Document Link field of the properties dialog for this Embedded MATLAB function.

Property	Type	Access	Description
FontSize	Double	RW	Size of the (default = 12) font of the label text for this Embedded MATLAB function. This property overrides the font size set for this Embedded MATLAB function at creation by the StateFont.Size property of the containing Chart object. Equivalent to selecting Font Size > <i>font size</i> in the context menu for this Embedded MATLAB function.
Id	Integer	RO	Unique identifier assigned to this Embedded MATLAB function to distinguish it from other objects in the model.
LabelString	String	RW	Full label for this Embedded MATLAB function (default = '()') including its return, name, and arguments. Equivalent to typing the label for this Embedded MATLAB function in its label text field in the diagram editor.
Machine	Machine	RO	Machine that contains this Embedded MATLAB function.
Name	String	RW	Name of this Embedded MATLAB function (default = ''). Equivalent to typing a name for this Embedded MATLAB function into the label text field of the truth table box in the diagram editor. Label syntax is <i>return = Name (arguments)</i> .

Property	Type	Access	Description
Position	Rect	RW	Position and size of this Embedded MATLAB function in the Stateflow chart, given in the form of a 1-by-4 array (default is [0 0 90 60]) consisting of the following: <ul style="list-style-type: none">• (x,y) coordinates for the box's left upper vertex relative to the upper left vertex of the Stateflow diagram editor workspace• Width and height of the box
Subviewer	Chart or State	RO	State or chart in which this Embedded MATLAB function can be graphically viewed.
Tag	Any Type	RW	Holds data of any type (default = []) for this Embedded MATLAB function.

Embedded MATLAB Function Methods

Embedded MATLAB Function objects have the methods displayed in the table below. For details on each method, see the reference pages.

See also “Embedded MATLAB Function Properties” on page 3-48.

Method	Description
delete	Delete this Embedded MATLAB function.
dialog	Display the properties dialog of this Embedded MATLAB function.
disp	Display the property names and their settings for this Embedded MATLAB function object.
find	Find all objects that this Embedded MATLAB function contains that meet the specified criteria.
get	Return the specified property settings for this Embedded MATLAB functions.
help	Display a list of properties for this Embedded MATLAB function with short descriptions.
methods	Display all nonglobal methods of this Embedded MATLAB object.
set	Set the specified property of this Embedded MATLAB object with the specified value.
struct	Return and display a MATLAB structure containing the property settings of this Embedded MATLAB object.
view	Opens this Embedded MATLAB function in the Embedded MATLAB Editor .

Note Properties

Stateflow API objects of type Note have the properties listed in the table below. See also “Note Methods” on page 3-54.

Property	Type	Access	Description
Alignment	Enum	RW	Alignment of text in note box. Can be 'LEFT' (default), 'CENTER', or 'RIGHT'.
Chart	Chart	RO	Chart object containing this note.
Description	String	RW	Description of this note (default = ''). Equivalent to entering a description in the Description field of the properties dialog for this note.
Document	String	RW	Document link to this note (default = ''). Equivalent to entering the Document Link field of the properties dialog for this note.
Font. Name	String	RO	Name of the font (default = 'Helvetica') for the text in this note. This property is read-only (RO) and set by the StateFont.Name property of the Chart object containing this note.
Font. Angle	String	RW	Style of the font for the text in this note. Can be 'ITALIC' or 'NORMAL' (default). This property overrides the default style for this note, which is set by the StateFont.Angle property of the Chart object containing this note.
Font. Size	Double	RW	Size of the font (default = 12) for the label text for this note. This property overrides the font size set for this note at creation by the StateFont.Size property of the containing Chart's object. Equivalent to selecting Font Size > in the context menu for this note.
Font. Weight	String	RW	Weight of the font for the label text for this note. Can be 'BOLD' or 'NORMAL' (default). This property overrides the default weight for the text in this note, which is set by the StateFont.Weight property of the Chart object containing this note.

Property	Type	Access	Description
Id	Integer	RO	Unique identifier assigned to this note to distinguish it from other objects in the model.
Interpretation	Enum	RW	How the text in this note is interpreted for text processing. Can be 'NORMAL' (default) or 'TEX'.
Machine	Machine	RO	Machine that contains this note.
Position	Rect	RW	Position and size of this note's box in the Stateflow chart, given in the form of a 1-by-4 array (default is [0 0 25 25]) consisting of the following: <ul style="list-style-type: none"> • (x,y) coordinates for the box's left upper vertex relative to the upper left vertex of the Stateflow diagram editor workspace • Width and height of the box
Subviewer	Chart or State	RO	State or chart in which this note can be graphically viewed.
Tag	Any Type	RW	Holds data of any type (default = []) for this note.
Text	String	RW	Label for this note (default = '?'). The text content for this note that you enter directly into the note in the diagram editor or in the Label field of the properties dialog for this note.

Note Methods

Note objects have the methods displayed in the table below. For details on each method, see the reference pages.

See also “Note Properties” on page 3-52.

Method	Description
delete	Delete this note.
dialog	Display the properties dialog of this note.
disp	Display the property names and their settings for this Note object.
get	Return the specified property settings for this note.
help	Display a list of properties for this Note object with short descriptions.
methods	Display all nonglobal methods of this Note object.
set	Set the specified property of this Note object with the specified value.
struct	Return and display a MATLAB structure containing the property settings of this Note object.
view	Display this note’s chart in a diagram editor with this note highlighted.

Transition Properties

Stateflow API objects of type Transition have the properties listed in the table below. See also “Transition Methods” on page 3-59.

Property	Type	Access	Description
ArrowSize	Double	RW	Size of the arrow (default = 10) for this transition.
Chart	Chart	RO	Stateflow chart object containing this transition.
Debug. Breakpoints. WhenTested	Boolean	RW	If set to true (default = false), set a debugging breakpoint to occur when this transition is tested to see whether it is a valid transition or not. Equivalent to selecting the When Tested check box in the properties dialog of this transition.
Debug. Breakpoints. WhenValid	Boolean	RW	If set to true (default = false), set a debugging breakpoint to occur when this transition has tested as valid. Equivalent to selecting the When Valid check box in the properties dialog of this transition.
Description	String	RW	Description of this transition (default = ''). Equivalent to entering a description in the Description field of the properties dialog for this transition.
Destination	State or Junction	RW	Destination state or junction (default = []) of this transition. Assign Destination the destination object for this transition. You can also use the property Destination to detach the destination end of a transition, through the command <code>t.Destination = []</code> where <code>t</code> is the Transition object.
DestinationOClock	Double	RW	Location of transition destination connection on state (default = 0). Varies from 0 to 12 for full clock cycle location. Its value is taken as modulus 12 of its assigned value.

Property	Type	Access	Description
Document	String	RW	Document link to this transition (default = ''). Equivalent to entering the Document Link field of the properties dialog for this transition.
DrawStyle	Enum	RW	<p>Drawing style for this transition. Set to 'SMART' (default) for smart transitions or 'STATIC' for static transitions. Equivalent to selecting Smart from the context menu for this transition to toggle between settings.</p> <hr/> <p>Note Transition must be connected to effect a change in the DrawStyle property. Otherwise, an error occurs.</p> <hr/>
ExecutionOrder	Integer	RW	Specifies the number for this transition in the execution order for its source (see “Transition Testing Order” in the Stateflow and Stateflow Coder User’s Guide). The UserSpecifiedStateTransitionExecutionOrder property of the parent chart has to be set to true, otherwise, this transition property will be ignored. ExecutionOrder has to be an integer between 1 and m , where m is the total number of transitions originating from the source.
FontSize	Double	RW	Size of the font (default = 12) for the label text for this box. This property overrides the default size for this box, which is set by the TransitionFont.Size property of the Chart object containing this box. Equivalent to selecting Font Size > in the context menu for this box.
Id	Integer	RO	Unique identifier assigned to this transition to distinguish it from other objects loaded in memory.

Property	Type	Access	Description
LabelPosition	Rect	RW	<p>Position and size of this note's box in the Stateflow chart, given in the form of a 1-by-4 array (default is [0 0 8 14]) consisting of the following:</p> <ul style="list-style-type: none"> • (x,y) coordinates for the box's left upper vertex relative to the upper left vertex of the Stateflow diagram editor workspace • Width and height of the box
LabelString	String	RW	Label for this transition (default = '?'). Equivalent to typing the label for this transition in its label text field in the diagram editor.
Machine	Machine	RO	Machine containing this transition.
MidPoint	Rect	RW	Position of the midpoint of this transition relative to the upper left corner of the Stateflow diagram editor workspace in an [x y] point array (default = [0 0]).
Source	State or Junction	RW	<p>Source state or junction of this transition (default = []). Assign Source the source object for this transition.</p> <p>You can also use the property Source to detach the source end of a transition, through the command <code>t.Source = []</code> where <code>t</code> is the Transition object.</p>
SourceEndPoint	Rect	RO*	[x y] spatial coordinates for the endpoint of a transition (default = [2 2]). This property is RW (read/write) only for default transitions. For all other transitions it is RO (read-only).
SourceOClock	Double	RW	Location of transition source connection on state (default = 0). Varies from 0 to 12 for full clock cycle location. The value taken for this property is the modulus 12 of the entered value.

Property	Type	Access	Description
Subviewer	Chart or State	RO	State or chart in which this transition can be graphically viewed.
Tag	Any Type	RW	Holds data of any type (default = []) for this transition.

Transition Methods

Transition objects have the methods displayed in the table below. For details on each method, see the reference pages.

See also “Transition Properties” on page 3-55.

Method	Description
delete	Delete this transition.
dialog	Display the properties dialog of this transition.
disp	Display the property names and their settings for this Transition object.
get	Return the specified property settings for this transition.
help	Display a list of properties for this Transition object with short descriptions.
methods	Display all nonglobal methods of this Transition object.
set	Set the specified property of this Transition object with the specified value.
struct	Return and display a MATLAB structure containing the property settings of this Transition object.
view	Display this transition’s chart in a diagram editor with this transition highlighted.

Junction Properties

Stateflow API objects of type **Junction** have the properties listed in the table below. See also “Junction Methods” on page 3-61.

Property	Type	Access	Description
ArrowSize	Double	RW	Size of transition arrows (default = 8) coming into this junction.
Chart	Chart	RO	Chart that this junction resides in.
Description	String	RW	Description of this junction (default = ''). Equivalent to entering a description in the Description field of the properties dialog for this junction.
Document	String	RW	Document link to this junction (default = ''). Equivalent to entering the Document Link field of the properties dialog for this junction.
Id	Integer	RO	Unique identifier assigned to this junction to distinguish it from other objects loaded in memory.
Machine	Machine	RO	Machine containing this junction.
Position.Center	Rect	RW	Position of the center of this junction (default = [10 10]) relative to the upper left corner of the parent chart or state as an [x, y] point array.
Position.Radius	Rect	RO	Radius of this junction (default = 10).
Subviewer	Chart or State	RO	State or chart in which this junction can be graphically viewed.
Tag	Any Type	RW	Holds data of any type (default = []) for this junction.
Type	Enum	RO	Type of this junction. For junctions, can be 'CONNECTIVE' (default) or 'HISTORY'

Junction Methods

Junction objects have the methods displayed in the table below. For details on each method, see the reference pages.

See also “Junction Properties” on page 3-60.

Method	Description
delete	Delete this junction.
dialog	Display the properties dialog for this junction
disp	Display the property names and their settings for this Junction object.
get	Return the specified property settings for this junction.
help	Display a list of properties for this Junction object with short descriptions.
methods	Display all nonglobal methods of this Junction object.
set	Set the specified property of this Junction object with the specified value.
sourcedTransitions	Return all inner and outer transitions whose source is this junction.
struct	Return and display a MATLAB structure containing the property settings of this Junction object.
view	Display this junction’s chart in a diagram editor with this junction highlighted.

Data Properties

Stateflow API objects of type Data have the properties listed in the table below. See also “Data Methods” on page 3-70.

Property	Type	Access	Description
CompiledSize	String	RW	Size of data as determined by the compiler.
CompiledType	String	RW	Type of data as determined by the compiler.
Debug. Watch	Integer	RW	If set to 1 (default = 0), causes the debugger to halt execution if this data is modified. Setting this property to 1 is equivalent to selecting the Watch in debugger check box in the properties dialog for this data.
Description	String	RW	Description of this data (default = ''). Equivalent to entering a description in the Description field of the properties dialog for this data.
Document	String	RW	Document link to this data (default = ''). Equivalent to entering a link in the Document Link field of the properties dialog for this data.
Id	Integer	RO	Unique identifier assigned to this data to distinguish it from other objects loaded in memory.
InitializeMethod	String	RW	Method for initializing value of this data, based on scope of data: <ul style="list-style-type: none"> • If scope is Local or Output, you can set InitializeMethod to Expression or Parameter. Equivalent to setting the Initial value field in the properties dialog for this data. • If scope is Parameter, Input, or Data Store Memory, you should set InitializeMethod to Not Needed as a read-only property. • If scope is Constant, you should set InitializeMethod to Expression as a read-only property.

Property	Type	Access	Description
Machine	Machine	RO	Stateflow machine that contains this data.
Name	String	RW	Name of this data. Equivalent to entering the name of this data in the Name field of its properties dialog.
OutputState	Integer	RO	If set to 1 (default = 0), this data represents the activity of the state in which it is defined as an output to Simulink. See “Outputting State Activity to Simulink” in the Simulink User’s Guide. Create this data for a state by using the State method <code>outputData</code> . Equivalent to selecting the Output State Activity check box for the state.
Path	String	RO	Location of this data in the model hierarchy
Port	Integer	RW	Port index number for this data (default = 1).
Props. Array. FirstIndex	String	RW	Index of the first element of this data (default = 0) if it is an array (that is, <code>Props.Array.Size > 1</code>). Equivalent to entering a value of zero or greater in the First Index field of the properties dialog for this data.
Props. Array. Size	String	RW	Size of this data. Assigning a positive value indicates that the data is an array of the specified size (default = 0). Equivalent to entering a positive value in the Size column for this data in the Model Explorer or in the Size field of the properties dialog for this data.
Props. Initial Value	String	RW	Initial value of this data (default = ''). Equivalent to entering a value in the Initial Value column for this data in the Model Explorer or Initial Value field in the properties dialog for this data.
Props. Range. Maximum	String	RW	Maximum value (default = '') that this data can have during execution or simulation of the state machine. Equivalent to entering value in the Maximum column for this data in the Model Explorer or the Maximum field in the properties dialog for this data.

Property	Type	Access	Description
Props. Range. Minimum	String	RW	Minimum value (default = ' ') that this data can have during execution or simulation of the state machine. Equivalent to entering a value in the Minimum column for this data in Explorer or the Minimum field in the properties dialog for this data.
Props. Type. BusObject	String	RW	If Props.Method is Bus Object, you must set this property to the name of the Simulink.Bus object that defines this data (see “Working with Structures and Bus Signals in Stateflow” in the Stateflow User’s Guide). Equivalent to setting Data type mode to Bus Object and entering the name of a Simulink.Bus object in the Data type field of the properties dialog for this data.
Props. Type. Expression	String	RW	If Props.Method is Expression, you must set this property to an expression that evaluates to a data type (see “Entering Expressions and Parameters for Data Properties” in the Stateflow User’s Guide). Equivalent to setting Data type mode to Expression and entering an expression in the Data type field of the properties dialog for this data.
Props. Type. Fixpt. Bias	String	RW	The bias value for fixed-point data (default = 0.0) if Props.Type.Fixpt.ScalingMode equals Slope and bias ; otherwise this value is ignored. Equivalent to entering a real number in the Bias field of the properties dialog for this data.
Props. Type. Fixpt. FractionLength	String	RW	The location of the binary point in fixed-point data (default = 0) if Props.Type.Fixpt.ScalingMode equals Binary point ; otherwise this value is ignored. Equivalent to entering a positive or negative integer in the Fraction length field of the properties dialog for this data.

Property	Type	Access	Description
Props. Type. Fixpt. Lock	Integer	RW	If set to 1 (default = 0), prevents Simulink from replacing the current fixed-point type with a type chosen by the autoscaling tool. Equivalent to selecting Lock output scaling against changes by the autoscaling tool in the properties dialog for this data.
Props. Type. Fixpt. ScalingMode	String	RW	Method for scaling fixed point data to avoid overflow conditions and minimize quantization errors. There are three settings: <ul style="list-style-type: none"> • None (the default) • Binary point • Slope and bias,
Props. Type. Fixpt. Slope	String	RW	The slope value for fixed-point data (default = 1.0) if Props.Type.Fixpt.ScalingMode equals Slope and bias ; otherwise this value is ignored. Equivalent to entering a positive real number in the Slope field in the properties dialog for this data.

Property	Type	Access	Description
Props. Type. Method	String	RW	<p>Method for setting the type of this data, based on scope:</p> <ul style="list-style-type: none"> • If scope is Local, you can set this property to Built-in, Expression, Bus Object, or Fixed point. • If scope is Constant, you can set this property to Built-in, Expression, or Fixed point. • If scope is Parameter, you can set this property to Inherited, Built-in, Expression, or Fixed point. • If scope is Input or Output, you can set this property to Inherited, Built-in, Expression, Bus Object, or Fixed point. • If scope is Data Store Memory, you should set Props.Type.Method to Inherited as a read-only property. <p>Equivalent to setting the Data type mode field in the properties dialog for this data.</p>
Props. Type. Signed	Integer	RW	<p>If set to 1, indicates that fixed-point data should be signed. The default value is 0, indicating that the data is unsigned. Equivalent to selecting the Signed check box in the properties dialog for this data.</p>
Props. Type. Units	String	RW	<p>Unit of measurement for the data value (default = '').</p>
Props. Type. WordLength	String	RW	<p>Size in bits of the word that will hold the quantized integer of fixed-point data. Equivalent to entering an integer between 0 and 32 in the Word length field of the properties dialog for this data.</p>

Property	Type	Access	Description
Props. SaveToWorkspace	Integer	RW	If set to 1, assigns the value of the data item to a variable of the same name in the model's base workspace at the end of simulation. The default value is 0, indicating that the data is not saved to the base workspace. Equivalent to selecting the check box Save final value to base workspace in the properties dialog for this data.

Property	Type	Access	Description
Scope	Enum	RW	<p>Scope of this data:</p> <ul style="list-style-type: none"> • Local — Data defined in current Stateflow chart. • Constant — Read-only constant value that is visible to the parent Stateflow object and its children. • Parameter — Constant defined in the MATLAB workspace or derived from a Simulink parameter that is defined and initialized in the parent masked subsystem. • Input — If the parent is a graphical, truth table, or Embedded MATLAB function, the data is an input argument. Otherwise, it is provided by the Simulink model to the Stateflow chart via an input port. • Output — If the parent is a graphical, truth table, or Embedded MATLAB function, the data is a return value. Otherwise, it is provided by the Stateflow chart to the Simulink model via an output port. • Data Store Memory — Data that binds to a Simulink data store. • Temporary — Data that persists only during the execution of a function. • Imported — Data parented by the Simulink model, but defined in external code embedded in the Stateflow machine. • Exported — Data from the Simulink model that you provide to external code defined in Stateflow. Parent must be a Stateflow machine. <p>Equivalent to selecting a scope from the menu in the Scope field of the properties dialog for this data. See “Scope” in the Stateflow User’s Guide.</p>

Property	Type	Access	Description
SaveToWorkspace	Integer	RW	If set to 1 (default = 0), this data is saved to the MATLAB workspace. Setting this property to 1 is equivalent to selecting the SaveToWorkspace column entry for this data in the Model Explorer or selecting the Save final value to base workspace field in the properties dialog for this data.
TestPoint	Integer	RW	If set to 1 (default = 0), sets this data as a Stateflow test point. You can monitor individual Stateflow test points with a floating scope during model simulation. You can also log test point values into MATLAB workspace objects. See “Monitoring Stateflow Test Points” in the Stateflow User’s Guide.
Units	String	RW	Units of measurement for the value of this data (default = '').

Data Methods

Data objects have the methods displayed in the table below. For details on each method, see the reference pages.

See also “Data Properties” on page 3-62.

Method	Description
delete	Delete this data.
dialog	Display the properties dialog of this Data object.
disp	Display the property names and their settings for this Data object.
get	Return the specified property settings for this data.
help	Display a list of properties for this Data object with short descriptions.
methods	Display all nonglobal methods of this Data object.
set	Set the specified property of this Data object with the specified value.
struct	Return and display a MATLAB structure containing the property settings of this Data object.
view	Display this data in the Data properties dialog.

Event Properties

Stateflow API objects of type Event have the properties listed in the table below. See also “Event Methods” on page 3-74.

Property	Type	Access	Description
Debug. Breakpoints. StartBroadcast	Boolean	RW	If set to true (default = false), set a debugger breakpoint for the start of the broadcast of this event. Equivalent to selecting the Start of broadcast check box in the properties dialog for this event.
Debug. Breakpoints. EndBroadcast	Boolean	RW	If set to true (default = false), set a debugger breakpoint for the end of the broadcast of this event. Equivalent to selecting the End of broadcast check box in the properties dialog for this event.
Description	String	RW	Description of this event (default = ''). Equivalent to entering a description in the Description field of the properties dialog for this event.
Document	String	RW	Document link to this event (default = ''). Equivalent to entering the Document Link field of the properties dialog for this event.
Id	Integer	RO	Unique identifier assigned to this event to distinguish it from other objects loaded in memory.
Machine	Machine	RO	Machine this event belongs to.

Property	Type	Access	Description
Name	String	RW	Name of this event (default = event n , where n is a counter of events with the name root event). Equivalent to entering the name in the Name field of the properties dialog for this event.
Port	Integer	RO	Port index number for this event (default = 1).
Scope	Enum	RW	<p>Scope of this event. Allowed values vary with the object containing this data.</p> <p>The following applies to any event:</p> <ul style="list-style-type: none"> • 'Local' <p>The following apply to events for charts only:</p> <ul style="list-style-type: none"> • 'input' (Input from Simulink in properties dialog) • 'Output' (Output to Simulink in properties dialog) <p>The following apply to events for machines only:</p> <ul style="list-style-type: none"> • 'Imported' • 'Exported'

Property	Type	Access	Description
Tag	Any Type	RW	Holds data of any type (default = []) for this event.
Trigger	Enum	RW	<p>Type of signal that triggers this chart input event. Also the type of trigger associated with this chart output event. Equivalent to the entries for the Trigger field in the Event dialog for this event.</p> <p>The following triggers apply to both chart input and output events:</p> <ul style="list-style-type: none"> • 'Either' (Either Edge) • 'Function call' (Function Call) <p>The following triggers apply only to chart input events:</p> <ul style="list-style-type: none"> • 'Rising' (Rising Edge) • 'Falling' (Falling Edge)

Event Methods

Event objects have the methods displayed in the table below. For details on each method, see the reference pages.

See also “Event Properties” on page 3-71.

Method	Description
delete	Delete this event.
dialog	Display the properties dialog for this event.
disp	Display the property names and their settings for this Event object.
get	Return the specified property settings for this event.
help	Display a list of properties for this Event object with short descriptions.
methods	Display all nonglobal methods of this Event object.
set	Set the specified property of this Event object with the specified value.
struct	Return and display a MATLAB structure containing the property settings of this Event object.
view	Display this event in its properties dialog.

Target Properties

Stateflow API objects of type Target have the properties listed in the table below. See also “Target Methods” on page 3-80.

Property	Type	Access	Description
ApplyToAllLibs	Boolean	RW	If set to true (default), use settings in this target for all libraries. Equivalent to selecting the Use settings for all libraries check box in this target’s Target Builder dialog.
CodeFlagsInfo	Array	RO	A MATLAB vector of structures containing information on the code flag settings for this target. See “CodeFlagsInfo Property of Targets” on page 3-77 for more information.
CodegenDirectory	String	RW	Directory to receive generated code (default = ''). Equivalent to the entry in the Generated Code Directory panel of the Target Options dialog for this target. Applies only to targets other than sfun and rtw targets.
CustomCode	String	RW	Custom code included at the top of the generated code (default = ''). Equivalent to the entry in the Include Code panel of the Target Options dialog for this target.
CustomInitializer	String	RW	Custom initialization code (default = ''). Equivalent to the entry in the Initialization Code panel of the Target Options dialog for this target. Applies only to sfun and rtw targets.

Property	Type	Access	Description
CustomTerminator	String	RW	Custom termination code (default = ''). Equivalent to the entry in the Termination Code panel of the Target Options dialog for this target. Applies only to sfun and rtw targets.
Description	String	RW	Description of this target (default = ''). Equivalent to entering a description in the Description field of the properties dialog for this target.
Document	String	RW	Document link to this target (default = ''). Equivalent to entering the Document Link field of the properties dialog for this target.
Id	Integer	RO	Unique identifier assigned to this Target object to distinguish it from other objects loaded in memory.
Machine	Machine	RO	Stateflow machine containing this target.
Name	String	RW	Name of this target (default = 'untitled'). Equivalent to naming or renaming this target in the Explorer.
ReservedNames	String	RW	Comma- or space-separated list of names to not use in Stateflow generated code. Equivalent to the entry in the Reserved Names panel of the Target Options dialog.
Tag	Any Type	RW	Holds data of any type (default = []) for this target.

Property	Type	Access	Description
UserIncludeDirs	String	RW	Custom include directory paths (default = ''). Equivalent to the entry in the Include Paths panel of the Target Options dialog for this target.
UserLibraries	String	RW	Custom libraries (default = ''). Equivalent to the entry in the Libraries panel of the Target Options dialog for this target.
UserSources	String	RW	Custom source files (default = ''). Equivalent to the entry in the Source Files panel of the Target Options dialog for this target.

CodeFlagsInfo Property of Targets

The CodeFlagsInfo property of a Target object is a read-only MATLAB vector of structures containing information on the code flag settings for its target. Each element in the vector has the following MATLAB structure of information about a particular code flag:

Element	Type	Description
name	String	Short name for this flag
type	String	The type of the code flag
description	String	A description of this code flag
defaultValue	Boolean	The default value of this code flag upon creation of its target
visible	Boolean	Whether or not this flag is visible
enable	Boolean	Whether or not to enable this flag
value	Boolean	The value of the flag

The first element of each structure is a shorthand name for the individual flag that you set in the **Coder Options** dialog. For example, the name

'comments' actually refers to the dialog setting **Comments in generated code**. While the CodeFlagsInfo property is informational only, you can use these shorthand flag names in the methods getCodeFlag and setCodeFlag to access and change the values of a flag.

The names of each of the possible code flags in the CodeFlagsInfo property along with the name of the flag as it appears in the **Coder Options** dialog for the target are as follows:

Name in CodeFlagsInfo	Name in Properties Dialog	Target	Default Value
debug	Enable debugging/animation	sfun	Enabled
overflow	Enable overflow detection (with debugging)	sfun	Enabled
echo	Echo expressions without semicolons	sfun	Enabled
comments	Comments in generated code	rtw, custom	Disabled Enabled
preservenames	Preserve symbol names	rtw, custom	Disabled
preservenameswithparent	Append symbol names with no mangling	rtw, custom	Disabled
exportcharts	Use chart names with no mangling	rtw, custom	Disabled
statebitsets	Use bitsets for storing state configuration	rtw, custom	Disabled
databitsets	Use bitsets for storing boolean data	rtw, custom	Disabled

Name in CodeFlagsInfo	Name in Properties Dialog	Target	Default Value
ioformat	Enumerated value can be one of the following: <ul style="list-style-type: none"> • 0 = Use global input/output data • 1 = Pack input/output data into structures • 2 = Separate argument for input/output data 	custom	0
initializer	Generate chart initializer function	custom	Disabled
multi instanced	Multi-instance capable code	custom	Disabled
ppcomments	Comments for Post-processing	custom	Disabled

For detailed descriptions of each of the preceding code flags, see “Configuring a Custom Target in Stateflow” in the Stateflow User’s Guide.

Target Methods

Target objects have the methods displayed in the table below. For details on each method, see the reference pages.

See also “Target Properties” on page 3-75.

Method	Description
build	Build this target incrementally for only those portions of the target's charts that have changed since the last build.
delete	Delete this target.
dialog	Display the properties dialog for this target.
disp	Display the property names and their settings for this Target object.
get	Return the specified property settings for this target.
getCodeFlag	Return the value of the specified code flag for this target.
help	Display a list of properties for this Target object with short descriptions.
make	Compile this target incrementally for only those portions of generated code that have changed since the last compilation. For a simulation target (sfun), compile a shared library (S-function MEX-file) from the generated code.
methods	Display all nonglobal methods of this Target object.
rebuildAll	Completely rebuild this target.
regenerateAll	Completely regenerate code for this target.
set	Set the specified property of this Target object with the specified value.
setCodeFlag	Set the specified code flag for this target with the specified value.
struct	Return and display a MATLAB structure containing the property settings of this Target object.
view	Display this target in the Target properties dialog.

API Methods — Alphabetical List

build

Purpose Build this target incrementally

Syntax `thisTarget.build`

Description The `build` method incrementally builds this target. It performs the following activities:

- Parses all charts completely.
- Generates code for charts incrementally.
- For a simulation target (`sfun`), compiles a shared library (MEX-file) from the generated code.

If a complete build has already taken place, this method builds only those portions of the target corresponding to charts that have changed since the last build.

Arguments `thisTarget` The Target object to build.

Returns None

Example If `t` is a Target object, the command `t.build` builds the target for the Stateflow charts that have changed in the target's model since the last build and/or code generation.

See Also The methods `rebuildAll`, `generate`, `regenerateAll`, and `make`

Syntax `thisTarget.build`

Purpose	Provide a handle to the schema class of this object's type		
Syntax	<code>handle = thisObject.classhandle</code>		
Description	The <code>classhandle</code> method returns a read-only handle to the schema class of this object's type. You can use the <code>classhandle</code> method to provide information about the structure of each object type.		
Arguments	<table><tr><td><code>thisObject</code></td><td>The object for which to return a handle. Can be any Stateflow object.</td></tr></table>	<code>thisObject</code>	The object for which to return a handle. Can be any Stateflow object.
<code>thisObject</code>	The object for which to return a handle. Can be any Stateflow object.		
Returns	<table><tr><td><code>handle</code></td><td>Handle to schema class of this object.</td></tr></table>	<code>handle</code>	Handle to schema class of this object.
<code>handle</code>	Handle to schema class of this object.		

Example

If `j` is a Junction object, the class handle of a Junction object is `j.classhandle`. You can see the class schema for a Junction object by using the following get command:

```
j.classhandle.get
```

Two member arrays of the displayed class schema are `Properties` and `Methods`. These two members are members of the schema class for every object.

List the class schema for `Properties` with the following command:

```
j.classhandle.Properties.get
```

Two displayed members of the `Properties` schema are `Name` and `DataType`. Finally, using the class handle for a junction, you can display the properties of a Junction object along with their data types with the following command:

```
get(j.classhandle.Properties,{'Name','DataType'})
```

copy

Purpose	Copy the specified array of objects to the clipboard						
Syntax	<code>cbObj.copy(objArray)</code>						
Description	<p>The copy method copies the specified objects to the clipboard. Objects to copy are specified through a single argument array of objects.</p> <p>Later, complete the copy operation by invoking the <code>pasteTo</code> method.</p>						
Arguments	<table><tr><td><code>cbObj</code></td><td>The Clipboard object to copy to.</td></tr><tr><td><code>objArray</code></td><td>Array of Stateflow objects to copy. These objects must conform to the following:</td></tr><tr><td></td><td><ul style="list-style-type: none">• The objects copied must be all graphical (states, boxes, functions, transitions, junctions) or all nongraphical (data, events, targets).• If all objects are graphical, they must all be seen in the same subviewer.</td></tr></table>	<code>cbObj</code>	The Clipboard object to copy to.	<code>objArray</code>	Array of Stateflow objects to copy. These objects must conform to the following:		<ul style="list-style-type: none">• The objects copied must be all graphical (states, boxes, functions, transitions, junctions) or all nongraphical (data, events, targets).• If all objects are graphical, they must all be seen in the same subviewer.
<code>cbObj</code>	The Clipboard object to copy to.						
<code>objArray</code>	Array of Stateflow objects to copy. These objects must conform to the following:						
	<ul style="list-style-type: none">• The objects copied must be all graphical (states, boxes, functions, transitions, junctions) or all nongraphical (data, events, targets).• If all objects are graphical, they must all be seen in the same subviewer.						
Returns	None						
Example	See “Copying Objects” on page 1-33.						

Purpose	Return the default transitions in this object at the top level of containment		
Syntax	<code>defaultTransitions = thisObject.defaultTransitions</code>		
Description	The <code>defaultTransitions</code> method returns the default transitions in this object at the top level of containment.		
Arguments	<table><tr><td><code>thisObject</code></td><td>The object for which to return default transitions. Can be an object of type <code>Chart</code>, <code>State</code>, <code>Box</code>, or <code>Function</code>.</td></tr></table>	<code>thisObject</code>	The object for which to return default transitions. Can be an object of type <code>Chart</code> , <code>State</code> , <code>Box</code> , or <code>Function</code> .
<code>thisObject</code>	The object for which to return default transitions. Can be an object of type <code>Chart</code> , <code>State</code> , <code>Box</code> , or <code>Function</code> .		
Returns	<table><tr><td><code>defaultTransitions</code></td><td>Array of default transitions in this object at the top level of containment.</td></tr></table>	<code>defaultTransitions</code>	Array of default transitions in this object at the top level of containment.
<code>defaultTransitions</code>	Array of default transitions in this object at the top level of containment.		
Example	If state A contains state A1, and state A1 contains state A11, and states A1 and A11 have default transitions attached to them, the <code>defaultTransitions</code> method of state A returns the default transition attached to state A1.		

delete

Purpose Delete this object

Syntax `thisObject.delete`

Description The delete method deletes this object from the model. This is true for all but objects of type Root, Chart, Clipboard, and Editor.

Arguments `thisObject` The object to delete. Can be an object of type Machine, State, Box, Function, Truth Table, Note, Transition, Junction, Data, Event, or Target.

Returns None

Example If a state A is represented by the State object `sA`, the command `sA.delete` deletes state A.

Purpose Open the Properties dialog of this object

Syntax `thisObject.dialog`

Description The dialog method opens the Properties dialog of its object.

Arguments

<code>thisObject</code>	The object for which to open the properties dialog. Can be an object of type Machine, State, Box, Function, Truth Table, Note, Transition, Junction, Data, Event, or Target.
-------------------------	--

Returns None

Example If state A is represented by State object `sA`, the MATLAB statement `sA.dialog` opens the Properties dialog for state A.

disp

Purpose	Display the properties and settings for this object		
Syntax	<code>thisObject.disp</code>		
Description	The <code>disp</code> method displays the properties and settings for this object. This is true for all but objects of type <code>Root</code> and <code>Clipboard</code> .		
Arguments	<table><tr><td><code>thisObject</code></td><td>The object to display properties and settings for. Can be an object of type <code>Machine</code>, <code>Chart</code>, <code>State</code>, <code>Box</code>, <code>Function</code>, <code>Truth Table</code>, <code>Note</code>, <code>Transition</code>, <code>Junction</code>, <code>Data</code>, <code>Event</code>, or <code>Target</code>.</td></tr></table>	<code>thisObject</code>	The object to display properties and settings for. Can be an object of type <code>Machine</code> , <code>Chart</code> , <code>State</code> , <code>Box</code> , <code>Function</code> , <code>Truth Table</code> , <code>Note</code> , <code>Transition</code> , <code>Junction</code> , <code>Data</code> , <code>Event</code> , or <code>Target</code> .
<code>thisObject</code>	The object to display properties and settings for. Can be an object of type <code>Machine</code> , <code>Chart</code> , <code>State</code> , <code>Box</code> , <code>Function</code> , <code>Truth Table</code> , <code>Note</code> , <code>Transition</code> , <code>Junction</code> , <code>Data</code> , <code>Event</code> , or <code>Target</code> .		
Returns	None		
Example	If a state A is represented by the <code>State</code> object <code>sA</code> , the command <code>sA.disp</code> displays the property names and their settings for state A.		

Purpose Return specified objects in this object at all levels of containment

Syntax `objArray = thisObject.find(Specifier, Value, ...)`

Note You can also nest specifications using braces ({}).

Description Using combinations of specifier-value argument pairs, the `find` method returns objects in this object that match the specified criteria. The specifier-value pairs can be property based or based on other attributes of the object such as its depth of containment. Specifiers can also be logical operators (`-and`, `-or`, etc.) that combine other specifier-value pairs.

By default, the `find` command finds objects at all depths of containment within an object. You can specify the maximum depth of search with the `-depth` specifier. However, the zeroth level of containment, i.e., the searched object itself, is always included if it happens to satisfy the search criteria.

If no arguments are specified, the `find` command returns all objects of this object at all levels of containment.

Arguments

<code>thisObject</code>	The object for which to find contained objects. Can be an object of type <code>Root</code> , <code>Machine</code> , <code>State</code> , <code>Box</code> , <code>Function</code> , or <code>Truth Table</code> .
<code>'-and'</code>	No value is paired to this specifier. Instead, this specifier relates a previous specifier-value pair to a following specifier-value pair in an AND relation.

find

- '-class' String class name of the class to search for. Use this option to find all objects whose class exactly matches a given class. To allow matches for subclasses of a given class, use the `-isa` specifier. Classes are specified as the string name (e.g., `'Stateflow.State'`, `'Stateflow.Transition'`, etc.) or as a handle to the class (see method `classhandle`).
- '-depth' Integer depth to search, which can be 0,1,2,...,infinite. The default search depth is infinite.

Note Do not use the `'-depth'` switch with the `find` method for a machine object.

- '-function' Handle to a function that evaluates each object visited in the search. The function must always return a logical scalar value that indicates whether or not the value is a match. If no property is specified, the function is passed the handle of the current object in the search. If a property is specified, the function is passed the value of that property.

In the following example, a function with handle `f` (defined in first line) is used to filter a `find` to return only those objects of type `'andState'`:

```
f = @(h) (strcmp(get(h,'type'), 'andState'));  
objArray = thisObject.find('-function', f);
```

- '-isa' Name of the type of objects to search for. Object types are specified as a string name (e.g., `'Stateflow.State'`, `'Stateflow.Transition'`, etc.) or as a handle to the object type (see method `classhandle`).

- '-method' String that specifies the name of a method belonging to the objects to search for.
- '-not' No value is paired to this specifier. Instead, this specifier searches for the negative of the following specifier-value pair.
- '-or' No value is paired to this specifier. Instead, this specifier relates the previous specifier-value pair to the following specifier-value pair in an OR relation.

Note If no logical operator is specified, -or is assumed.

- '*property*' The specifier takes on the name of the property. Value is the string value of the specified property for the objects you want to find.
- '-property' String name of the property that belongs to the objects you want to find.
- '-xor' No value is paired to this specifier. Instead, this specifier relates the previous specifier-value pair to the following specifier-value pair in an XOR relation.
- '-regex' No value follows this specifier. Instead, this specifier indicates that the value of the following specifier-value pair contains a regular expression.

Returns

objArray Array of objects found matching the criteria specified.

Example

If a Chart object *c* represents a Stateflow chart, the command `states=c.find('-isa','Stateflow.State')` returns an array, `states`, of all the states in the chart, and the command `states=c.find('Name','A')` returns an array of all objects whose Name property is 'A'.

find

If state A, which is represented by State object sA, contains two states, A1 and A2, and you specify a find command that finds all the states in A as follows,

```
states= sA.find( '-isa','Stateflow.State')
```

then the above command finds three states: A, A1, and A2.

Purpose Generate code incrementally for this target

Syntax `thisTarget.generate`

Description The `generate` method generates code incrementally for this target. If a complete code generation has already taken place, it performs an incremental generation for only those portions of the target corresponding to charts that have changed since the last code generation.

Arguments `thisTarget` The target for which to generate code.

Returns None

Example If `t` is a `Target` object, the command `t.generate` generates code for the Stateflow charts that have changed in the target's model since the last code generation.

See Also The methods `build`, `rebuildAll`, `regenerateAll`, and `make`

get

Purpose Return a MATLAB structure containing the property settings of this object or an array of objects

Syntax `propList = thisObject.get(prop)`

Description The `get` method returns and displays a MATLAB structure containing the settings for the specified property of this object. If no property is specified, the settings for all properties are returned.

The `get` method is also vectorized so that it returns an m -by- n cell array of values for an array of m objects and an array of n properties.

Arguments

<code>thisObject</code>	The object for which to get specified property.
<code>prop</code>	String name of property (e.g., 'FontSize') to get value for. Can also be an array of properties (see return <code>propList</code> below). If no property is specified, a list of all properties is returned.

Returns

<code>propList</code>	MATLAB structure listing the properties of this object. Can also be an m by n cell array of values if <code>thisObject</code> is an array of m objects and <code>prop</code> is an array of n properties.
-----------------------	---

Example State A is represented by the State object `sA`.

The following command lists the properties of state A:

```
sA.get
```

The following command returns a handle to a MATLAB structure of the properties of state A to the workspace variable `Aprops`:

```
Aprops = sA.get
```

Purpose Return the specified code flag

Syntax `thisTarget.getCodeFlag(name)`

Description The `getCodeFlag` method returns the value of a particular code flag whose name you specify.

Arguments

<code>thisTarget</code>	The target for which to get code flag value
<code>name</code>	The short string name of the code flag for which to get value. See “CodeFlagsInfo Property of Targets” on page 3-77 for a list of these names.

Returns None

Example Assume that the Target object `x` represents the simulation target `sfun` for the loaded model. If `m` is the Stateflow machine object for this model, you can obtain `x` with the following command:

```
x = m.find('-isa','Stateflow.Target','-and','Name','sfun')
```

The simulation target has two code flags: `debug` and `echo`. You can verify this by looking at the `CodeFlagsInfo` property of `x`. See the description of this property in “Target Properties” on page 3-75 for more information.

In the Stateflow user interface, the `debug` code flag is enabled or disabled through the **Enable debugging/animation** check box in the **Coder Options** dialog. By default, this flag is turned on for the simulation target. You can verify this with the following command:

```
t.getCodeFlag('debug')
```

getCodeFlag

Similarly, you can check the value of the echo code flag, which is enabled or disabled through the **Echo expressions without semicolons** check box of the same dialog, with the following command:

```
t.getCodeFlag('echo')
```

See Also

The method `setCodeFlag`

Purpose	Display the list of properties for this object along with accompanying descriptions
Syntax	<code>thisObject.help</code>
Description	The <code>help</code> method returns a list of properties for any object. To the right of this list appear simple descriptions for each property. Some properties do not have descriptions because their names are descriptive in themselves.
Arguments	None
Returns	None
Example	If <code>j</code> is an API handle to a Stateflow junction, the command <code>j.help</code> returns a list of the property names and descriptions for a Stateflow API object of type Junction.

innerTransitions

Purpose Return the inner transitions that originate with this chart or state and terminate on a contained object

Syntax `transitions = thisObject.innerTransitions`

Description The `innerTransitions` method returns the inner transitions that originate with this object and terminate on a contained object.

Arguments None

Returns

- `thisObject` Object for which to get inner transitions. Can be of type `State` or `Box`.
- `transitions` Array of inner transitions originating with this object and terminating on a contained state or junction.

Example State A contains state A1, and state A1 contains state A11. State A has two transitions, each originating from its inside edge and terminating inside it. These are inner transitions. One transition terminates with state A1 and the other terminates with state A11. The `innerTransitions` method of state A returns both of these transitions.

Purpose	Incrementally compile this target with no code generation
Syntax	<code>thisTarget.make</code>
Description	For a simulation target (<code>sfun</code>), compiles a shared library (S-function MEX-file) from the generated code. This method incrementally compiles the target, but does not generate code. It compiles only those portions of generated code that have changed since the last compilation.
Arguments	<code>thisTarget</code> The target for which to do make.
Returns	None
Example	Suppose you define <code>t</code> as a target object for a Simulink model <code>mytest</code> , which contains a Stateflow chart. The command <code>t.make</code> incrementally compiles generated code for that target. However, if <code>t</code> is a simulation target (<code>sfun</code>), <code>t.make</code> compiles the generated code into a shared library (S-function MEX-file). MEX-file extensions vary according to platform (see “Using MEX-Files” in the MATLAB External Interfaces documentation). For example, on a 32-bit Windows machine, the <code>t.make</code> command in this example generates a MEX-file called <code>mytest_sfun.mexw32</code> for the simulation target <code>t</code> .
See Also	The methods <code>build</code> , <code>rebuildAll</code> , <code>generate</code> , and <code>regenerateAll</code>

methods

Purpose List the names of the methods belonging to this object

Syntax `thisObject.methods`

Description The methods method lists the names of the methods belonging to this object.

Note The methods method for this object displays some internal methods that are not applicable to Stateflow use, and are not documented. These are as follows: `areChildrenOrdered`, `getChildren`, `getDialogInterface`, `getDialogSchema`, `getDisplayClass`, `getDisplayIcon`, `getDisplayLabel`, `getFullName`, `getHierarchicalChildren`, `getPreferredProperties`, `isHierarchical`, `isLibrary`, `isLinked`, `isMasked`.

Arguments

<code>thisObject</code>	Object for which to list methods. Can be of any Stateflow object type.
-------------------------	--

Returns None

Example If state A is represented by State object `sA`, the command `sA.methods` lists the methods of state A.

Purpose Return an array of outer transitions for this object

Syntax `transitions = thisObject.outerTransitions`

Description The `outerTransitions` method returns an array of transitions that exit the outer edge of this object and terminate on objects outside the containment of this object.

Arguments None

Returns

<code>thisObject</code>	The object for which to find outer transitions. Can be of object type <code>State</code> or <code>Box</code> .
<code>transitions</code>	An array of transitions exiting the outer edge of this state.

Example A chart contains three states, A, B, and C. State A is connected to state B through a transition from state A to state B. State B is connected to state C through a transition from state B to state C. And state C is connected to state A through a transition from state C to state A. If state A is represented by State object handle `sA`, the command `sA.outerTransitions` returns the transition from state A to state B.

outputData

Purpose Create, retrieve, or delete a data output to Simulink of this state's activity status

Syntax `StateData = thisState.outputData (action)`

Description The `outputData` method of this state creates, retrieves, or deletes a special data object of type `State`. This data is attached internally to an output port on this state's Stateflow block in Simulink to output the activity status of this state to Simulink during run-time.

Note You cannot use the Stateflow Explorer to create Data objects of type `State`.

Arguments

<code>thisState</code>	The state object for which to add a special port.
<code>action</code>	This string value can be one of the following: <ul style="list-style-type: none">• 'create' — Returns a new data object of type <code>State</code> and attaches it internally to a new state activity output port on this state's Stateflow block.• 'get' — Returns this state's existing data object of type <code>State</code> attached internally to an existing state activity output port on this state's Stateflow block.• 'delete' — Deletes this state's data object of type <code>State</code> and the state activity output port on its Stateflow block to which it is attached.

Returns `StateData` The data object of type `State` for this state

Example

If state A is represented by State object `sA`, the following command creates a new data object of type `State`, which is output to Simulink and contains state A's activity:

```
s.outputData('create')
```

The Stateflow Chart block in Simulink that contains state A now has an output port labeled A, the name of state A. In Explorer, state A now contains a data object of type `State` whose scope is `Output` to Simulink.

The following command returns a `Data` object, `d`, for the data output to Simulink containing state A's activity:

```
s.outputData('get')
```

The following command deletes the data output to Simulink containing state A's activity:

```
s.outputData('delete')
```

parse

Purpose For Chart objects, parse this chart; for Machine objects, parse the charts in this machine

Syntax `thisChart.parse`
`thisMachine.parse`

Description For Chart objects, the parse method parses this chart. This is equivalent to selecting **Parse** from the **Tools** menu of the Stateflow diagram editor for this chart.

For Machine objects, the parse method parses all the charts in this machine.

Arguments

<code>thisChart</code>	The chart to parse.
<code>thisMachine</code>	The machine containing charts to parse.

Returns None

Example If `ch` is a handle to an API object representing a chart, then the command `ch.parse` parses the chart.

Purpose	Paste the objects in the Clipboard to the specified container object		
Syntax	<code>clipboard.pasteTo(newContainer)</code>		
Description	The paste method pastes the contents of the Clipboard to the specified container object. The receiving container is specified through a single argument. Use of this method assumes that you placed objects in the Clipboard with the copy method.		
Arguments	<table><tr><td><code>newContainer</code></td><td>The Stateflow object to receive a copy of the contents of the Clipboard object. If the objects in the Clipboard are all graphical (states, boxes, functions, notes, transitions, junctions), this object must be a chart or subchart.</td></tr></table>	<code>newContainer</code>	The Stateflow object to receive a copy of the contents of the Clipboard object. If the objects in the Clipboard are all graphical (states, boxes, functions, notes, transitions, junctions), this object must be a chart or subchart.
<code>newContainer</code>	The Stateflow object to receive a copy of the contents of the Clipboard object. If the objects in the Clipboard are all graphical (states, boxes, functions, notes, transitions, junctions), this object must be a chart or subchart.		
Returns	None		
Example	See the section “Copying Objects” on page 1-33.		

rebuildAll

Purpose Completely rebuild this target

Syntax `thisTarget.rebuildAll`

Description The `rebuildAll` method completely rebuilds this target with the following actions:

- Parses all charts completely.
- Regenerates code for all charts completely.
- For a simulation target (`sfun`), compiles a shared library (S-function MEX-file) from the generated code.

Arguments `thisTarget` The Stateflow target to rebuild.

Returns None

Example If `t` is a Target object, the command `t.rebuildAll` completely rebuilds that target.

See Also The methods `build`, `generate`, `regenerateAll`, and `make`

Purpose	Completely regenerate code for this target
Syntax	<code>thisTarget.regenerateAll</code>
Description	The <code>regenerateAll</code> method regenerates this target. Regardless of previous code generations, it regenerates code for all charts in this target's model.
Arguments	<code>thisTarget</code> The Stateflow target for which to regenerate code.
Returns	None
Example	If <code>t</code> is a Target object, the command <code>t.regenerateAll</code> completely regenerates code for the Stateflow charts in that target's model.
See Also	The methods <code>build</code> , <code>rebuildAll</code> , <code>generate</code> , and <code>make</code>

set

Purpose Set specified properties with the specified values

Syntax `thisObject.set(propName,value,...)`

Note Arguments can consist of an indefinite number of property (name, value) pairs.

Description The set method sets the value of a specified property or sets the values of a set of specified properties for this object. You specify properties and values through pairs of property (name, value) arguments.

The get method is also vectorized so that it sets an m-by-n cell array of values for an array of m objects and an array of n properties.

Arguments

<code>thisObject</code>	The object for which the specified property is set. Can be any Stateflow object.
<code>propName</code>	String name of the property to set (e.g., 'FontSize'). Can also be a cell array of m property names.
<code>value</code>	New value for the specified property. Can be a cell array of m-by-n values if <code>thisObject</code> is an array of m objects and <code>propName</code> is an array of n property names.

Returns None

Example The following command sets the Name and Description properties of the State object s:

```
s.set('Name', 'Kentucky', 'Description', 'Bluegrass State')
```


The following command sets the Position property of the State object s:

```
s.set('Position', [200, 119, 90, 60])
```

setCodeFlag

Purpose Set the specified code flag to the value you specify

Syntax `thisTarget.setCodeFlag(name,value)`

Description The `setCodeFlag` method sets the value of a code flag whose name you specify.

Arguments

<code>thisTarget</code>	Target object for which to set code flag.
<code>name</code>	String name of code flag. See “CodeFlagsInfo Property of Targets” on page 3-77 for a list of these names.
<code>value</code>	Value of code flag. Can be of any type.

Flag values can vary in type. Use the property `CodeFlagsInfo` to obtain the type for a particular flag.

Returns None

Example Assume that the Target object `x` represents the simulation target `sfun` for the loaded model. If `m` is the Stateflow machine object for this model, you can obtain `x` with the following command:

```
x = m.find('-isa','Stateflow.Target','-and','Name','sfun')
```

The simulation target has two code flags: `debug` and `echo`. You can verify this by looking at the `CodeFlagsInfo` property of `x` with the following command:

```
x.CodeFlagsInfo.name
```

In the Stateflow user interface the `debug` code flag is enabled or disabled through the **Enable debugging/animation** check box in the **Coder Options** dialog. By default, this flag is turned on (`==1`) for the simulation target, which you can verify with the following command:

```
t.getCodeFlag('debug')
```

If you want to disable debugging, enter the following command:

```
t.setCodeFlag('debug',0)
```

See Also

The method `getCodeFlag`

sourcedTransitions

Purpose Return the transitions that have this object as their source

Syntax `transitions = thisObject.sourcedTransitions`

Description The `sourcedTransitions` method returns all inner and outer transitions that have their source in this object.

Arguments `transitions` Source object of the returned transitions. Can be of type `State`, `Box`, `Function`, or `Junction`.

Returns `transitions` Array of all transitions whose source is this object

Example Suppose that a chart contains three states, A, B, and state A1, which is contained by state A. The chart also has three transitions: one from A to B labeled AtoB, one from B to A labeled BtoA, and one from the inner edge of A to its state A1 (inner transition) labeled AtoA1. If State object `sA` represents state A, the command `sA.sourcedTransitions` returns two transitions: the outer transition labeled AtoB and the inner transition labeled AtoA1.

Purpose Constructor for creating a box

Syntax `box_new = Stateflow.Box(parent)`

Description The `Stateflow.Box` method is a constructor method for creating boxes in a parent chart, state, box, or function, that returns a handle to an Event object for the new function.

Arguments

`parent` Handle to an object for the parent chart, state, box, or function of the new box

Returns `box_new` Handle to the Box object for the new box

Example If `sA` is a handle to a State object for an existing state A, the following command creates a new box parented (contained by) state A:

```
box_new = Stateflow.Box(sA)
```

The new box is unnamed and appears in the upper left-hand corner inside state A. `box_new` is a handle to a Box object for the new box.

Stateflow.Data

Purpose Constructor for creating a data

Syntax `data_new = Stateflow.Data(parent)`

Description The `Stateflow.Data` method is a constructor method for creating data for a parent machine, chart, state, box, or function, that returns a handle to the Data object for the new data.

Arguments `parent` Handle to an object for the parent machine, chart, state, box, or function of the new data

Returns `data_new` Handle to the Data object for the new data

Example If `sA` is a handle to a State object for an existing state A, the following command creates a new data parented (contained by) state A:

```
data_new = Stateflow.Data(sA)
```

The new data is named 'data' with an incremented integer suffix to distinguish additional creations. `data_new` is a handle to the Data object for the new data.

Purpose

Constructor for creating an Embedded MATLAB function

Syntax

```
efunction_new = Stateflow.EMFunction(parent)
```

Description

The `Stateflow.EMFunction` method is a constructor method for creating an Embedded MATLAB function in a parent chart, state, box, or graphical function. It returns a handle to the `EMFunction` object for the new Embedded MATLAB function.

Arguments

`parent` Handle to parent chart or state of the new Embedded MATLAB function

Returns

`efunction_new` Handle to a Function object for the new Embedded MATLAB function

Example

If `sA` is a handle to a State object for the existing state A, the following command creates a new Embedded MATLAB function parented (contained by) state A:

```
efunction_new = Stateflow.EMFunction(sA)
```

The new Embedded MATLAB function is unnamed and appears in the upper left corner inside of state A in the diagram editor. `efunction_new` is a handle to the `EMFunction` object, which you use to rename the function, set its properties, and execute its methods.

Stateflow.Event

Purpose Constructor for creating an event

Syntax `event_new = Stateflow.Event(parent)`

Description The `Stateflow.Event` method is a constructor method for creating an event for a parent machine, chart, state, box, or function, that returns a handle to an Event object for the new event.

Arguments `parent` Handle to parent machine, chart, state, box, or function of new event

Returns `event_new` Handle to the Event object for the new event

Example If `sA` is a handle to a State object for an existing state A, the following command creates a new event parented (contained by) state A:

```
event_new = Stateflow.Event(sA)
```

The new event is named 'event' with an incremented suffix to distinguish additional creations. `event_new` is a handle to an Event object for the new event that you use to rename the event, set its properties, and execute Event methods for the event.

Purpose	Constructor for creating a function
Syntax	<code>function_new = Stateflow.Function(parent)</code>
Description	The <code>Stateflow.Function</code> method is a constructor method for creating functions in a parent chart, state, box, or function, that returns a handle to a <code>Function</code> object for the new function.
Arguments	<code>parent</code> Handle to parent chart or state of the new function
Returns	<code>function_new</code> Handle to a <code>Function</code> object for the new function
Example	<p>If <code>sA</code> is a handle to a <code>State</code> object for the existing state A, the following command creates a new function parented (contained by) state A:</p> <pre>function_new = Stateflow.Function(sA)</pre> <p>The new function is unnamed and appears in the upper left corner inside of state A in the diagram editor. <code>function_new</code> is a handle to the <code>Function</code> object for the new function that you use to rename the function, set its properties, and execute its methods.</p>

Stateflow.Junction

Purpose Constructor for creating a junction

Syntax `junc_new = Stateflow.Junction(parent)`

Description The `Stateflow.Junction` method is a constructor method for creating a junction in a parent chart, state, box, or function, that returns a handle to the `Junction` object for the new junction.

Arguments `parent` Handle to the object for the parent chart, state, box, or function of the new junction

Returns `junc_new` Handle to the `Junction` object for new junction

Example If `sA` is a handle to a `State` object for the existing state A, the following command creates a new junction parented (contained by) state A:

```
junc_new = Stateflow.Junction(sA)
```

The new junction appears in the middle of state A in the diagram editor. `junc_new` is a handle to the `Junction` object for the new junction that you use to set its properties, and execute its methods.

Purpose	Constructor for creating a note		
Syntax	<code>note_new = Stateflow.Note(parent)</code>		
Description	The <code>Stateflow.Note</code> method is a constructor method for creating notes for a parent chart, state, box, or function, that returns a handle to the Note object for the new note.		
Arguments	<table><tr><td><code>parent</code></td><td>Handle to the object for the parent chart, or subchart for the new note</td></tr></table>	<code>parent</code>	Handle to the object for the parent chart, or subchart for the new note
<code>parent</code>	Handle to the object for the parent chart, or subchart for the new note		
Returns	<table><tr><td><code>note_new</code></td><td>Handle to the Note object for the newly created note</td></tr></table>	<code>note_new</code>	Handle to the Note object for the newly created note
<code>note_new</code>	Handle to the Note object for the newly created note		
Example	<p>If <code>sA</code> is a handle to a State object for the existing state A, the following command creates a new note parented (contained by) state A:</p> <pre>note_new = Stateflow.Note(sA)</pre> <p>The new note is placed in the upper left-hand corner of state A in the diagram editor, but is invisible because it has no text content. <code>note_new</code> is a handle to the Note object for the new note, that you use to set its text content with a command like the following:</p> <pre>note_new.Text = 'This is a note'</pre>		

Stateflow.State

Purpose Constructor for creating a state

Syntax `state_new = Stateflow.State(parent)`

Description The `Stateflow.State` method is a constructor method for creating a state for a parent chart, state, box, or function, that returns a handle to the State object for the new state.

Arguments `parent` Handle to the object for the parent chart, state, box, or function for the new state

Returns `state_new` Handle to State object for newly created state

Example If `sA` is a handle to a State object for the existing state A, the following command creates a new state parented (contained by) state A:

```
state_new = Stateflow.State(sA)
```

The new state appears in the upper left-hand corner of state A in the diagram editor. `state_new` is a handle to the State object for the new state that you use to rename the state, set its properties, and execute its methods.

Purpose Constructor for creating a target

Syntax `target_new = Stateflow.Target(parent_m)`

Description The `Stateflow.Target` method is a constructor method for creating a target for a parent machine, that returns a handle to the Target object for the new target.

Arguments `parent_m` Handle to object for the parent machine of the new target

Returns `target_new` Handle to the Target object for the newly created target

Example The following command creates a new target for the machine with the Machine object whose handle is `pm`:

```
target_new = Stateflow.Target(pm)
```

The preceding command creates a custom target with name `untitled`. `target_new` is a handle to the Target object of the new target which you can use to rename and set properties for the target. The following command renames the new target to `rtw`, thus making it the Real-Time Workshop® (RTW) target for its parent machine:

```
target_new.Name = 'rtw'
```

Stateflow.Transition

Purpose Constructor for creating a transition

Syntax `transition_new = Stateflow.Transition(parent)`

Description The `Stateflow.Transition` method is a constructor method for creating transitions in a parent chart, state, box, or function that returns a handle to a Transition object for the new transition.

Arguments

<code>parent</code>	Handle to parent chart, state, box, or function of new transition
---------------------	---

Returns `transition_new` Handle to Transition object for the new transition

Example If `sA` is a handle to a State object for the existing state A, the following command creates a new transition parented by state A:

```
transition_new = Stateflow.Transition(sA)
```

The new transition is unlabeled and appears in the upper left corner of the chart in the diagram editor. `transition_new` is a handle to the Transition object for the new transition that you use to rename the transition, set its properties, and execute its methods.

Purpose

Constructor for creating a truth table

Syntax

```
truth_table_new = Stateflow.TruthTable(parent)
```

Description

The `Stateflow.TruthTable` method is a constructor method for creating truth tables in a parent chart, state, box, or function, that returns a handle to a Truth Table object for the new truth table.

Arguments

`parent` Handle to parent chart or state of new truth table

Returns

`truth_table_new` Handle to Truth Table object for new truth table

Example

If `sA` is a handle to a State object for the existing state A, the following command creates a new truth table parented (contained by) state A:

```
truth_table_new = Stateflow.TruthTable(sA)
```

The new truth table is unnamed and appears in the upper left corner inside of state A in the diagram editor. `truth_table_new` is a handle to the Truth Table object for the new truth table that you use to rename the truth table, set its properties, and execute its methods.

struct

Purpose Return a MATLAB structure containing the property settings of this object

Syntax `propList = thisObject.struct`

Description The `struct` method returns and displays a MATLAB structure containing the property settings of this object.

Note You can change the values of the properties in this structure just as you would a property of the object. However, the MATLAB structure is not a Stateflow object and changing it does not affect the Stateflow model.

Arguments `transitions` The object for which to display property settings. Can be any Stateflow object type.

Returns `propList` MATLAB structure listing the properties of this object

Example If State object `sA` represents a state A, the command `x = sA.struct` returns a MATLAB structure `x`. You can use dot notation on `x` to report properties or set the values of other variables. For example, the command `y=x.Name` sets the MATLAB variable `y` to the value of the `Name` property of state A, which is 'A'. The command `x.Name = 'Kansas'` sets the `Name` property of `x` to 'Kansas' but does not change the `Name` property of state A.

Purpose	Return the parent object of this object		
Syntax	<code>parentObject = thisObject.up</code>		
Description	The up method returns a handle to the object that contains an this object.		
Arguments	<table><tr><td><code>thisObject</code></td><td>Object for which to return parent (containing) object</td></tr></table>	<code>thisObject</code>	Object for which to return parent (containing) object
<code>thisObject</code>	Object for which to return parent (containing) object		
Returns	<table><tr><td><code>parentObject</code></td><td>Object containing thisObject</td></tr></table>	<code>parentObject</code>	Object containing thisObject
<code>parentObject</code>	Object containing thisObject		
Example	<p>Assume that a Stateflow diagram has two states, A and B, and state A contains state B. If the object sB represents the state B, then the command</p> <pre>p = sB.up</pre> <p>returns a handle p to the parent of B, which is state A.</p>		

view

Purpose Make this object visible for editing

Syntax `thisObject.view`

Description The view method opens the object in its appropriate editing environment as follows:

- For Chart objects, the view method opens the chart in a diagram editor, if it is not already open, and brings it to the foreground.
- For State, Box, Function, Note, Junction, and Transition objects, the view method does the following:
 - a** Opens the chart containing the object in a diagram editor if it is not already open.
 - b** Highlights the object.
 - c** Zooms the object's diagram editor to the level of full expanse of the object's containing state or chart.
 - d** Brings the diagram editor for this object to the foreground.
- For Truth Table objects, the view method opens the truth table editor for this truth table:
- For Event, Data, and Target objects, the view method opens the Explorer window.

Arguments `thisObject` Object for which to display editing environment. Can be an object of type Chart, State, Box, Function, Truth Table, Note, Transition, Junction, Event, Data, or Trigger.

Returns None

Purpose Zoom in or out on this chart

Syntax `thisChart.zoomIn`
`thisChart.zoomOut`

Description The methods `zoomIn` and `zoomOut` cause the Stateflow diagram editor window for this chart to zoom in or zoom out, respectively, by 20 percentage points.

Note The `zoomIn` and `zoomOut` methods do not open or give focus to the Stateflow diagram editor for this chart.

Arguments `thisChart` Chart object to zoom in or out on.

Returns None

Example If the Chart object `ch` represents a Stateflow chart at the zoom level of 100%, the command `ch.zoomIn` raises the zoom level to 120%.

A

- accessing existing objects (API)
 - with the find method 1-27
- API
 - See Stateflow API 1-3

B

- BadIntersection property (API) 1-25
- behavioral properties and methods (API) 2-25
- Box object (API)
 - description 1-6
 - methods 3-35
 - properties 3-33
- build method (API) 4-2

C

- Chart object (API)
 - accessing 1-10
 - create new objects in 1-11
 - methods 3-26
 - open 1-11
 - properties 3-18
- classhandle method (API) 4-3
- Clipboard object (API)
 - connecting to 1-37
 - copying 1-33
 - description 1-6
 - methods 3-10
- connecting to
 - Clipboard object (API) 1-37
 - Editor object (API) 1-37
 - Stateflow objects (API) 1-23
- constructor for Stateflow objects (API) 1-23
- containment of Stateflow objects 1-25
- copy method (API) 4-4
 - features and limitations 1-33
- copying objects (API)
 - by grouping (recommended) 1-34

- copy method 1-33
- Data, Event, and Target objects 1-35
- individual objects 1-35
- overview 1-33
- using the Clipboard object 1-33

create (API)

- handle to Stateflow objects (API) 1-23
- new model and chart (API) 1-9
- new objects in chart (API) 1-11
- new state (API) 1-11
- object containment 1-25
- Stateflow objects (API) 1-23
- transition (API) 1-12

D

- Data object (API)
 - methods 3-70
 - properties 3-62
- default transitions
 - creating in API 1-40
- defaultTransitions method (API) 4-5
- delete method (API) 4-6 4-8
 - example 1-26
- deployment properties and methods (API) 2-30
- destroying Stateflow objects (API) 1-26
- dialog method (API) 4-7
- disp method (API) 4-8
- displaying
 - enumerated values for properties (API) 1-22
 - properties and methods (API) 1-20
 - subproperties (API) 1-21
- dot (.) notation (API)
 - nesting 1-18

E

- Editor object (API)
 - connecting to 1-37
 - description 1-6

- graphical changes 1-37
- methods (API) 3-9
- properties 3-8
- Event object (API)
 - methods 3-74
 - properties 3-71

F

- find method (API) 4-9
 - examples 1-9 to 1-10
 - how to use 1-27
- function notation for API methods 1-18
- Function object (API)
 - description 1-6
 - methods 3-39
 - properties 3-36

G

- generate method (API) 4-13
- get method (API) 4-14
 - examples 1-20
 - getting and setting properties of objects 1-31
- getCodeFlag method (API) 4-15
- graphical properties and methods (API) 2-12

H

- help method (API) 4-17
 - example 1-20

I

- innerTransitions method (API) 4-18

J

- Junction object (API)
 - properties 3-60

L

- labels
 - multiline labels using API 1-39
- listing
 - enumerated values for properties (API) 1-22
 - properties and methods (API) 1-20
 - subproperties (API) 1-21

M

- Machine object (API)
 - accessing 1-9
 - description 1-6
 - methods 3-17
 - properties 3-13
- make method (API) 4-19
- MATLAB
 - API scripts 1-43
- methods (API)
 - description of 1-7
 - displaying 1-20
 - function notation 1-18
 - naming 1-17
 - nesting 1-18
 - of Box object 3-35
 - of Chart object 3-26
 - of Clipboard object 3-10
 - of Data object 3-70
 - of Editor object 3-9
 - of Event object 3-74
 - of Function object 3-39
 - of Machine object 3-17
 - of Note object 3-54
 - of State object 3-31
 - of Transition object 3-59
 - of Truth Table object 3-43 3-51
- methods method (API) 4-20
 - example 1-20

N

- naming of properties and methods (API) 1-17
- Note object (API)
 - methods 3-54
 - properties (API) 3-52

O

- objects (API)
 - copying 1-33
 - getting and setting properties 1-31
- outerTransitions method (API) 4-21
- outputData method (API) 4-22
- overlapping object edges 1-25

P

- parse method (API) 4-24
- pasteTo method (API) 4-25
- properties (API)
 - description of 1-7
 - displaying 1-20
 - displaying enumerated values for 1-22
 - displaying subproperties 1-21
 - getting and setting 1-31
 - naming 1-17
 - nesting 1-18
 - of Box object 3-33
 - of Chart object 3-18
 - of Data object 3-62
 - of Editor object 3-8
 - of Event object 3-71
 - of Function object 3-36
 - of Junction object 3-60
 - of Machine object 3-13
 - of Note object 3-52
 - of State object 3-27
 - of Target object 3-75
 - of Transition object 3-55
 - of Truth Table object 3-40 3-48

- properties and methods (API)
 - behavioral 2-25
 - deployment 2-30
 - graphical 2-12
 - structural 2-21
 - utility and convenience 2-35

Q

- Quick Start
 - Stateflow API 1-9

R

- rebuildAll method (API) 4-26
- regenerateAll method (API) 4-27
- Root object (API)
 - access 1-9
 - description 1-5

S

- saving
 - Simulink model (API) 1-16
- script of API commands 1-43
- set method (API) 4-28
- setCodeFlag method (API) 4-30
- sfclipboard method (API)
 - example 1-37
- sourcedTransitions method (API) 4-32
- State object (API)
 - description 1-6
 - methods 3-31
 - properties 3-27
- Stateflow API
 - Box object 1-6
 - Chart object (API), accessing 1-10
 - Clipboard object 1-6
 - common properties and methods 1-7
 - create new model and chart 1-9
 - Editor object (API) 1-6

- Function object 1-6
- Machine object 1-6
- Machine object (API), access 1-9
- methods of objects 1-7
- naming and notation 1-17
- object hierarchy 1-4
- open chart 1-11
- overview 1-3
- properties of objects 1-7
- Quick Start 1-9
- references to properties and methods 1-8
- Root object 1-5 1-9
- State object 1-6
 - unique properties and methods 1-7
- Stateflow.State method (API) 4-33 to 4-41 4-43
- states
 - create (API) 1-11
 - label, multiline (API) 1-39
- struct method (API) 4-44
- structural properties and methods (API) 2-21
- supertransitions
 - working with in the API 1-41

T

- Target object (API)

- properties 3-75
- transition labels
 - multiline (API) 1-39
- Transition object (API)
 - labels, multiline 1-39
 - methods 3-59
 - properties 3-55
- transitions
 - create (API) 1-12
 - default transitions (API) 1-40
 - supertransitions in the API 1-41
- Truth Table object (API)
 - methods 3-43 3-51
 - properties 3-40 3-48

U

- utility and convenience properties and methods (API) 2-35

V

- view method (API) 4-45 to 4-46

Z

- zoomIn and zoomOut methods (API) 4-47