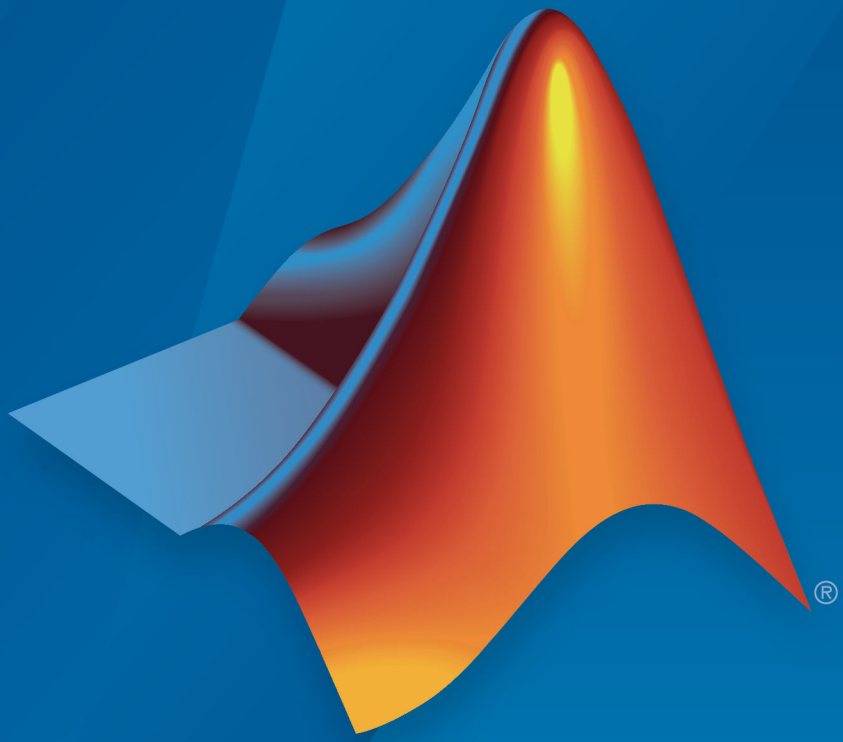
Stateflow®

API



MATLAB® & SIMULINK®

R2019b

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

Technical support: `www.mathworks.com/support/contact_us`

Phone: 508-647-7000

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

Stateflow® API

Revision History

June 2004	Online only	Revised for Version 6.0 (Release 14)
October 2004	Online only	Revised for Version 6.1 (Release 14SP1)
March 2005	Online only	Revised for Version 6.2 (Release 14SP2)
September 2005	Online only	Revised for Version 6.3 (Release 14SP3)
March 2006	Online only	Revised for Version 6.4 (Release 2006a)
September 2006	Online only	Revised for Version 6.5 (Release 2006b)
September 2007	Online only	Rereleased for Version 7.0 (Release 2007b)
March 2008	Online only	Revised for Version 7.1 (Release 2008a)
October 2008	Online only	Revised for Version 7.2 (Release 2008b)
March 2009	Online only	Revised for Version 7.3 (Release 2009a)
September 2009	Online only	Revised for Version 7.4 (Release 2009b)
March 2010	Online only	Revised for Version 7.5 (Release 2010a)
September 2010	Online only	Revised for Version 7.6 (Release 2010b)
April 2011	Online only	Revised for Version 7.7 (Release 2011a)
September 2011	Online only	Revised for Version 7.8 (Release 2011b)
March 2012	Online only	Revised for Version 7.9 (Release 2012a)
September 2012	Online only	Revised for Version 8.0 (Release 2012b)
March 2013	Online only	Revised for Version 8.1 (Release 2013a)
September 2013	Online only	Revised for Version 8.2 (Release 2013b)
March 2014	Online only	Revised for Version 8.3 (Release 2014a)
October 2014	Online only	Revised for Version 8.4 (Release 2014b)
March 2015	Online only	Revised for Version 8.5 (Release 2015a)
September 2015	Online only	Revised for Version 8.6 (Release 2015b)
October 2015	Online only	Rereleased for Version 8.5.1 (Release 2015aSP1)
March 2016	Online only	Revised for Version 8.7 (Release 2016a)
September 2016	Online only	Revised for Version 8.8 (Release 2016b)
March 2017	Online only	Revised for Version 8.9 (Release 2017a)
September 2017	Online only	Revised for Version 9.0 (Release 2017b)
March 2018	Online only	Revised for Version 9.1 (Release 2018a)
September 2018	Online only	Revised for Version 9.2 (Release 2018b)
March 2019	Online only	Revised for Version 10.0 (Release 2019a)
September 2019	Online only	Revised for Version 10.1 (Release 2019b)

Contents

1 Using the API

Overview of the Stateflow API **1-2**
- Stateflow API Object Hierarchy **1-2**
- Get a Handle on Stateflow API Objects **1-4**
- Access API Object Properties and Methods **1-5**

Create Charts by Using the Stateflow API **1-7**
- Create a New Model and Chart **1-7**
- Access the Machine Object **1-7**
- Access the Chart Object **1-8**
- Create New Objects in the Chart **1-9**

Access Properties and Methods of Stateflow Objects **1-14**
- Naming Conventions for Properties and Methods **1-14**
- Using Dot Notation with Properties and Methods **1-14**
- Access Methods Using Function Notation **1-15**

Display Properties and Methods of Stateflow Objects **1-16**
- Display Properties **1-16**
- Display Names of Methods **1-16**
- Display Property Subproperties **1-17**
- Display Enumerated Values for Properties **1-17**

Create and Destroy Stateflow Objects **1-19**
- About Creating and Destroying API Objects **1-19**
- Create Stateflow Objects **1-19**
- Establish the Parent (Container) of an Object **1-21**
- Destroy Stateflow Objects **1-22**

Access Existing Stateflow Objects **1-23**
- About Stateflow Object Handles **1-23**
- Find Objects and Properties **1-23**
- Find Objects at Different Levels of Containment **1-24**

Retrieve Recently Selected Objects 1-26
Get and Set the Properties of Objects 1-27

Move Stateflow Graphical Objects 1-28
How to Move Objects Programmatically 1-28
Move a Subcharted State 1-28
Rules for Moving Objects Programmatically 1-29

Copy and Paste Stateflow Objects 1-30
Access the Clipboard Object 1-30
copy Method Limitations 1-30
Copy by Grouping 1-31
Copy Objects Individually 1-32

View Stateflow Graphical Objects 1-33
Objects You Can Zoom 1-33
Zoom States in a Chart 1-33

Modify the Graphical Properties of Your Chart 1-36
About Editor Objects 1-36
Access the Editor Object 1-36
Change the Display in the Stateflow Editor 1-36

Enter Multiline Labels in States and Transitions 1-38

Create Default Transition Objects 1-39

Create Supertransition Objects 1-41

Create Charts by Using a MATLAB Script 1-43

2 API Object Reference

Properties and Methods Sorted by Stateflow Object 2-2
Constructor Methods 2-2
Root Object 2-3
Stateflow.Annotation 2-4
Stateflow.AtomicBox 2-7
Stateflow.AtomicSubchart 2-10

Stateflow.Box 2-13
Stateflow.Chart 2-15
Stateflow.Clipboard 2-23
Stateflow.Data 2-23
Stateflow.Editor 2-30
Stateflow.EMFunction 2-30
Stateflow.Event 2-33
Stateflow.Function 2-35
Stateflow.Junction 2-38
Stateflow.Machine 2-40
Stateflow.Message 2-42
Stateflow.SimulinkBasedState 2-46
Stateflow.SLFunction 2-51
Stateflow.State 2-53
Stateflow.StateTransitionTableChart 2-58
Stateflow.Transition 2-66
Stateflow.TruthTable 2-69
Stateflow.TruthTableChart 2-72

3 API Object Properties and Methods

Properties and Methods Sorted By Application 3-2
Access 3-3
Creation and Deletion 3-19
Debugging 3-21
Display Control 3-26
Graphical Appearance 3-31
Identifiers 3-40
Interface with Simulink 3-46
Logging 3-50
Specification 3-52

4 API Method Reference

Using the API

- “Overview of the Stateflow API” on page 1-2
- “Create Charts by Using the Stateflow API” on page 1-7
- “Access Properties and Methods of Stateflow Objects” on page 1-14
- “Display Properties and Methods of Stateflow Objects” on page 1-16
- “Create and Destroy Stateflow Objects” on page 1-19
- “Access Existing Stateflow Objects” on page 1-23
- “Move Stateflow Graphical Objects” on page 1-28
- “Copy and Paste Stateflow Objects” on page 1-30
- “View Stateflow Graphical Objects” on page 1-33
- “Modify the Graphical Properties of Your Chart” on page 1-36
- “Enter Multiline Labels in States and Transitions” on page 1-38
- “Create Default Transition Objects” on page 1-39
- “Create Supertransition Objects” on page 1-41
- “Create Charts by Using a MATLAB Script” on page 1-43

Overview of the Stateflow API

In this section...
“Stateflow API Object Hierarchy” on page 1-2 “Get a Handle on Stateflow API Objects” on page 1-4 “Access API Object Properties and Methods” on page 1-5

The Stateflow Application Programming Interface (API) is a tool you use to create or change Stateflow charts through MATLAB commands. By placing Stateflow API commands in a MATLAB script, you can automate chart editing processes in a single command.

Applications for the Stateflow API include:

- Creating a script that performs common graphical edits and simplifies editing of Stateflow charts
- Creating a script that creates a repetitive "base" Stateflow chart
- Creating a script that produces a specialized report of your model

The Stateflow API consists of objects that represent graphical and nongraphical objects of a Stateflow chart. For example, API object of type `State` and `Transition` represent states and transitions in a Stateflow chart. The correspondence between API objects and objects in a chart is bidirectional. When you modify the property of an API object or call one of its methods, you affect the corresponding object in the Stateflow chart. When you use the Stateflow Editor to perform an operation on an object in the chart, you affects the corresponding API object.

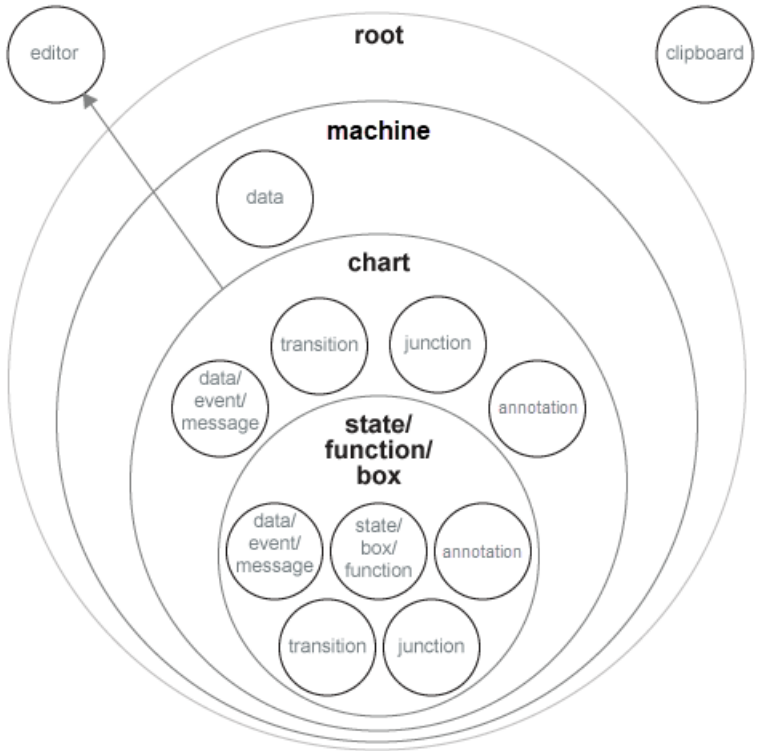
Note You cannot undo any operation in the Stateflow Editor that you perform using the Stateflow API. If you perform an editing operation through the API, the undo and redo buttons are disabled from undoing and redoing any prior operations.

Stateflow API Object Hierarchy

Stateflow API objects contain or are contained by other API objects. For example, if state `A` contains state `B` in a Stateflow chart, then the API object for state `A` contains the API object for state `B`. Rules of containment define the Stateflow hierarchy and the Stateflow

API object hierarchy. For example, charts can contain states but states cannot contain charts. For more information, see “Overview of Stateflow Objects”.

This diagram shows the Stateflow API hierarchy of objects.



The Stateflow API hierarchy consists of these layers of containment:

- **Root** — The `Root` object is the parent of all Stateflow API objects. It is a placeholder at the top of the Stateflow API hierarchy to distinguish Stateflow objects from Simulink® model objects. You automatically create the `Root` object when you load a Simulink model containing a Stateflow chart or call the function `sfnew` to create a model with a Stateflow chart.
- **Machine** — The Stateflow machine contains all the charts in a Simulink model. `Machine` objects are equivalent to Simulink models from a Stateflow perspective. All

`Machine` objects are contained in the `Root` object. `Machine` objects can hold `Chart` objects that represent Stateflow charts, state transition tables, and truth table blocks.

- **Chart** — Within any `Machine` object there can be any number of `Chart` objects. Each object of type `Chart` can contain objects that represent states, transitions, junctions, boxes, functions, annotations, data, events, and messages. These objects represent the components of a Stateflow chart.
- **States, Functions, and Boxes** — Nested within objects of type `State`, `Function`, and `Box`, there can be other objects that represent states, transitions, junctions, boxes, functions, annotations, data, events, and messages. Levels of nesting can continue indefinitely.

The hierarchy diagram shows two object types that exist outside of the Stateflow containment hierarchy:

- **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 Stateflow Editor. For more information, see “Modify the Graphical Properties of Your Chart” on page 1-36.
- **Clipboard** — The `Clipboard` object has two methods, `copy` and `pasteTo`, that use the clipboard as a staging area to implement copy and paste functionality in the Stateflow API. For more information, see “Copy and Paste Stateflow Objects” on page 1-30.

Get 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 you need in the Stateflow API is a handle to the `Root` object, which is the parent of all objects in the Stateflow API. In this command, the function `sfroot` returns a handle to the `Root` object:

```
rt = sfroot
```

Once you have a `Root` object handle, you can find a handle to the Model object for the Stateflow machine with which you want to work. Once you have a handle to a `Machine` object, you can find a handle to a `Chart` object for the chart you want to edit. Later, when

you create objects or find existing objects in a Stateflow chart, you receive a handle to the object that you can use to manipulate the actual object in the Stateflow Editor.

To learn how to use API object handles to create and edit Stateflow charts, see “Create Charts by Using the Stateflow API” on page 1-7.

Access 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 Object Properties

API properties correspond to values that you normally set for an object through the user interface of the Stateflow 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 Editor, you can click-drag the source, end, or midpoint of a transition to change its position.

API Object Methods

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 Editor. For example, you can delete a transition in the Stateflow Editor by calling the `delete` method of the `Transition` object that represents the transition. Deleting a transition in the Stateflow Editor is normally done by selecting a transition and pressing the **Delete** key.

Common API Properties and Methods

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.

Unique API Properties and Methods

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 Stateflow Editor. A `Data` object, however, has no `Position` property.

See Also

`delete` | `get` | `set` | `sfroot`

More About

- “Create Charts by Using the Stateflow API” on page 1-7
- “Properties and Methods Sorted by Stateflow Object” on page 2-2
- “Properties and Methods Sorted By Application” on page 3-2

Create Charts by Using the Stateflow API

In this section...
“Create a New Model and Chart” on page 1-7
“Access the Machine Object” on page 1-7
“Access the Chart Object” on page 1-8
“Create New Objects in the Chart” on page 1-9

Create a New Model and Chart

1. Close all models.
2. Type the function `sfnew` to create a new, untitled model with a new Stateflow chart in it.

MATLAB is the default action language of a chart you create with `sfnew`. To open a new C chart or to change the default action language, see “Modify the Action Language for a Chart”.

You have only one Simulink model in memory. Do not open the chart. You can now access the API `Machine` object that represents the model itself.

Access the Machine Object

In the Stateflow API, each model you create or load into memory is represented by an object of type `Machine`. Before accessing the Stateflow chart you created in the previous section, you must first connect to its `Machine` object. However, in the Stateflow API, all `Machine` objects are contained by the Stateflow API `Root` object, so you must use the `Root` object returned by the function `sfroot` to access a `Machine` object:

1. Use this command to obtain a handle to the `Root` object:

   ```
   rt = sfroot;
   ```

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

   ```
   m = rt.find('-isa','Stateflow.Machine');
   ```

If, instead of one model, there are several models open, this command returns an array of different `Machine` objects that you can access through indexing (`m(1)`,`m(2)`, etc.) You

can identify a specific `Machine` 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 `Machine` object named **`myMachine`** with this command:

```
m = rt.find('-isa', 'Stateflow.Machine', '-and', ...
'Name', 'myMachine');
```

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

Access the Chart Object

In “Access the Machine Object” on page 1-7, you accessed the `Machine` object containing your new chart to return a handle to the `Machine` object for your new model, `m`. Perform these steps to access the new chart:

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

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

   In the preceding command, the `find` method of the `Machine` 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, `ch(1)`, `ch(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 `Machine` object handle, `m`.

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

2. Open the Stateflow chart with this API command:

   ```
   ch.view;
   ```

   The preceding command calls the `view` method of the `Chart` object whose handle is `ch`. The specified chart appears. 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, `ch`. Continue by creating new objects for your chart using these steps:

1. Create a new state in the `Chart` object `ch` with this command:

   ```
   sA = Stateflow.State(ch);
   ```

   This command is a Stateflow API constructor for a new state in which `Stateflow.State` is the object type for a state, `ch` 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 chart.

2. Use the `ch.view` command to bring the chart to the foreground for viewing.

3. Assign a name and position to the new state by assigning values to the properties of the new `State` object 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 these commands:

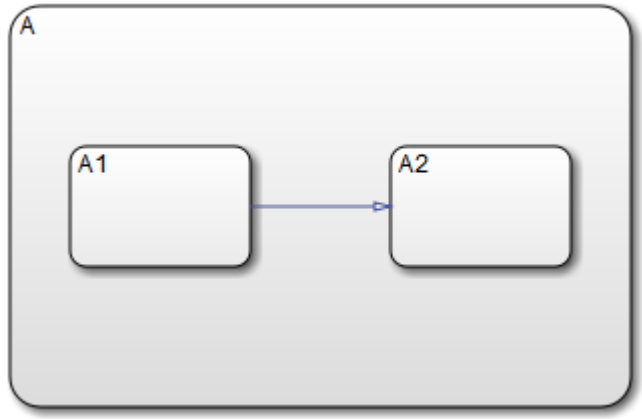
   ```
   sA1 = Stateflow.State(ch);
   sA1.Name = 'A1';
   sA1.Position = [80 120 90 60];
   sA2 = Stateflow.State(ch);
   sA2.Name = 'A2';
   sA2.Position = [240 120 90 60];
   ```

   These commands create and use the workspace variables `sA`, `sA1`, and `sA2` as handles to the new states, which now appear as follows.

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 these commands:

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

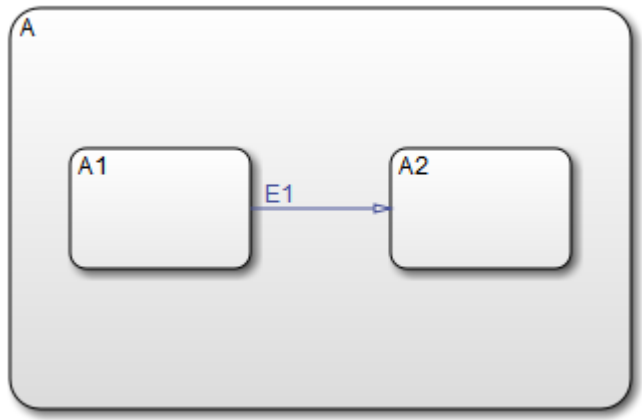
A transition now appears as shown.



6 Add the label `E1` to the transition from state `A1` to state `A2` with this command:

```
tA1A2.LabelPosition = [180 140 0 0];
tA1A2.LabelString = 'E1';
```

The chart now looks like this:



The state and transition labels in this chart are simple one-line labels. To enter more complex multiline labels, see “Enter Multiline Labels in States and Transitions” on page 1-38. Labels for transitions also have a `LabelPosition` property you can use to move the labels to better locations.

7 Use these commands to move the label for the transition from A1 to A2 to the right by 5 pixels:

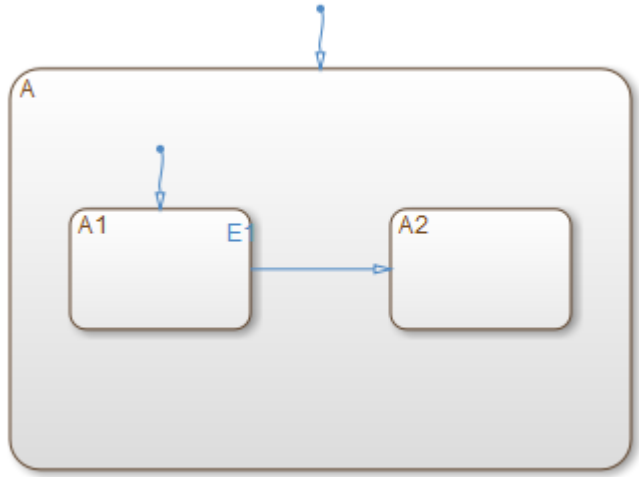
```
pos = tA1A2.LabelPosition;
pos(1) = pos(1)+5;
tA1A2.LabelPosition = pos;
```

8 Finish your new chart 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:

```
% Add a default transition to state A
dtA = Stateflow.Transition(ch);
dtA.Destination = sA;
dtA.DestinationOClock = 0;
xsource = sA.Position(1)+sA.Position(3)/2;
ysource = sA.Position(2)-30;
dtA.SourceEndPoint = [xsource ysource];
dtA.MidPoint = [xsource ysource+15];
```

```
% Add a default transition to state A1
dtA1 = Stateflow.Transition(ch);
dtA1.Destination = sA1;
dtA1.DestinationOClock = 0;
xsource = sA1.Position(1)+sA1.Position(3)/2;
ysource = sA1.Position(2)-30;
dtA1.SourceEndPoint = [xsource ysource];
dtA1.MidPoint = [xsource ysource+15];
```

Your complete chart looks like this:



9 Save the model with the new chart to the current folder as `myModel`:

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

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

See Also

`Stateflow.State` | `Stateflow.Transition` | `find` | `sfroot` | `sfsave` | `view`

More About

- “Overview of the Stateflow API” on page 1-2
- “Access Properties and Methods of Stateflow Objects” on page 1-14
- “Create Charts by Using a MATLAB Script” on page 1-43

Access Properties and Methods of Stateflow Objects

In this section...
“Naming Conventions for Properties and Methods” on page 1-14
“Using Dot Notation with Properties and Methods” on page 1-14
“Access Methods Using Function Notation” on page 1-15

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, this command returns the Type property for a State object represented by the handle `s`:

```
stype = s.Type;
```

This command calls the `dialog` method of the State object `s` to open a properties dialog box 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 a chart, this command returns the label for state A1's inner transition to a substate A11.

```
label = sA1.innerTransitions.LabelString;
```

The preceding command uses the `LabelString` property of a Transition object and the `innerTransitions` method for a State object. The command works as shown only when 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:

```
innerTransitions = sA1.innerTransitions;
label1 = innerTransitions(1).LabelString;
label2 = innerTransitions(2).LabelString;
```

Access Methods Using Function Notation

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 these commands:

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

If you have array arguments to methods you call, use function notation. This example returns a cell array of character vectors 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 this dot notation, an error results:

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

See Also

More About

- “Overview of the Stateflow API” on page 1-2
- “Create Charts by Using the Stateflow API” on page 1-7

Display Properties and Methods of Stateflow Objects

In this section...
“Display Properties” on page 1-16
“Display Names of Methods” on page 1-16
“Display Property Subproperties” on page 1-17
“Display Enumerated Values for Properties” on page 1-17

Display 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 this 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, this 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.

Display 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 this command to list the methods for any Transition object:

```
t.methods;
```

Note These internal methods may be displayed by the `methods` method for an object, but you cannot use them and they are 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 this command:

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

Display 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 Stateflow.StateFont]
```

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

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

This list shows that `Name`, `Size`, `Weight`, and `Angle` are subproperties of the property `StateFont`. In the API reference pages for this guide, these properties are listed by their full names: `StateFont.Name`, `StateFont.Size`, and so on.

Display Enumerated Values for Properties

Many API object properties can be set only to one of a group of values. You can identify these properties from the API reference pages. When you use the `get` method to access object properties (see “Display Properties” on page 1-16) the values for these properties appear.

You can use the `set` method to display a list of acceptable values for a property. 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 this command:

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

See Also

More About

- “Overview of the Stateflow API” on page 1-2
- “Create Charts by Using the Stateflow API” on page 1-7

Create and Destroy Stateflow Objects

About Creating and Destroying API Objects

You create (construct), parent (contain), and delete (destroy) objects in Stateflow charts 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 charts.

Stateflow objects are contained (parented) by other objects as defined in the Stateflow hierarchy of objects (see “Stateflow API Object Hierarchy” on page 1-2). You control containment of nongraphical objects in the Model Explorer.

Create 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 2-2 for a list of the valid constructor methods.

Use this 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 Machine Object” on page 1-7 and “Access the Chart Object” on page 1-8.

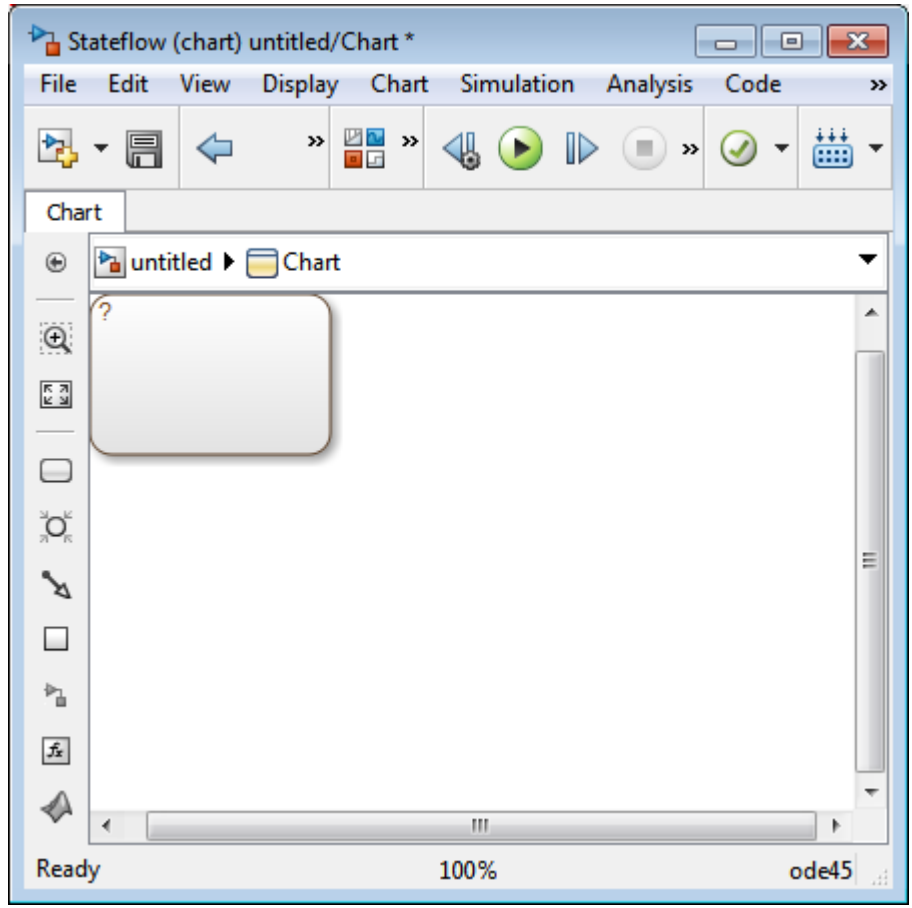
   See also “Access Existing Stateflow Objects” on page 1-23 for a more general means of accessing (getting an object handle to) an existing Stateflow object.

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

   For example, this 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 chart:



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 the chart.

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 1-9 for examples.

Use the preceding process to create all Stateflow objects in your chart. “Create New Objects in the Chart” on page 1-9 gives examples for creating states and transitions. You can also create objects of other types. For example, this 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.

Establish the Parent (Container) of an Object

As discussed in the previous section, “Create Stateflow Objects” on page 1-19, 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 you create graphical objects (states, boxes, notes, functions, transitions, junctions), they appear completely inside their containing parent object. In the chart, 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 the Stateflow parser cannot resolve. 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 must set the size and position of objects so that they are separate from other objects.

Nongraphical Object Parentage

When you create nongraphical objects (data, events, messages), they appear in the Model Explorer at the hierarchical level of their owning object. Containment for nongraphical objects is established through the Model Explorer only. See “Use the Model Explorer with Stateflow Objects”.

Destroy Stateflow Objects

Most Stateflow objects have a destructor method named `delete`. In this 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 chart. Upon deletion, graphical Stateflow objects are sent to the clipboard; nongraphical objects, such as data, events, and message are completely deleted. The workspace variable `s` still exists but is no longer a handle to the deleted state.

See Also

More About

- “Overview of the Stateflow API” on page 1-2
- “Create Charts by Using the Stateflow API” on page 1-7

Access Existing Stateflow Objects

About Stateflow Object Handles

Creating Stateflow objects through the Stateflow API gives you an immediate handle to the newly created objects (see “Create Stateflow Objects” on page 1-19). You can also connect to Stateflow objects that already exist for which you have no current API handle.

Find Objects and Properties

There are several object methods that you use to traverse the Stateflow hierarchy to locate existing objects. For example, you can use the `find` method.

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

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

This 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. This 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 charts, this command returns the Name property of the second chart found:

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

Tip To access the property of a Stateflow object in a linked library chart, do one of the following:

- Open the library model explicitly.

- View a linked subsystem or block in the main model.
- Compile or simulate the model.

Doing one of those steps loads a library model into the Simulink workspace. Just opening a main model that refers to a linked Stateflow chart does not guarantee that the Stateflow API can find a linked chart.

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, suppose that state `A`, which corresponds to State object `sA`, contains two states, `A1` and `A2`. Use a `find` command that finds all the states in `A`:

```
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.

Find Objects at Different Levels of Containment

Once you find a particular object in a Stateflow chart 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`.

Find 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 this 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);
```

This 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);
```

This command returns an array of all transitions inside state A at all levels of containment:

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

Find a Parent Object

The `up` method finds the parent container object of any given object. Suppose that you have a chart where 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.

Retrieve Recently Selected Objects

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

If...	Then `sfgco` returns...
There are no open charts	An empty matrix
There is no selection list	Handle of the chart most recently clicked
You select one object in a chart	Handle to the selected object
You select multiple objects in a chart	Vector of handles for the selected objects
You select objects in multiple charts	Handles of the most recently selected objects in the most recently selected chart

For example, suppose that you run the `sf_boiler` model and open the `Bang-Bang Controller` chart. If you select the `Off` state in the chart, `sfgco` returns:

```
ans =

               Path: 'sf_boiler/Bang-Bang Controller/Heater'
                 Id: 20
            Machine: [1x1 Stateflow.Machine]
               Name: 'Off'
        Description: ''
        LabelString: [1x27 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 214.1807 88.1000]
      Decomposition: 'EXCLUSIVE_OR'
               Type: 'OR'
          IsSubchart: 0
```

```
          IsGrouped: 1
              Debug: [1x1 Stateflow.StateDebug]
```

Get and Set 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 these 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 these commands:

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

See Also

More About

- “Overview of the Stateflow API” on page 1-2
- “Create Charts by Using the Stateflow API” on page 1-7

Move Stateflow Graphical Objects

In this section...
“How to Move Objects Programmatically” on page 1-28 “Move a Subcharted State” on page 1-28 “Rules for Moving Objects Programmatically” on page 1-29

How to Move Objects Programmatically

To move a graphical object programmatically, choose one of these techniques:

Technique	Example
Change the `Position` property of the object directly.	`object.Position = [40 40 100 60];`
Use the `set` method to change the `Position` property of the object.	`object.set('Position', [40 40 100 60]);`
	`set(object, 'Position', [40 40 100 60]);`

In each 1-by-4 array, the first two values are the (x,y) coordinates of the upper left corner of the object. The last two values are the width and height, respectively.

Note These programmatic techniques work only for objects that have the `Position` property.

Move a Subcharted State

You can adjust the location of a subcharted state as follows:

1. Open the `sf_elevator` model.
2. Get a handle to the root object.

   ```
   rt = slroot;
   ```

3. Get a handle to the subcharted state `Elevator_Manager` in the Elevator System chart.

```
em = rt.find('-isa','Stateflow.State','Name','Elevator_Manager');
```

4 Change the chart position of `Elevator_Manager`.

```
em.set('Position', [20 250 200 60]);
```

The following changes occur:

- The `Elevator_Manager` subchart moves to the location (20,250) from the upper left corner of the chart.
- The subchart now has a width of 200 and a height of 60.

Rules for Moving Objects Programmatically

- You cannot change the position of a subchart boundary in the subviewer programmatically.
- For objects in a subcharted state, box, or graphical function, you cannot use the `set` method to move these objects between different levels of the chart hierarchy. See “Copy and Paste Stateflow Objects” on page 1-30 for directions on copying and pasting objects from one container object to another.

See Also

More About

- “Overview of the Stateflow API” on page 1-2
- “Create Charts by Using the Stateflow API” on page 1-7

Copy and Paste Stateflow Objects

Access the Clipboard Object

The Clipboard object (only one exists) 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 these limitations for all objects:

- The objects you copy must be *all* graphical (states, boxes, functions, transitions, junctions) or *all* nongraphical (data, events, messages).

  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 try 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, you copy all the objects contained in the state, as well as all the relations among the objects in the grouped state. See “Copy by Grouping” on page 1-31.

Copy Graphical Objects

The `copy` method is subject to these limitations for all graphical objects:

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

  In other words, all graphical objects copied in a single copy command must reside in the same chart or subchart.

Copy by Grouping

Copying a grouped state in a Stateflow chart 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 method is the simplest way of copying objects. Use it 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.

This 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 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','Name',sA.Name);
sNew.IsGrouped=prevGrouping;
```

Copy 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 the MATLAB workspace 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 1-9, you can copy states A1, A2, and the transition connecting them to another state, B, with these API commands, where `sA` and `sB` are object 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);
cb.pasteTo(sB);
```

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

Note Copying objects individually is harder than copying grouped objects. See “Copy by Grouping” on page 1-31.

See Also

More About

- “Overview of the Stateflow API” on page 1-2
- “Create Charts by Using the Stateflow API” on page 1-7

View Stateflow Graphical Objects

Use the Stateflow API method `fitToView` to zoom in on a graphical object in the chart. (See “Get a Handle on Stateflow API Objects” on page 1-4 for information about obtaining object handles.)

Objects You Can Zoom

You can zoom the following chart objects:

- Charts
- Subcharts
- States
- Transitions
- Graphical functions
- Truth table functions
- MATLAB functions
- Simulink functions
- Connective junctions
- History junctions
- Boxes
- Notes

Zoom States in a Chart

Follow these steps to zoom in on different states:

1. At the MATLAB command prompt, type:

   ```
   old_sf_car;
   ```

   The chart `shift_logic` appears.
2. To define an object handle for the chart `shift_logic`, type:

   ```
   myChart = find(sfroot,'-isa','Stateflow.Chart','Name', ...
   'shift_logic');
   ```

3. To define an object handle for the state `upshifting`, type:

   ```
   myState = find(sfroot,'-isa','Stateflow.State','Name', ...
   'upshifting');
   ```

4. To zoom in on the state `upshifting`, type:

   ```
   myState.fitToView;
   ```

   The chart zooms in on the state and highlights it.

5. To define an object handle for the state `downshifting`, type:

   ```
   myState = find(sfroot,'-isa','Stateflow.State','Name', ...
   'downshifting');
   ```

6. To zoom in on the state `downshifting`, type:

   ```
   myState.fitToView;
   ```

   The chart zooms in on and highlights the state.

7. To zoom out to the chart-level view, type:

   ```
   myChart.fitToView;
   ```

   The chart `shift_logic` reappears.

8. You can also zoom in on a state using the `sfgco` function. Follow these steps:

   a. Click any state in the chart.

   b. At the MATLAB command prompt, type:

      ```
      myState = sfgco;
      ```

      This command assigns the selected state to the object handle `myState`.

   c. To zoom in on the selected state, type:

      ```
      myState.fitToView;
      ```

      The chart zooms in on the state and highlights it.

See Also

More About

- “Overview of the Stateflow API” on page 1-2
- “Create Charts by Using the Stateflow API” on page 1-7

Modify the Graphical Properties of Your Chart

In this section...
“About Editor Objects” on page 1-36
“Access the Editor Object” on page 1-36
“Change the Display in the Stateflow Editor” on page 1-36

About Editor Objects

The Editor object provides access to the purely graphical properties and methods of Chart objects. Each Chart object has its own Editor object.

Access 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 charts.

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 this command:

```
ed = ch.Editor;
```

Change the Display in the Stateflow Editor

Use the handle `ed` from the preceding example to access the Editor object properties and methods. For example, this 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 to an absolute zoom factor of 150%:

```
ed.ZoomFactor = 1.5;
```

The `ZoomFactor` is based on a Dots Per Inch (DPI) of 72. If your screen DPI is different, for example 96, and you want to change the zoom level of the chart, you must to scale

your `ZoomFactor` accordingly. For example, if you want 100%, you would set the `ZoomFactor` to `72/96` (`0.75`).

You can also use an Editor object to change the window position of the Stateflow Editor. For more information, see “Stateflow.Editor” on page 2-30.

See Also

More About

- “Overview of the Stateflow API” on page 1-2
- “Create Charts by Using the Stateflow API” on page 1-7

Enter Multiline Labels in States and Transitions

The following state uses a multiline label:

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

- Use the MATLAB function `sprintf`:

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

  In this example, the escape sequence `\n` inserts a new line into an expression.

- Use a concatenated text expression:

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

  In this example, the ASCII equivalent of a new line, the integer 10, inserts new lines into a concatenated text expression.

See Also

More About

- “Overview of the Stateflow API” on page 1-2
- “Create Charts by Using the Stateflow API” on page 1-7

Create Default Transition Objects

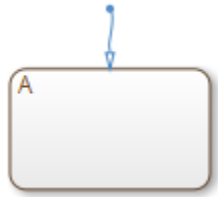
Default transitions differ from normal transitions in not having a source object. You can create a default transition with these steps:

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 variable `sA` is a handle to state A, these commands create a default transition and position the 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;
ysource = sA.Position(2)-30;
dt.SourceEndPoint = [xsource ysource];
dt.MidPoint = [xsource ysource+15];
```

The created default transition looks like this:



This method is also used for adding the default transitions toward the end of the example chart constructed in “Create New Objects in the Chart” on page 1-9.

See Also

More About

- “Overview of the Stateflow API” on page 1-2

- “Create Charts by Using the Stateflow API” on page 1-7

Create Supertransition Objects

The Stateflow API does not currently support the direct creation of supertransitions. Supertransitions are transitions between different levels in a chart. A supertransition can be between a state in a top-level chart and a state in one of its substates, or between states residing in different substates. For a better understanding of supertransitions, see “Move Between Levels of Hierarchy by Using Supertransitions”.

You can use a workaround for indirectly creating supertransitions. In this 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 step 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 these steps:

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 this command:

```
sA_pos = sA.Position;
```

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

   ```
   sA.IsSubchart = 0;
   ```

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

   ```
   sA.IsGrouped = 0;
   ```

   When you convert a subchart to 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 1-9 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 position can be restored after the transition connection is made.

See Also

More About

- “Overview of the Stateflow API” on page 1-2
- “Create Charts by Using the Stateflow API” on page 1-7

Create Charts by Using a MATLAB Script

In “Create Charts by Using the Stateflow API” on page 1-7, 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 lets you 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
ch = m.find('-isa', 'Stateflow.Chart');

% Create state A in chart
sA = Stateflow.State(ch);
sA.Name = 'A';
sA.Position = [50 50 310 200];

% Create state A1 inside of state A
sA1 = Stateflow.State(ch);
sA1.Name = 'A1';
sA1.Position = [80 120 90 60];

% Create state A2 inside of state A
sA2 = Stateflow.State(ch);
sA2.Name = 'A2';
sA2.Position = [240 120 90 60];

% Create a transition from A1 to A2
tA1A2 = Stateflow.Transition(ch);
tA1A2.Source = sA1;
tA1A2.Destination = sA2;
tA1A2.SourceOClock = 3;
tA1A2.DestinationOClock = 9;
```

```
% Label transition from state A1 to state A2
tA1A2.LabelPosition = [180 140 0 0];
tA1A2.LabelString = 'E1';

% Create the Event E1
E1 = Stateflow.Event(ch);
E1.Name = 'E1';

% Move label for transition A1-A2 to the right a bit
pos = tA1A2.LabelPosition;
pos(1) = pos(1)+5;
tA1A2.LabelPosition = pos;

% Add a default transition to state A
dtA = Stateflow.Transition(ch);
dtA.Destination = sA;
dtA.DestinationOClock = 0;
xsource = sA.Position(1)+sA.Position(3)/2;
ysource = sA.Position(2)-30;
dtA.SourceEndPoint = [xsource ysource];
dtA.MidPoint = [xsource ysource+15];

% Add a default transition to state A1
dtA1 = Stateflow.Transition(ch);
dtA1.Destination = sA1;
dtA1.DestinationOClock = 0;
xsource = sA1.Position(1)+sA1.Position(3)/2;
ysource = sA1.Position(2)-30;
dtA1.SourceEndPoint = [xsource ysource];
dtA1.MidPoint = [xsource ysource+15];
```

See Also

More About

- “Overview of the Stateflow API” on page 1-2
- “Create Charts by Using the Stateflow API” on page 1-7

API Object Reference

Properties and Methods Sorted by Stateflow Object

The following reference tables for Stateflow API properties and methods have these columns:

- **Name** — The name of the property or method. To access or set a property value or to call a method, use its name in dot notation along with a Stateflow object. Properties with multiple levels of hierarchy (such as the `LoggingInfo` and `Props` properties of data objects) must be set individually. For more information, see “Access Properties and Methods of Stateflow Objects” on page 1-14.
- **Type** — The data type for the property. Some property types are other Stateflow API objects. For example, the `Machine` property of an object is the `Stateflow.Machine` object that contains the object.
- **Access** — An access type for the property.
  - RW (read/write): You can access or set the value of these properties by using the Stateflow API.
  - RO (read-only): These properties are set by the Stateflow software.
- **Description** — A description of the property or method.

Constructor Methods

These methods create Stateflow API objects. Each method takes a parent object as an input and returns a handle to the new object. For more information, see “Create and Destroy Stateflow Objects” on page 1-19.

Name	Description
`Stateflow.Annotation`	Create an annotation in a parent chart, state, box, or function. See “Properties” on page 2-4 and “Methods” on page 2-7.
`Stateflow.AtomicBox`	Create an atomic box in a parent chart, state, box, or function. See “Properties” on page 2-8 and “Methods” on page 2-9.
`Stateflow.AtomicSubchart`	Create an atomic subchart in a parent chart, state, or box. See “Properties” on page 2-10 and “Methods” on page 2-12.
`Stateflow.Box`	Create a box in a parent chart, state, box, or function. See “Properties” on page 2-13 and “Methods” on page 2-14.

Name	Description
`Stateflow.Data`	Create a data in a parent machine, chart, state, box, or function. See “Properties” on page 2-24 and “Methods” on page 2-29.
`Stateflow.EMFunction`	Create a MATLAB function in a parent chart, state, box, or function. See “Properties” on page 2-31 and “Methods” on page 2-32.
`Stateflow.Event`	Create an event in a parent chart, state, or box. See “Properties” on page 2-33 and “Methods” on page 2-35.
`Stateflow.Function`	Create a graphical function in a parent chart, state, box, or function. See “Properties” on page 2-35 and “Methods” on page 2-37.
`Stateflow.Junction`	Create a junction in a parent chart, state, box, or function. See “Properties” on page 2-38 and “Methods” on page 2-39.
`Stateflow.Message`	Create a message in a parent chart, state, or box. See “Properties” on page 2-43 and “Methods” on page 2-46.
`Stateflow.SimulinkBasedState`	Create a Simulink based state in a parent chart, state, or box. See “Properties” on page 2-47 and “Methods” on page 2-50.
`Stateflow.SLFunction`	Create a Simulink function in a parent chart, state, box, or function. See “Properties” on page 2-51 and “Methods” on page 2-52.
`Stateflow.State`	Create a state in a parent chart, state, or box. See “Properties” on page 2-53 and “Methods” on page 2-57.
`Stateflow.Transition`	Create a transition in a parent chart, state, box, or function. See “Properties” on page 2-66 and “Methods” on page 2-68.
`Stateflow.TruthTable`	Create a truth table function in a parent chart, state, box, or function. See “Properties” on page 2-69 and “Methods” on page 2-71.

Root Object

The `Root` object is the parent of all Stateflow API objects. You automatically create a `Root` object when you load a Simulink model that contains a Stateflow chart or call the function `sfnew`. To create a handle to the `Root` object, call the `sfroot` function:

```
rt = sfroot;
```

For more information, see “Access the Machine Object” on page 1-7.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `Root` object type.
`find`	Find all objects inside the `Root` object that match the specified criteria.
`get`	Return the value of the specified property for the `Root` object.
`set`	Set the value of the specified property for the `Root` object.
`struct`	Return a MATLAB structure that contains all of the property values for the `Root` object.

Stateflow.Annotation

To create an annotation in a parent chart, state, box, or function, use the constructor method `Stateflow.Annotation`. For example, if `ch` is a handle to a `Chart` object, enter:

```
an = Stateflow.Annotation(ch);
```

For more information, see “Add Descriptive Comments in a Chart”.

Properties

Name	Type	Access	Description
`Alignment`	Enum	RW	Alignment of the text in this annotation. Options are `'CENTER'`, `'LEFT'` (default), or `'RIGHT'`.
`AutoBackgroundColor`	Boolean	RW	Use the automatic background color for this annotation. Default value is `true`.
`AutoForegroundColor`	Boolean	RW	Use the automatic foreground text color for this annotation. Default value is `true`.
`BackgroundColor`	Numeric vector	RW	Background color for this annotation. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[1 1 1]`.
`Chart`	`Chart`	RO	Chart that contains this annotation.

Name	Type	Access	Description
`ClickFcn`	Character vector	RW	MATLAB code to execute when you click this annotation. Default value is `''`. See "Add an Annotation Callback" (Simulink).
`DeleteFcn`	Character vector	RW	MATLAB code to execute before you delete this annotation. Default value is `''`. See "Add an Annotation Callback" (Simulink).
`Description`	Character vector	RW	Description of this annotation. Default value is `''`.
`Document`	Character vector	RW	Document link for this annotation. Default value is `''`.
`DropShadow`	Boolean	RW	Display a drop shadow. Default value is `false`.
`FixedHeight`	Boolean	RW	Fix the height of the annotation box. Options are: • `true` — Fixes the height of the annotation box and hides content that is longer than the box. • `false` — Resizes the annotation box vertically as you add content (default).
`FixedWidth`	Boolean	RW	Fix the width of the annotation box. Options are: • `true` — Fixes the width of the annotation box and wraps text that is longer than the box. • `false` — Resizes the annotation box horizontally as you add content (default).
`Font.Angle`	Enum	RW	Font angle for the text in this annotation. Options are `'ITALIC'` or `'NORMAL'` (default).
`Font.Name`	Character vector	RO	Font for the text in this annotation. The `StateFont.Name` property of the chart that contains the annotation sets the value of this property.
`Font.Size`	Double	RW	Font size for the text in this annotation. The `StateFont.Size` property of the chart that contains the annotation sets the initial value of this property.

Name	Type	Access	Description
`Font.Weight`	Enum	RW	Font weight for the text in this annotation. Options are `'BOLD'` or `'NORMAL'` (default).
`ForegroundColor`	Numeric vector	RW	Foreground color for this annotation. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0 0 0]`.
`Id`	Integer	RO	Unique identifier that distinguishes this annotation from other objects in the model.
`InternalMargins`	Numeric vector	RW	Space from the bounding box of the text to the borders of this annotation. Numeric vector `[left top right bottom]`. Default value is `[0 0 0 0]`.
`Interpretation`	Enum	RW	Specify how to interpret the contents of the `Text` property in this annotation. Options are `'OFF'` (default), `'RICH'`, or `'TEX'`.
`LoadFcn`	Character vector	RW	MATLAB code to execute when you load the model that contains this annotation. Default value is `''`. See “Add an Annotation Callback” (Simulink).
`Machine`	`Machine`	RO	Machine that contains this annotation.
`Path`	Character vector	RO	Location of the parent of this annotation in the model hierarchy.
`PlainText`	Character vector	RO	Text for this annotation without formatting.
`Position`	Numeric vector	RW	Position and size of this annotation in the chart. Numeric vector `[left top width height]`. Default value is `[0 0 8 16]`.
`Subviewer`	`Chart`, `State`, `Box`, or `Function`	RO	Chart or subchart where you can graphically view this annotation.
`Tag`	Any type	RW	Tag for this annotation. Holds data of any type. Default value is `[]`.

Name	Type	Access	Description
`Text`	Character vector	RW	Text for this annotation. Default value is `'?'`.
`UseDisplayTextAsClickCallback`	Boolean	RW	Use the contents of the `Text` property as the click function for this annotation. Default value is `false`. See “Add an Annotation Callback” (Simulink).

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `Annotation` object type.
`delete`	Delete this annotation.
`dialog`	Open the Annotation properties dialog box.
`disp`	Display all properties and values for this annotation.
`fitToView`	Zoom in on this annotation in the chart.
`get`	Return the value of the specified property for this annotation.
`help`	Display all properties and descriptions for this annotation.
`methods`	Display all methods for this annotation.
`set`	Set the value of the specified property for this annotation.
`setImage`	Insert an image into this annotation.
`struct`	Return a MATLAB structure that contains all of the property values for this annotation.
`up`	Return a handle to the object that contains this annotation.
`view`	Zoom in and select this annotation.

Stateflow.AtomicBox

To create an atomic box in a parent chart, state, box, or function, use the constructor method `Stateflow.AtomicBox`. For example, if `ch` is a handle to a `Chart` object, enter:

```
ab = Stateflow.AtomicBox(ch);
```

For more information, see “Reuse Functions by Using Atomic Boxes”.

Properties

Name	Type	Access	Description
`ArrowSize`	Double	RW	Size of incoming transition arrows for this atomic box. Default value is `8`.
`BadIntersection`	Boolean	RO	Indicates if this atomic box graphically intersects a box, state, or function.
`Chart`	`Chart`	RO	Chart that contains this atomic box.
`Description`	Character vector	RW	Description of this atomic box. Default value is `''`.
`Document`	Character vector	RW	Document link for this atomic box. Default value is `''`.
`FontSize`	Double	RW	Font size for the label of this atomic box. The `StateFont.Size` property of the chart that contains the atomic box sets the initial value of this property.
`Id`	Integer	RO	Unique identifier that distinguishes this atomic box from other objects in the model.
`IsExplicitlyCommented`	Boolean	RW	Explicitly comment out this atomic box. Default value is `false`. Equivalent to right-clicking the atomic box and selecting **Comment Out**.
`IsImplicitlyCommented`	Boolean	RO	Indicates if this atomic box is implicitly commented out. An atomic box is implicitly commented out when you comment out a superstate in its hierarchy.
`IsLink`	Boolean	RO	Indicates if this atomic box is a library link.
`LabelString`	Character vector	RW	Full label for this atomic box. Default value is `'?'`.
`Machine`	`Machine`	RO	Machine that contains this atomic box.
`Name`	Character vector	RW	Name of this atomic box. Default value is `''`.
`Path`	Character vector	RO	Location of the parent of this atomic box in the model hierarchy.

Name	Type	Access	Description
`Position`	Numeric vector	RW	Position and size of this atomic box in the chart. Numeric vector `[left top width height]`. Default value is `[0 0 90 60]`.
`Subviewer`	`Chart`, `State`, `Box`, or `Function`	RO	Chart or subchart where you can graphically view this atomic box.
`Tag`	Any type	RW	Tag for this atomic box. Holds data of any type. Default value is `[]`.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `AtomicBox` object type.
`delete`	Delete this atomic box.
`dialog`	Open the Box properties dialog box.
`disp`	Display all properties and values for this atomic box.
`find`	Find all objects inside this atomic box that match the specified criteria.
`fitToView`	Zoom in on this atomic box in the chart.
`get`	Return the value of the specified property for this atomic box.
`help`	Display all properties and descriptions for this atomic box.
`highlight`	Highlight this atomic box in the chart.
`isCommented`	Return a Boolean value that indicates if this atomic box is explicitly or implicitly commented out.
`methods`	Display all methods for this atomic box.
`set`	Set the value of the specified property for this atomic box.
`struct`	Return a MATLAB structure that contains all of the property values for this atomic box.
`up`	Return a handle to the object that contains this atomic box.
`view`	Display the contents of this atomic box.

Stateflow.AtomicSubchart

To create an atomic subchart in a parent chart, state, or box, use the constructor method `Stateflow.AtomicSubchart`. For example, if `ch` is a handle to a `Chart` object, enter:

```
as = Stateflow.AtomicSubchart(ch);
```

For more information, see “Create Reusable Subcomponents by Using Atomic Subcharts”.

Properties

Name	Type	Access	Description
`ArrowSize`	Double	RW	Size of incoming transition arrows for this atomic subchart. Default value is `8`.
`BadIntersection`	Boolean	RO	Indicates if this atomic subchart graphically intersects a box, state, or function.
`Chart`	`Chart`	RO	Chart that contains this atomic subchart.
`Debug.Breakpoints.OnDuring`	Boolean	RW	Set the `During State` breakpoint for this atomic subchart. Default value is `false`.
`Debug.Breakpoints.OnEntry`	Boolean	RW	Set the `On State Entry` breakpoint for this atomic subchart. Default value is `false`.
`Debug.Breakpoints.OnExit`	Boolean	RW	Set the `On State Exit` breakpoint for this atomic subchart. Default value is `false`.
`Description`	Character vector	RW	Description of this atomic subchart. Default value is `''`.
`Document`	Character vector	RW	Document link for this atomic subchart. Default value is `''`.
`ExecutionOrder`	Integer	RW	Order in which this atomic subchart wakes up in parallel (AND) decomposition. This property applies only when the `UserSpecifiedStateTransitionExecutionOrder` property of the chart that contains the atomic subchart is `true`.

Name	Type	Access	Description
`FontSize`	Double	RW	Font size for the label of this atomic subchart. The `StateFont.Size` property of the chart that contains the atomic subchart sets the initial value of this property.
`HasOutputData`	Boolean	RW	Create an active state data output port for this atomic subchart. Default value is `false`. See “Monitor State Activity Through Active State Data”.
`Id`	Integer	RO	Unique identifier that distinguishes this atomic subchart from other objects in the model.
`IsExplicitlyCommented`	Boolean	RW	Explicitly comment out this atomic subchart. Default value is `false`. Equivalent to right-clicking the atomic subchart and selecting **Comment Out**.
`IsImplicitlyCommented`	Boolean	RO	Indicates if this atomic subchart is implicitly commented out. An atomic subchart is implicitly commented out when you comment out a superstate in its hierarchy.
`IsLink`	Boolean	RO	Indicates if this atomic subchart is a library link.
`LabelString`	Character vector	RW	Full label for this atomic subchart. Default value is `'?'`.
`Machine`	`Machine`	RO	Machine that contains this atomic subchart.
`Name`	Character vector	RW	Name of this atomic subchart. Default value is `''`.
`OutputData`	`Data`	RO	Active state data object for this atomic subchart. This property applies only when the `HasOutputData` property for this atomic subchart is `true`. See “Monitor State Activity Through Active State Data”.
`OutputMonitoringMode`	Character vector	RO	Indicates the monitoring mode for the active state output data. For atomic subcharts, the only option is `'SelfActivity'`.
`Path`	Character vector	RO	Location of the parent of this atomic subchart in the model hierarchy.

Name	Type	Access	Description
`Position`	Numeric vector	RW	Position and size of this atomic subchart in the chart. Numeric vector `[left top width height]`. Default value is `[0 0 90 60]`.
`Subviewer`	`Chart`, `State`, or `Box`	RO	Chart or subchart where you can graphically view this atomic subchart.
`Tag`	Any type	RW	Tag for this atomic subchart. Holds data of any type. Default value is `[]`.
`TestPoint`	Boolean	RW	Set this atomic subchart as a Stateflow test point. Default value is `false`. See “Monitor Test Points in Stateflow Charts”.
`Type`	Enum	RO	Type of decomposition for this atomic subchart. Options are `'AND'` (parallel) or `'OR'` (exclusive). The atomic subchart inherits this property from the `Decomposition` property of its parent.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `AtomicSubchart` object type.
`delete`	Delete this atomic subchart
`dialog`	Open the State properties dialog box.
`disp`	Display all properties and values for this atomic subchart.
`find`	Find all objects inside this atomic subchart that match the specified criteria.
`fitToView`	Zoom in on this atomic subchart in the chart.
`get`	Return the value of the specified property for this atomic subchart.
`help`	Display all properties and descriptions for this atomic subchart.
`highlight`	Highlight this atomic subchart in the chart.
`isCommented`	Return a Boolean value that indicates if this atomic subchart is explicitly or implicitly commented out.

Name	Description
`methods`	Display all methods for this atomic subchart.
`set`	Set the value of the specified property for this atomic subchart.
`struct`	Return a MATLAB structure that contains all of the property values for this atomic subchart.
`up`	Return a handle to the object that contains this atomic subchart.
`view`	Display the contents of this atomic subchart.

Stateflow.Box

To create a box in a parent chart, state, box, or function, use the constructor method `Stateflow.Box`. For example, if `ch` is a handle to a `Chart` object, enter:

```
bx = Stateflow.Box(ch);
```

For more information, see “Group Chart Objects by Using Boxes”.

Properties

Name	Type	Access	Description
`ArrowSize`	Double	RW	Size of incoming transition arrows for this box. Default value is `8`.
`BadIntersection`	Boolean	RO	Indicates if this box graphically intersects a box, state, or function.
`Chart`	`Chart`	RO	Chart that contains this box.
`Description`	Character vector	RW	Description of this box. Default value is `''`.
`Document`	Character vector	RW	Document link for this box. Default value is `''`.
`FontSize`	Double	RW	Font size for the label of this box. The `StateFont.Size` property of the chart that contains the box sets the initial value of this property.
`Id`	Integer	RO	Unique identifier that distinguishes this box from other objects in the model.

Name	Type	Access	Description
`IsExplicitlyCommented`	Boolean	RW	Explicitly comment out this box. Default value is `false`. Equivalent to right-clicking the box and selecting **Comment Out**.
`IsGrouped`	Boolean	RW	Specify if this box is a group. Default value is `false`. See “Copy by Grouping” on page 1-31.
`IsImplicitlyCommented`	Boolean	RO	Indicates if this box is implicitly commented out. A box is implicitly commented out when you comment out a superstate in its hierarchy.
`IsSubchart`	Boolean	RW	Specify if this box is a subchart. Default value is `false`.
`LabelString`	Character vector	RW	Full label for this box. Default value is `'?'`.
`Machine`	`Machine`	RO	Machine that contains this box.
`Name`	Character vector	RW	Name of this box. Default value is `''`.
`Path`	Character vector	RO	Location of the parent of this box in the model hierarchy.
`Position`	Numeric vector	RW	Position and size of this box in the chart. Numeric vector `[left top width height]`. Default value is `[0 0 90 60]`.
`Subviewer`	`Chart`, `State`, `Box`, or `Function`	RO	Chart or subchart where you can graphically view this box.
`Tag`	Any type	RW	Tag for this box. Holds data of any type. Default value is `[]`.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `Box` object type.
`defaultTransitions`	Return the default transitions at the top level of containment of this box.
`delete`	Delete this box.

Name	Description
`dialog`	Open the Box properties dialog box.
`disp`	Display all properties and values for this box.
`find`	Find all objects inside this box that match the specified criteria.
`fitToView`	Zoom in on this box in the chart.
`get`	Return the value of the specified property for this box.
`help`	Display all properties and descriptions for this box.
`highlight`	Highlight this box in the chart.
`innerTransitions`	Return an array of the transitions that originate in this box and terminate on a contained object.
`isCommented`	Return a Boolean value that indicates if this box is explicitly or implicitly commented out.
`methods`	Display all methods for this box.
`outerTransitions`	Return an array of the transitions that exit the outer edge of this box and terminate on an object outside the containment of this box.
`set`	Set the value of the specified property for this box.
`sinkedTransitions`	Return all inner and outer transitions whose destination is this box.
`sourcedTransitions`	Return all inner and outer transitions whose source is this box.
`struct`	Return a MATLAB structure that contains all of the property values for this box.
`up`	Return a handle to the object that contains this box.
`view`	Zoom in and select this box. If the box is a subchart, display its contents.

Stateflow.Chart

To create a Simulink model that contains an empty Stateflow chart, call the function `sfnew`. For more information, see “Create Charts by Using the Stateflow API” on page 1-7.

Properties

Name	Type	Access	Description
`ActionLanguage`	Enum	RW	Action language used to program this chart. Options are `'C'` or `'MATLAB'` (default). See "Differences Between MATLAB and C as Action Language Syntax".
`ChartColor`	Numeric vector	RW	Background color for this chart. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[1 0.9608 0.8824]`.
`ChartUpdate`	Enum	RW	Activation method for this chart. Options are `'CONTINUOUS'`, `'DISCRETE'`, or `'INHERITED'` (default). See "Update Method".
`Debug.Breakpoints.OnEntry`	Boolean	RW	Set the `On Chart Entry` breakpoint for this chart. Default value is `false`.
`Decomposition`	Enum	RW	State decomposition at the top level of containment in this chart. Options are `'EXCLUSIVE_OR'` (default) and `'PARALLEL_AND'`.
`Description`	Character vector	RW	Description of this chart. Default value is `''`.
`Dirty`	Boolean	RW	Indicates if this chart has changed since being opened or saved. Default value is `false`.
`Document`	Character vector	RW	Document link for this chart. Default value is `''`.
`Editor`	`Editor`	RO	`Editor` object for this chart.

Name	Type	Access	Description
`EmlDefaultFimath`	Enum	RW	Default `fimath` properties for this chart. Options are: • `'Same as MATLAB Default'` — Use the same `fimath` properties as the current default `fimath` (default). • `'Other:UserSpecified'` — Use the `InputFimath` property to specify the default `fimath` object. This property applies only to charts that use MATLAB as the action language.
`EnableBitOps`	Boolean	RW	Use C-bit operations in state and transition actions in this chart. Default value is `false`. This property applies only to charts that use C as the action language. See “Supported Operations for Chart Data”.
`EnableNonTerminalStates`	Boolean	RW	Use super step semantics for this chart. Default value is `false`. See “Super Step Semantics”.
`EnableZeroCrossings`	Boolean	RW	Use zero-crossing detection on state transitions in this chart. Default value is `true`. This property applies only when the `ChartUpdate` property for this chart is set to `'CONTINUOUS'`. See “Disable Zero-Crossing Detection”.
`ErrorColor`	Numeric vector	RW	Color for errors in this chart. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[1 0 0]`.
`ExecuteAtInitialization`	Boolean	RW	Initialize the state configuration of this chart at time zero instead of at the first input event. Default value is `false`. See “Execution of a Chart at Initialization”.
`ExportChartFunctions`	Boolean	RW	Export graphical functions at the chart level to other blocks in the Simulink model. Default value is `false`. See “Export Stateflow Functions for Reuse”.

Name	Type	Access	Description
`HasOutputData`	Boolean	RW	Create an active state data output port for this chart. Default value is `false`. See “Monitor State Activity Through Active State Data”.
`Iced`	Boolean	RO	Equivalent to property `Locked`. Used internally to prevent changes in this chart during simulation.
`Id`	Integer	RO	Unique identifier that distinguishes this chart from other objects in the model.
`InitializeOutput`	Boolean	RW	Apply the initial value of the output data every time this chart wakes up. Default value is `false`. See “Initialize Outputs Every Time Chart Wakes Up”.
`InputFimath`	Character vector	RW	Specify the `embedded.fimath` object associated with inputs from Simulink blocks. You can: • Enter an expression that constructs a `fimath` object. • Enter the variable name for a `fimath` object in the MATLAB or model workspace. This property applies only when the `EmlDefaultFimath` property for this chart is `'Other:UserSpecified'`.
`JunctionColor`	Numeric vector	RW	Color for junctions in this chart. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0.6824 0.3294 0]`.
`Locked`	Boolean	RW	Prevent changes in this chart. Default value is `false`.
`Machine`	`Machine`	RO	Machine that contains this chart.
`Name`	Character vector	RW	Name of this chart. Default value is `'Chart'`.

Name	Type	Access	Description
`NonTerminalMaxCounts`	Integer	RW	Maximum number of transitions this chart can take in one super step. Default value is `1000`. This property applies only when the `EnableNonTerminalStates` property for this chart is `true`. See “Super Step Semantics”.
`NonTerminalUnstableBehavior`	Enum	RW	Behavior during simulation if this chart exceeds the maximum number of transitions specified in the `NonTerminalMaxCounts` property before reaching a stable state. Options are: • `'PROCEED'` — The chart goes to sleep with the last active state configuration (default). • `'THROW ERROR'` — The chart generates an error. This property applies only when the `EnableNonTerminalStates` property for this chart is `true`. See “Super Step Semantics”.
`OutputData`	`Data`	RO	Active state data object for this chart. This property applies only when the `HasOutputData` property for this chart is `true`. See “Monitor State Activity Through Active State Data”.
`OutputMonitoringMode`	Enum	RW	Indicates the monitoring mode for the active state output data. Options are `'ChildActivity'` (default) or `'LeafStateActivity'`. This property applies only when the `HasOutputData` property for this chart is `true`. See “Monitor State Activity Through Active State Data”.
`Path`	Character vector	RO	Location of this chart in the model hierarchy.
`RegisterExportedFunctionsWithSimulink`	Boolean	RW	Enable the Simulink model to call graphical, truth table, and MATLAB functions in this chart. Default value is `false`. See “Export Stateflow Functions for Reuse”.

Name	Type	Access	Description
`SampleTime`	Character vector	RW	Sample time for activating this chart. Default value is `'-1'`. This property applies only when the `ChartUpdate` property for this chart is `'DISCRETE'`.
`SaturateOnIntegerOverflow`	Boolean	RW	Specify the behavior of integer overflows in this chart. Options are: • `true` — The chart saturates integer overflows (default). • `false` — The chart wraps integer overflows. For more information, see “Handle Integer Overflow for Chart Data”.
`StateColor`	Numeric vector	RW	Color for state boxes in this chart. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0 0 0]`.
`StateFont.Angle`	Enum	RW	Font angle for the labels in boxes, functions, and states in this chart. Options are `'ITALIC'` or `'NORMAL'` (default).
`StateFont.Name`	Character vector	RW	Font for the labels in annotations, boxes, functions, and states in this chart. Default value is `'Helvetica'`.
`StateFont.Size`	Integer	RW	Initial font size for the labels in annotations, boxes, functions, and states in this chart. Default value is 12.
`StateFont.Weight`	Enum	RW	Font weight for the labels in boxes, functions, and states in this chart. Options are `'BOLD'` or `'NORMAL'` (default).
`StateLabelColor`	Numeric vector	RW	Color for state labels in this chart. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0 0 0]`.

Name	Type	Access	Description
`StateMachineType`	Enum	RW	Type of state machine semantics. Options are `'Classic'` (default), `'Mealy'`, or `'Moore'`. See “Overview of Mealy and Moore Machines”.
`StatesWhenEnabling`	Enum	RW	Specify the behavior of states when a function-call input event reenables this chart. Options are: • `'held'` — The chart maintains the most recent values of the states (default). • `'reset'` — The chart reverts to the initial conditions of the states. This property applies only when the chart contains function-call input events. See “Control States in Charts Enabled by Function-Call Input Events”.
`StrongDataTypingWithSimulink`	Boolean	RW	Use strong data typing when this chart interfaces with Simulink input and output signals. Default value is `true`. This property applies only to charts that use C as the action language. See “Use Strong Data Typing with Simulink I/O”.
`SupportVariableSizing`	Boolean	RW	Support input and output data that vary in dimension when simulating this chart. Default value is `true`. See “Declare Variable-Size Data in Stateflow Charts”.
`Tag`	Any type	RW	Tag for this chart. Holds data of any type. Default value is `[]`.
`TransitionColor`	Numeric vector	RW	Color for transitions in this chart. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0.2902 0.3294 0.6039]`.
`TransitionFont.Angle`	Enum	RW	Font angle for the transition labels in this chart. Options are `'ITALIC'` or `'NORMAL'` (default).

Name	Type	Access	Description
`TransitionFont.Name`	Character vector	RW	Font for the transition labels in this chart. Default value is `'Helvetica'`.
`TransitionFont.Size`	Integer	RW	Initial font size for the transition labels in this chart. Default value is `12`.
`TransitionFont.Weight`	Enum	RW	Font weight for the transition labels in this chart. Options are `'BOLD'` or `'NORMAL'` (default).
`TransitionLabelColor`	Numeric vector	RW	Color for the transition labels in this chart. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0.2902 0.3294 0.6039]`.
`TreatAsFi`	Enum	RW	Treat inherited fixed-point and integer signals as Fixed-Point Designer™ `fi` objects. Options are: • `'Fixed-point'` — The chart treats all fixed-point inputs as `fi` objects (default). • `'Fixed-point & Integer'` — The chart treats all fixed-point and integer inputs as `fi` objects. This property applies only to charts that use MATLAB as the action language.
`UserSpecifiedStateTransitionExecutionOrder`	Boolean	RW	Use explicit ordering of parallel states and transitions. Default value is `true`. This property applies only to charts that use C as the action language. See “User Specified State/Transition Execution Order”.
`Visible`	Boolean	RW	Indicates if the editor is displaying this chart.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `Chart` object type.
`defaultTransitions`	Return the default transitions at the top level of containment of this chart.
`dialog`	Open the Chart properties dialog box.

Name	Description
`disp`	Display all properties and values for this chart.
`find`	Find all objects inside this chart that match the specified criteria.
`fitToView`	Zoom in on this chart.
`get`	Return the value of the specified property for this chart.
`help`	Display all properties and descriptions for this chart.
`methods`	Display all methods for this chart.
`parse`	Parse this chart.
`set`	Set the value of the specified property for this chart.
`struct`	Return a MATLAB structure that contains all of the property values for this chart.
`view`	Display the contents of this chart.

Stateflow.Clipboard

To create a handle to the `Clipboard` object, call the `sfclipboard` function:

```
cb = sfclipboard;
```

For more information, see “Copy and Paste Stateflow Objects” on page 1-30.

Methods

Name	Description
`copy`	Copy the specified objects to this clipboard.
`methods`	Display all methods for this clipboard.
`pasteTo`	Paste the contents of this clipboard to the specified container object.

Stateflow.Data

To create a data object in a parent machine, chart, state, box, or function, use the constructor method `Stateflow.Data`. For example, if `ch` is a handle to a `Chart` object, enter:

```
x = Stateflow.Data(ch);
```

For more information, see “Set Data Properties”.

Properties

Name	Type	Access	Description
`CompiledSize`	Character vector	RO	Size of this data object as determined by the compiler.
`CompiledType`	Character vector	RO	Type of this data object as determined by the compiler.
`DataType`	Character vector	RW	Type of this data object. • If the `Props.Type.Method` property of this data object is `'Built-in'`, you can specify this property with one of these options: `'double'` (default), `'single'`, `'int8'`, `'int16'`, `'int32'`, `'int64'` (C charts only), `'uint8'`, `'uint16'`, `'uint32'`, `'uint64'` (C charts only), `'boolean'`, `'ml'`, or `'string'` (C charts only). • Otherwise, the `Props.Type` properties of this data object determine the value of this property.
`Description`	Character vector	RW	Description of this data object. Default value is `''`.
`Document`	Character vector	RW	Document link for this data object. Default value is `''`.
`Id`	Integer	RO	Unique identifier that distinguishes this data object from other objects in the model.
`InitializeMethod`	Enum	RW	Method for initializing the value of this data object. Options depend on the scope of the data: • For local and output data, use `'Expression'` (default) or `'Parameter'`. • For constant data, use `'Expression'` (read-only). • For data store memory, input, and parameter data, use `'Not Needed'` (read-only).

Name	Type	Access	Description
`LoggingInfo.DataLogging`	Boolean	RW	Enable signal logging for this data object. Default value is `false`.
`LoggingInfo.DecimateData`	Boolean	RW	Limit the amount of data logged by skipping samples. Uses the interval specified by the `LoggingInfo.Decimation` property of this data object. Default value is `false`.
`LoggingInfo.Decimation`	Integer	RW	Decimation interval. Default value is `2`. This value means the chart logs every other sample of this data object.
`LoggingInfo.LimitDataPoints`	Boolean	RW	Limit the number of data points to log. Uses the value specified by the `LoggingInfo.MaxPoints` property of this data object. Default value is `false`.
`LoggingInfo.MaxPoints`	Integer	RW	Maximum number of data points to log. Default value is `5000`. This value means the chart logs the last 5000 data points generated by the simulation for this data object.
`LoggingInfo.NameMode`	Enum	RW	Source of the signal name used to log this data object. Options are: • `'Custom'` — Use the custom signal name specified by the `LoggingInfo.LoggingName` property. • `'SignalName'` — Use the name of the data object (default).
`LoggingInfo.LoggingName`	Character vector	RW	Custom signal name used for logging this data object.
`Machine`	`Machine`	RO	Machine that contains this data object.
`Name`	Character vector	RW	Name of this data object.
`Path`	Character vector	RO	Location of the parent of this data object in the model hierarchy.
`Port`	Integer	RW	Port index for this data object. This property applies only to input and output data.

Name	Type	Access	Description
`Props.Array.FirstIndex`	Character vector	RW	Index for the first element of this data object. Default value is `0`. This property applies only to array data in charts that use C as the action language.
`Props.Array.IsDynamic`	Boolean	RW	Allow the size of this data object to change at run time. Default value is `false`. Equivalent to the **Variable Size** check box in the Data properties dialog box. See “Declare Variable-Size Data in Stateflow Charts”.
`Props.Array.Size`	Character vector	RW	Size of this data object. Default value is `'0'`. See “Specify Size of Stateflow Data”.
`Props.Complexity`	Enum	RW	Enable this data object to take complex values. Options are `'On'` or `'Off'` (default). See “Complex Data in Stateflow Charts”.
`Props.Frame`	Enum	RW	Enable this data object to accept frame-based signals. Options are: • `'Frame based'` — The data object supports frame-based signals. • `'Sample based'` — The data object supports sample-based signals (default).
`Props.InitialValue`	Character vector	RW	Initial value of this data object. Default value is `''`.
`Props.Range.Maximum`	Character vector	RW	Maximum value for this data object. Default value is `''`.
`Props.Range.Minimum`	Character vector	RW	Minimum value for this data object. Default value is `''`.
`Props.ResolveToSignalObject`	Boolean	RW	Specify if this data object resolves to a `Simulink.Signal` object that you define in the model or base workspace. Default value is `false`. See “Resolve Data Properties from Simulink Signal Objects”.

Name	Type	Access	Description
`Props.Type.BusObject`	Character vector	RW	Name of the `Simulink.Bus` object that defines this data object. This property applies only when the `Props.Type.Method` property of this data object is `'Bus Object'`. See “Access Bus Signals Through Stateflow Structures”.
`Props.Type.EnumType`	Character vector	RW	Name of the enumerated type that defines this data object. This property applies only when the `Props.Type.Method` property of this data object is `'Enumerated'`. See “Reference Values by Name by Using Enumerated Data”.
`Props.Type.Expression`	Character vector	RW	Expression that evaluates to a data type for this data object. This property applies only when the `Props.Type.Method` property of this data object is `'Expression'`. See “Specify Data Properties by Using MATLAB Expressions”.
`Props.Type.Fixpt.Bias`	Character vector	RW	Bias for this data object. This property applies only to fixed-point data with the `Props.Type.Fixpt.ScalingMode` property set to `'Slope and bias'`. See “Fixed-Point Data in Stateflow Charts”.
`Props.Type.Fixpt.FractionLength`	Character vector	RW	Location of the binary point for this data object. This property applies only to fixed-point data when the `Props.Type.Fixpt.ScalingMode` property is `'Binary point'`. See “Fixed-Point Data in Stateflow Charts”.
`Props.Type.Fixpt.Lock`	Boolean	RW	Prevents replacement of the fixed-point type of this data object with an autoscaled type chosen by the Fixed-Point Tool. Default value is `false`. See “Autoscaling Using the Fixed-Point Tool” (Fixed-Point Designer).
`Props.Type.Fixpt.ScalingMode`	Enum	RW	Method for scaling this data object. Options are `'Binary point'`, `'Slope and bias'`, or `'None'` (default). This property applies to fixed-point data. See “Fixed-Point Data in Stateflow Charts”.

Name	Type	Access	Description
`Props.Type.Fixpt.Slope`	Character vector	RW	Slope for this data object. This property applies only to fixed-point data when the `Props.Type.Fixpt.ScalingMode` property set to `'Slope and bias'`. See “Fixed-Point Data in Stateflow Charts”.
`Props.Type.Method`	Enum	RW	Method for setting the type of this data object. Options depend on the scope of the data: • For local, input, output, or parameter data, use `'Built-in'`, `'Fixed point'`, `'Enumerated'`, `'Expression'`, `'Inherited'` (default), or `'Bus Object'`. • For constant data, use `'Built-in'` (default), `'Fixed point'`, or `'Expression'`. • For data store memory data, use `'Inherited'` (read-only). Equivalent to the **Mode** field of the Data Type Assistant in the Data properties dialog box. See “Specify Type of Stateflow Data”.
`Props.Type.Signed`	Boolean	RW	Specify if this data object is signed. Default value is `true`. This property applies only to fixed-point data. See “Fixed-Point Data in Stateflow Charts”.
`Props.Type.WordLength`	Character vector	RW	Bit size of the word that holds the quantized integer of this data object. This property applies only to fixed-point data. See “Fixed-Point Data in Stateflow Charts”.
`Props.Unit.Name`	Character vector	RW	Units of measurement for this data object. Default value is `''`. See “Specify Units for Stateflow Data”.
`SaveToWorkspace`	Boolean	RW	Assign the value of this data object to a variable of the same name in the MATLAB base workspace at the end of the simulation. Default value is `false`. See “Save Final Value to Base Workspace”.

Name	Type	Access	Description
`Scope`	Enum	RW	Scope of this data object. Options are `'Local'` (default), `'Constant'`, `'Parameter'`, `'Input'`, `'Output'`, `'Data Store Memory'`, `'Temporary'`, `'Imported'`, or `'Exported'`. For more information, see “Scope”.
`Tag`	Any type	RW	Tag for this data object. Holds data of any type. Default value is `[]`.
`TestPoint`	Boolean	RW	Set this data object as a Stateflow test point. Default value is `false`. See “Monitor Test Points in Stateflow Charts”.
`Tunable`	Boolean	RW	Indicates that the value of this data object can be modified during simulation. Default is `true`. This property applies only to parameter data.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `Data` object type.
`delete`	Delete this data object.
`dialog`	Open the Data properties dialog box.
`disp`	Display all properties and values for this data object.
`get`	Return the value of the specified property for this data object.
`help`	Display all properties and descriptions for this data object.
`methods`	Display all methods for this data object.
`set`	Set the value of the specified property for this data object.
`struct`	Return a MATLAB structure that contains all of the property values for this data object.
`up`	Return a handle to the object that contains this data object.
`view`	Display this data object in the Model Explorer.

Stateflow.Editor

Each chart has its own `Editor` object. To create a handle to an `Editor` object, access the `Editor` property for the chart. For example, if `ch` is a handle to a `Chart` object, enter:

```
ed = ch.Editor;
```

For more information, see “Modify the Graphical Properties of Your Chart” on page 1-36.

Properties

Name	Type	Access	Description
`WindowPosition`	Numeric vector	RW	Position and size of the Stateflow editor window. Numeric vector `[left top width height]`.
`ZoomFactor`	Double	RW	Magnification level of this chart in the editor. A value of `1` corresponds to a magnification of 100%.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `Editor` object type.
`disp`	Display all properties and values for this `Editor` object.
`get`	Return the value of the specified property for this `Editor` object.
`help`	Display all properties and descriptions for this `Editor` object.
`methods`	Display all methods for this `Editor` object.
`set`	Set the value of the specified property for this `Editor` object.
`struct`	Return a MATLAB structure that contains all of the property values for this `Editor` object.
`zoomIn`	Zoom in on the Stateflow chart that contains this `Editor` object.
`zoomOut`	Zoom out on the Stateflow chart that contains this `Editor` object.

Stateflow.EMFunction

To create a MATLAB function in a parent chart, state, box, or function, use the constructor method `Stateflow.EMFunction`. For example, if `ch` is a handle to a `Chart` object, enter:

```
f = Stateflow.EMFunction(ch);
```

For more information, see “Reuse MATLAB Code by Defining MATLAB Functions”.

Properties

Name	Type	Access	Description
`BadIntersection`	Boolean	RO	Indicates if this MATLAB function graphically intersects a box, state, or function.
`Chart`	`Chart`	RO	Chart that contains this MATLAB function.
`Description`	Character vector	RW	Description of this MATLAB function. Default value is `''`.
`Document`	Character vector	RW	Document link for this MATLAB function. Default value is `''`.
`FontSize`	Double	RW	Font size for the label of this MATLAB function. The `StateFont.Size` property of the chart that contains the function sets the initial value of this property.
`Id`	Integer	RO	Unique identifier that distinguishes this MATLAB function from other objects in the model.
`InlineOption`	Character vector	RW	Specify how this MATLAB function appears in generated code. Options are: • `'Inline'` — Calls to the MATLAB function are replaced by code. • `'Function'` — MATLAB function is implemented as a separate C function. • `'Auto'` — An internal calculation determines the appearance of function calls in generated code (default). For more information, see “Inline State Functions in Generated Code” (Simulink Coder).
`IsExplicitlyCommented`	Boolean	RW	Explicitly comment out this MATLAB function. Default value is `false`. Equivalent to right-clicking the function and selecting **Comment Out**.

Name	Type	Access	Description
`IsImplicitlyCommented`	Boolean	RO	Indicates if this MATLAB function is implicitly commented out. A function is implicitly commented out when you comment out a superstate in its hierarchy.
`LabelString`	Character vector	RW	Full label of this MATLAB function. Label syntax is '*`return = Name(arguments)`*'. Default value is `'()'`.
`Machine`	`Machine`	RO	Machine that contains this MATLAB function.
`Name`	Character vector	RW	Name of this MATLAB function. Default value is `''`.
`Path`	Character vector	RO	Location of the parent of this MATLAB function in the model hierarchy.
`Position`	Numeric vector	RW	Position and size of this MATLAB function in the chart. Numeric vector `[left top width height]`. Default value is `[0 0 90 60]`.

`Script`	Character vector	RW	Code for this MATLAB function. To specify the value of this property, create a character vector by calling the `sprintf` function. For example, if `f` is a handle to this function, enter:
			`str = sprintf('function y=f(x) \n y=x+1;');`
			`f.Script = str;`

`Subviewer`	`Chart`, `State`, `Box`, or `Function`	RO	Chart or subchart where you can graphically view this MATLAB function.
`Tag`	Any type	RW	Tag for this MATLAB function. Holds data of any type. Default value is `[]`.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `EMFunction` object type.
`delete`	Delete this function.

Name	Description
`dialog`	Open the MATLAB Function properties dialog box.
`disp`	Display all properties and values for this function.
`find`	Find all objects inside this function that match the specified criteria.
`fitToView`	Zoom in on this function in the chart.
`get`	Return the value of the specified property for this function.
`help`	Display all properties and descriptions for this function.
`highlight`	Highlight this function in the chart.
`isCommented`	Return a Boolean value that indicates if this function is explicitly or implicitly commented out.
`methods`	Display all methods for this function.
`set`	Set the value of the specified property for this function.
`struct`	Return a MATLAB structure that contains all of the property values for this function.
`up`	Return a handle to the object that contains this function.
`view`	Open this function in the MATLAB Editor.

Stateflow.Event

To create an event in a parent chart, state, or box, use the constructor method `Stateflow.Event`. For example, if `ch` is a handle to a `Chart` object, enter:

```
e = Stateflow.Event(ch);
```

For more information, see “Synchronize Model Components by Broadcasting Events”.

Properties

Name	Type	Access	Description
`Debug.Breakpoints.EndBroadcast`	Boolean	RW	Set the `End of Broadcast` breakpoint of this event. Default value is `false`. This property applies only to local events.

Name	Type	Access	Description
`Debug.Breakpoints.StartBroadcast`	Boolean	RW	Set the `Start of Broadcast` breakpoint of this event. Default value is `false`. This property applies only to local or input events.
`Description`	Character vector	RW	Description of this event. Default value is `''`.
`Document`	Character vector	RW	Document link for this event. Default value is `''`.
`Id`	Integer	RO	Unique identifier that distinguishes this event from other objects in the model.
`Machine`	`Machine`	RO	Machine that contains this event.
`Name`	Character vector	RW	Name of this event.
`Path`	Character vector	RO	Location of the parent of this event in the model hierarchy.
`Port`	Integer	RW	Port index for this event. This property applies only to input and output events.
`Scope`	Enum	RW	Scope of this event. Options are `'Input'`, `'Local'` (default), or `'Output'`. For more information, see “Scope”.
`Tag`	Any type	RW	Tag for this event. Holds data of any type. Default value is `[]`.
`Trigger`	Enum	RW	Type of trigger associated with this event. Options depend on the scope of the event: • For input events, use `'Rising'`, `'Falling'`, `'Either'`, or `'Function call'` (default). • For output events, use `'Either'` or `'Function call'` (default). This property does not apply to local events.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `Event` object type.
`delete`	Delete this event.
`dialog`	Open the Event properties dialog box.
`disp`	Display all properties and values for this event.
`get`	Return the value of the specified property for this event.
`help`	Display all properties and descriptions for this event.
`methods`	Display all methods for this event.
`set`	Set the value of the specified property for this event.
`struct`	Return a MATLAB structure that contains all of the property values for this event.
`up`	Return a handle to the object that contains this event.
`view`	Display this data object in the Model Explorer.

Stateflow.Function

To create a graphical function in a parent chart, state, box, or function, use the constructor method `Stateflow.Function`. For example, if `ch` is a handle to a `Chart` object, enter:

```
f = Stateflow.Function(ch);
```

For more information, see “Reuse Logic Patterns by Defining Graphical Functions”.

Properties

Name	Type	Access	Description
`BadIntersection`	Boolean	RO	Indicates if this graphical function graphically intersects a box, state, or function.
`Chart`	`Chart`	RO	Chart that contains this graphical function.
`Debug.Breakpoints.OnDuring`	Boolean	RW	Set the `During Function Call` breakpoint for this graphical function. Default value is `false`.

Name	Type	Access	Description
`Description`	Character vector	RW	Description of this graphical function. Default value is `''`.
`Document`	Character vector	RW	Document link for this graphical function. Default value is `''`.
`FontSize`	Double	RW	Font size for the label of this graphical function. The `StateFont.Size` property of the chart that contains the function sets the initial value of this property.
`Id`	Integer	RO	Unique identifier that distinguishes this graphical function from other objects in the model.
`InlineOption`	Character vector	RW	Specify how this graphical function appears in generated code. Options are: • `'Inline'` — Calls to the graphical function are replaced by code. • `'Function'` — The graphical function is implemented as a separate C function. • `'Auto'` — An internal calculation determines the appearance of function calls in generated code (default). For more information, see “Inline State Functions in Generated Code” (Simulink Coder).
`IsExplicitlyCommented`	Boolean	RW	Explicitly comment out this graphical function. Default value is `false`. Equivalent to right-clicking the function and selecting **Comment Out**.
`IsGrouped`	Boolean	RW	Specify if this graphical function is a group. Default value is `false`. See “Copy by Grouping” on page 1-31.
`IsImplicitlyCommented`	Boolean	RO	Indicates if this graphical function is implicitly commented out. A function is implicitly commented out when you comment out a superstate in its hierarchy.

Name	Type	Access	Description
`IsSubchart`	Boolean	RW	Specify if this graphical function is a subchart. Default value is `false`.
`LabelString`	Character vector	RW	Full label of this graphical function. Label syntax is '*`return = Name(arguments)`*'. Default value is `'()'`.
`Machine`	`Machine`	RO	Machine that contains this graphical function.
`Name`	Character vector	RW	Name of this graphical function. Default value is `''`.
`Path`	Character vector	RO	Location of the parent of this graphical function in the model hierarchy.
`Position`	Numeric vector	RW	Position and size of this graphical function in the chart. Numeric vector `[left top width height]`. Default value is `[0 0 90 60]`.
`Subviewer`	`Chart`, `State`, `Box`, or `Function`	RO	Chart or subchart where you can graphically view this graphical function.
`Tag`	Any type	RW	Tag for this graphical function. Holds data of any type. Default value is `[]`.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `Function` object type.
`defaultTransitions`	Return the default transitions at the top level of containment of this function.
`delete`	Delete this function.
`dialog`	Open the Function properties dialog box.
`disp`	Display all properties and values for this function.
`find`	Find all objects inside this function that match the specified criteria.
`fitToView`	Zoom in on this function in the chart.

Name	Description
get	Return the value of the specified property for this function.
help	Display all properties and descriptions for this function.
highlight	Highlight this function in the chart.
isCommented	Return a Boolean value that indicates if this function is explicitly or implicitly commented out.
methods	Display all methods for this function.
set	Set the value of the specified property for this function.
struct	Return a MATLAB structure that contains all of the property values for this function.
up	Return a handle to the object that contains this function.
view	Zoom in and select this function. If the function is a subchart, display its contents.

Stateflow.Junction

To create a junction in a parent chart, state, box, or function, use the constructor method `Stateflow.Junction`. For example, if `ch` is a handle to a `Chart` object, enter:

```
j = Stateflow.Junction(ch);
```

For more information, see “Connective Junctions” and “Record State Activity by Using History Junctions”.

Properties

Name	Type	Access	Description
ArrowSize	Double	RW	Size of incoming transition arrows for this junction. Default value is 8.
Chart	Chart	RO	Chart that contains this junction.
Description	Character vector	RW	Description of this junction. Default value is ''.
Document	Character vector	RW	Document link for this junction. Default value is ''.

Name	Type	Access	Description
`Id`	Integer	RO	Unique identifier that distinguishes this junction from other objects in the model.
`IsExplicitlyCommented`	Boolean	RW	Explicitly comment out this junction. Default value is `false`. Equivalent to right-clicking the junction and selecting **Comment Out**.
`IsImplicitlyCommented`	Boolean	RO	Indicates if this junction is implicitly commented out. A junction is implicitly commented out when you comment out a superstate in its hierarchy.
`Machine`	`Machine`	RO	Machine that contains this junction.
`Path`	Character vector	RO	Location of the parent of this junction in the model hierarchy.
`Position.Center`	Numeric vector	RW	Position of the center of this junction. Numeric vector `[x y]` of coordinates relative to the upper left corner of the parent chart or state. Default value is `[7 7]`.
`Position.Radius`	Integer	RW	Radius of this junction. Default value is `7`.
`Subviewer`	`Chart`, `State`, `Box`, or `Function`	RO	Chart or subchart where you can graphically view this junction.
`Tag`	Any type	RW	Tag for this junction. Holds data of any type. Default value is `[]`.
`Type`	Enum	RW	Type for this junction. Options are `'CONNECTIVE'` (default) or `'HISTORY'`.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `Junction` object type.
`delete`	Delete this junction.
`dialog`	Open the Connective Junction properties dialog box.
`disp`	Display all properties and values for this junction.

Name	Description
fitToView	Zoom in on this junction in the chart.
get	Return the value of the specified property for this junction.
help	Display all properties and descriptions for this junction.
highlight	Highlight this junction in the chart.
isCommented	Return a Boolean value that indicates if this junction is explicitly or implicitly commented out.
methods	Display all methods for this junction.
set	Set the value of the specified property for this junction.
sinkedTransitions	Return all inner and outer transitions whose destination is this junction.
sourcedTransitions	Return all inner and outer transitions whose source is this junction.
struct	Return a MATLAB structure that contains all of the property values for this junction.
up	Return a handle to the object that contains this junction.
view	Zoom in and select this junction.

Stateflow.Machine

The Stateflow machine contains all the charts in a Simulink model. You automatically create a `Machine` object when you load a model that contains a Stateflow chart or call the function `sfnew`.

To create a handle to the `Machine` object, use the `find` method of the `Root` object. For example, if `rt` is a handle to a `Root` object, enter:

```
m = rt.find('-isa','Stateflow.Machine');
```

For more information, see “Access the Machine Object” on page 1-7.

Properties

Name	Type	Access	Description
Created	Character vector	RO	Date of the creation of this machine.

Name	Type	Access	Description
`Creator`	Character vector	RW	Creator of this machine. Default value is `'Unknown'`.
`Debug.Animation.Delay`	Double	RW	Delay value to slow down the animation for charts in this machine. Default value is `0`.
`Debug.Animation.Enabled`	Boolean	RW	Enable animation for charts in this machine. Default value is `true`.
`Description`	Character vector	RW	Description of this machine. Default value is `''`.
`Dirty`	Boolean	RW	Indicates if the Simulink model for this machine has changed since being opened or saved.
`Document`	Character vector	RW	Document link for this machine. Default value is `''`.
`FullFileName`	Character vector	RO	Full path name of file that contains the Simulink model for this machine. Default value is `''`.
`Iced`	Boolean	RO	Equivalent to property `Locked`. Used internally to prevent changes in this machine during simulation.
`Id`	Integer	RO	Unique identifier that distinguishes this machine from other objects loaded in memory.
`IsLibrary`	Boolean	RO	Indicates if the Simulink model for this machine builds a library and not an application. Default value is `false`.
`Locked`	Boolean	RW	Prevents changes in the Simulink model for this machine. Default value is `false`.
`Machine`	`Machine`	RO	A handle to this `Machine` object.
`Modified`	Character vector	RW	Comment text for recording modifications to the Simulink model that defines this machine. Default value is `''`.
`Name`	Character vector	RO	Name of the Simulink model that defines this machine. Default value is `'untitled'`.
`Path`	Character vector	RO	Location of this machine in the model hierarchy.

Name	Type	Access	Description
`Tag`	Any type	RW	Tag for this machine. Holds data of any type. Default value is `[]`.
`Version`	Character vector	RW	Comment text for recording the version of the Simulink model that defines this machine. Default value is `'none'`.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `Machine` object type.
`dialog`	Open the Machine properties dialog box.
`disp`	Display all properties and values for this machine.
`find`	Find all objects inside this machine that match the specified criteria. Do not use the `-depth` switch with the `find` method for a `Machine` object.
`get`	Return the value of the specified property for this machine.
`help`	Display all properties and descriptions for this machine.
`methods`	Display all methods for this machine.
`parse`	Parse all the charts in this machine.
`set`	Set the value of the specified property for this machine.
`struct`	Return a MATLAB structure that contains all of the property values for this machine.

Stateflow.Message

To create a message in a parent chart, state, or box, use the constructor method `Stateflow.Message`. For example, if `ch` is a handle to a `Chart` object, enter:

```
m = Stateflow.Message(ch);
```

For more information, see “Communicate with Stateflow Charts by Sending Messages”.

Properties

Name	Type	Access	Description
`CompiledSize`	Character vector	RW	Size of the data for this message as determined by the compiler.
`CompiledType`	Character vector	RW	Type of the data for this message as determined by the compiler.
`DataType`	Character vector	RW	Type of the data for this message. • If the `Props.Type.Method` property of this data object is `'Built-in'`, you can specify this property with one of these options: `'double'` (default), `'single'`, `'int8'`, `'int16'`, `'int32'`, `'uint8'`, `'uint16'`, `'uint32'`, `'boolean'`, `'ml'`, or `'string'` (C charts only). • Otherwise, the `Props.Type` properties of this data object determine the value of this property.
`Description`	Character vector	RW	Description of this message. Default value is `''`.
`Document`	Character vector	RW	Document link for this message. Default value is `''`.
`Id`	Integer	RO	Unique identifier that distinguishes this message from other objects in the model.
`Machine`	`Machine`	RO	Machine that contains this message.
`MessagePriorityOrder`	Enum	RW	Type of priority queue for this message. Options are: • `'Ascending'` — Messages are received in ascending order of the message data value (default). • `'Descending'` — Messages are received in descending order of the message data value. This property applies only when the `QueueType` property of this message is `Priority`.

Name	Type	Access	Description
`Name`	Character vector	RW	Name of this message.
`Path`	Character vector	RO	Location of the parent of this message in the model hierarchy.
`Port`	Integer	RW	Port index for this message. This property applies only to input and output messages.
`Props.Array.Size`	Character vector	RW	Size of the data for this message. Default value is `'0'`. See “Specify Size of Stateflow Data”.
`Props.Complexity`	Enum	RW	Enable the data for this message to take complex values. Options are `'On'` or `'Off'` (default). See “Complex Data in Stateflow Charts”.
`Props.Frame`	Enum	RW	Enable the data for this message to accept frame-based signals. Options are: • `'Frame based'` — The message supports frame-based signals. • `'Sample based'` — The message supports sample-based signals (default).
`Props.InitialValue`	Character vector	RW	Initial value of the data for this message. Default value is `''`.
`Props.Type.BusObject`	Character vector	RW	Name of the `Simulink.Bus` object that defines the data for this message. This property applies only when the `Props.Type.Method` property of this message is `'Bus Object'`. See “Access Bus Signals Through Stateflow Structures”.
`Props.Type.EnumType`	Character vector	RW	Name of the enumerated type that defines the data for this message. This property applies only when the `Props.Type.Method` property of this message is `'Enumerated'`. See “Reference Values by Name by Using Enumerated Data”.

Name	Type	Access	Description
`Props.Type.Expression`	Character vector	RW	Expression that evaluates to a data type for the data for this message. This property applies only when the `Props.Type.Method` property of this message is `'Expression'`. See “Specify Data Properties by Using MATLAB Expressions”.
`Props.Type.Method`	Enum	RW	Method for setting the data type for this message. Options are `'Inherited'`, `'Built-in'`, `'Enumerated'`, `'Expression'`, or `'Bus Object'`. Equivalent to the **Mode** field of the Data Type Assistant in the Data properties dialog box. See “Specify Type of Stateflow Data”.
`QueueCapacity`	Integer	RW	Length of the internal queue for this message. Default value is `10`. This property applies only to input and local messages. For more information, see “Message Queue Properties”.
`QueueOverflowDiagnostic`	Enum	RW	Level of diagnostic action when the number of incoming messages exceeds the queue capacity for this message. Options are `'Error'` (default), `'Warning'`, or `'None'`. This property applies only to input and local messages. For more information, see “Message Queue Properties”.
`QueueType`	Enum	RW	Indicates the order in which messages are removed from the receiving queue. Options are: • `'FIFO'` — First in, first out (default). • `'LIFO'` — Last in, first out. • `'Priority'` — Remove messages according to the value in the data field. To specify the order, use the `MessagePriorityOrder` property for the message. This property applies only to input and local messages. For more information, see “Message Queue Properties”.

Name	Type	Access	Description
`Scope`	Enum	RW	Scope of this message. Options are `'Input'`, `'Local'`, or `'Output'` (default). For more information, see “Scope”.
`Tag`	Any type	RW	Tag for this message. Holds data of any type. Default value is `[]`.
`UseInternalQueue`	Boolean	RW	Indicates that the Stateflow chart maintains an internal receiving queue for this input message. Default value is `true`. This property applies only to input messages. For more information, see “Message Queue Properties”.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `Mesage` object type.
`delete`	Delete this message.
`dialog`	Open the Message properties dialog box.
`disp`	Display all properties and values for this message.
`get`	Return the value of the specified property for this message.
`help`	Display all properties and descriptions for this message.
`methods`	Display all methods for this message.
`set`	Set the value of the specified property for this message.
`struct`	Return a MATLAB structure that contains all of the property values for this message.
`up`	Return a handle to the object that contains this message.
`view`	Display this data object in the Model Explorer.

Stateflow.SimulinkBasedState

To create a Simulink based state in a parent chart, state, or box, use the constructor method `Stateflow.SimulinkBasedState`. For example, if `ch` is a handle to a `Chart` object, enter:

```
sbs = Stateflow.SimulinkBasedState(ch);
```

For more information, see “Simulink Subsystems as States”.

Properties

Name	Type	Access	Description
`ArrowSize`	Double	RW	Size of incoming transition arrows for this Simulink based state. Default value is `8`.
`BadIntersection`	Boolean	RO	Indicates if this Simulink based state graphically intersects a box, state, or function.
`Chart`	`Chart`	RO	Chart that contains this Simulink based state.
`ContentPreviewEnabled`	Boolean	RW	Display a preview of the contents of this Simulink based state at the Stateflow level. Default value is `true`.
`Debug.Breakpoints.OnDuring`	Boolean	RW	Set the `During State` breakpoint for this Simulink based state. Default value is `false`.
`Debug.Breakpoints.OnEntry`	Boolean	RW	Set the `On State Entry` breakpoint for this Simulink based state. Default value is `false`.
`Debug.Breakpoints.OnExit`	Boolean	RW	Set the `On State Exit` breakpoint for this Simulink based state. Default value is `false`.
`Description`	Character vector	RW	Description of this Simulink based state. Default value is `''`.
`Document`	Character vector	RW	Document link for this Simulink based state. Default value is `''`.
`ExecutionOrder`	Integer	RW	Order in which this Simulink based state wakes up in parallel (AND) decomposition. This property applies only when the `UserSpecifiedStateTransitionExecutionOrder` property of the chart that contains the state is `true`.
`FontSize`	Double	RW	Font size for the label of this Simulink based state. The `StateFont.Size` property of the chart that contains the state sets the initial value of this property.

Name	Type	Access	Description
`HasOutputData`	Boolean	RW	Create an active state data output port for this Simulink based state. Default value is `false`. See “Monitor State Activity Through Active State Data”.
`Id`	Integer	RO	Unique identifier that distinguishes this Simulink based state from other objects in the model.
`IsExplicitlyCommented`	Boolean	RW	Explicitly comment out this Simulink based state. Default value is `false`. Equivalent to right-clicking the state and selecting **Comment Out**.
`IsImplicitlyCommented`	Boolean	RO	Indicates if this Simulink based state is implicitly commented out. A state is implicitly commented out when you comment out a superstate in its hierarchy.
`IsLink`	Boolean	RO	Indicates if this Simulink based state is a library link.
`LabelString`	Character vector	RW	Full label of this Simulink based state. Default value is `'?'`.
`LoggingInfo.DataLogging`	Boolean	RW	Enable signal logging for this Simulink based state. Default value is `false`.
`LoggingInfo.DecimateData`	Boolean	RW	Limit the amount of data logged by skipping samples. Uses the interval specified by the `LoggingInfo.Decimation` property of this Simulink based state. Default value is `false`.
`LoggingInfo.Decimation`	Integer	RW	Decimation interval. Default value is `2`. This value means the chart logs every other sample of this Simulink based state.
`LoggingInfo.LimitDataPoints`	Boolean	RW	Limit the number of data points to log. Uses the value specified by the `LoggingInfo.MaxPoints` property of this Simulink based state. Default value is `false`.
`LoggingInfo.MaxPoints`	Integer	RW	Maximum number of data points to log. Default value is `5000`. This value means the chart logs the last 5000 data points generated by the simulation for this Simulink based state.

Name	Type	Access	Description
`LoggingInfo.NameMode`	Enum	RW	Source of the signal name used to log this Simulink based state. Options are: • `'Custom'` — Use the custom signal name specified by the `LoggingInfo.LoggingName` property. • `'SignalName'` — Use the name of the Simulink based state (default).
`LoggingInfo.LoggingName`	Character vector	RW	Custom signal name used for logging this Simulink based state.
`Machine`	`Machine`	RO	Machine that contains this Simulink based state.
`Name`	Character vector	RW	Name of this Simulink based state. Default value is `''`.
`OutputData`	`Data`	RO	Active state data object for this Simulink based state. This property applies only when the `HasOutputData` property for this state is `true`. See “Monitor State Activity Through Active State Data”.
`OutputMonitoringMode`	Character vector	RO	Indicates the monitoring mode for the active state output data. For Simulink based states, the only option is `'SelfActivity'`.
`Path`	Character vector	RO	Location of the parent of this Simulink based state in the model hierarchy.
`Position`	Numeric vector	RW	Position and size of this Simulink based state in the chart. Numeric vector `[left top width height]`. Default value is `[0 0 90 60]`.
`Subviewer`	`Chart`, `State`, or `Box`	RO	Chart or subchart where you can graphically view this Simulink based state.
`Tag`	Any type	RW	Tag for this Simulink based state. Holds data of any type. Default value is `[]`.
`TestPoint`	Boolean	RW	Set this Simulink based state as a Stateflow test point. Default value is `false`. See “Monitor Test Points in Stateflow Charts”.

Name	Type	Access	Description
`Type`	Enum	RO	Type of decomposition for this Simulink based state. Options are `'AND'` (parallel) or `'OR'` (exclusive). The Simulink based state inherits this property from the `Decomposition` property of its parent.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `SimulinkBasedState` object type.
`delete`	Delete this Simulink based state.
`dialog`	Open the Simulink Based State properties dialog box.
`disp`	Display all properties and values for this Simulink based state.
`find`	Find all objects inside this Simulink based state that match the specified criteria.
`fitToView`	Zoom in on this Simulink based state in the chart.
`get`	Return the value of the specified property for this Simulink based state.
`help`	Display all properties and descriptions for this Simulink based state.
`highlight`	Highlight this Simulink based state in the chart.
`isCommented`	Return a Boolean value that indicates if this Simulink based state is explicitly or implicitly commented out.
`methods`	Display all methods for this Simulink based state.
`set`	Set the value of the specified property for this Simulink based state.
`struct`	Return a MATLAB structure that contains all of the property values for this Simulink based state.
`up`	Return a handle to the object that contains this Simulink based state.
`view`	Display the contents of this Simulink based state.

Stateflow.SLFunction

To create a Simulink function in a parent chart, state, box, or function, use the constructor method `Stateflow.SLFunction`. For example, if `ch` is a handle to a `Chart` object, enter:

```
f = Stateflow.SLFunction(ch);
```

For more information, see “Reuse Simulink Components in Stateflow Charts”.

Properties

Name	Type	Access	Description
`BadIntersection`	Boolean	RO	Indicates if this Simulink function graphically intersects a box, state, or function.
`Chart`	`Chart`	RO	Chart that contains this Simulink function.
`Description`	Character vector	RW	Description of this Simulink function. Default value is `''`.
`Document`	Character vector	RW	Document link for this Simulink function. Default value is `''`.
`FontSize`	Double	RW	Font size for the label of this Simulink function. The `StateFont.Size` property of the chart that contains the function sets the initial value of this property.
`Id`	Integer	RO	Unique identifier that distinguishes this Simulink function from other objects in the model.
`IsExplicitlyCommented`	Boolean	RW	Explicitly comment out this Simulink function. Default value is `false`. Equivalent to right-clicking the function and selecting **Comment Out**.
`IsImplicitlyCommented`	Boolean	RO	Indicates if this Simulink function is implicitly commented out. A function is implicitly commented out when you comment out a superstate in its hierarchy.
`LabelString`	Character vector	RW	Full label of this Simulink function. Label syntax is `'return = Name(arguments)'`. Default value is `'()'`.

Name	Type	Access	Description
`Machine`	`Machine`	RO	Machine that contains this Simulink function.
`Name`	Character vector	RW	Name of this Simulink function. Default value is `''`.
`Path`	Character vector	RO	Location of the parent of this Simulink function in the model hierarchy.
`Position`	Numeric vector	RW	Position and size of this Simulink function in the chart. Numeric vector `[left top width height]`. Default value is `[0 0 90 60]`.
`Subviewer`	`Chart`, `State`, `Box`, or `Function`	RO	Chart or subchart where you can graphically view this Simulink function.
`Tag`	Any type	RW	Tag for this Simulink function. Holds data of any type. Default value is `[]`.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `SLFunction` object type.
`delete`	Delete this function.
`dialog`	Open the Block Parameters properties dialog box.
`disp`	Display all properties and values for this function.
`find`	Find all objects inside this function that match the specified criteria.
`fitToView`	Zoom in on this function in the chart.
`get`	Return the value of the specified property for this function.
`help`	Display all properties and descriptions for this function.
`highlight`	Highlight this function in the chart.
`isCommented`	Return a Boolean value that indicates if this function is explicitly or implicitly commented out.
`methods`	Display all methods for this function.
`set`	Set the value of the specified property for this function.

Name	Description
`struct`	Return a MATLAB structure that contains all of the property values for this function.
`up`	Return a handle to the object that contains this function.
`view`	Display the contents of this function.

Stateflow.State

To create a state in a parent chart, state, or box, use the constructor method `Stateflow.State`. For example, if `ch` is a handle to a `Chart` object, enter:

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

For more information, see “Represent Operating Modes by Using States”.

Properties

Name	Type	Access	Description
`ArrowSize`	Double	RW	Size of incoming transition arrows for this state. Default value is `8`.
`BadIntersection`	Boolean	RO	Indicates if this state graphically intersects a box, state, or function.
`Chart`	`Chart`	RO	Chart that contains this state.
`Debug.Breakpoints.OnDuring`	Boolean	RW	Set the `During State` breakpoint for this state. Default value is `false`.
`Debug.Breakpoints.OnEntry`	Boolean	RW	Set the `On State Entry` breakpoint for this state. Default value is `false`.
`Debug.Breakpoints.OnExit`	Boolean	RW	Set the `On State Exit` breakpoint for this state. Default value is `false`.
`Decomposition`	Enum	RW	State decomposition at the top level of containment in this state. Options are `'EXCLUSIVE_OR'` (default) and `'PARALLEL_AND'`.
`Description`	Character vector	RW	Description of this state. Default value is `''`.

Name	Type	Access	Description
`Document`	Character vector	RW	Document link for this state. Default value is `''`.
`ExecutionOrder`	Integer	RW	Order in which this state wakes up in parallel (AND) decomposition. This property applies only when the `UserSpecifiedStateTransitionExecutionOrder` property of the chart that contains the state is `true`.
`FontSize`	Double	RW	Font size for the label of this state. The `StateFont.Size` property of the chart that contains the state sets the initial value of this property.
`HasOutputData`	Boolean	RW	Create an active state data output port for this state. Default value is `false`. See “Monitor State Activity Through Active State Data”.
`Id`	Integer	RO	Unique identifier that distinguishes this state from other objects in the model.
`InlineOption`	Character vector	RW	Specify how this state appears in generated code. Options are: • `'Inline'` — Calls to state functions are replaced by code. • `'Function'` — State functions are implemented as separate C functions. • `'Auto'` — An internal calculation determines the appearance of state functions in generated code (default). For more information, see “Inline State Functions in Generated Code” (Simulink Coder).
`IsExplicitlyCommented`	Boolean	RW	Explicitly comment out this state. Default value is `false`. Equivalent to right-clicking the state and selecting **Comment Out**.
`IsGrouped`	Boolean	RW	Specify if this state is a group. Default value is `false`. See “Copy by Grouping” on page 1-31.

Name	Type	Access	Description
`IsImplicitlyCommented`	Boolean	RO	Indicates if this state is implicitly commented out. A state is implicitly commented out when you comment out a superstate in its hierarchy.
`IsSubchart`	Boolean	RW	Specify if this state is a subchart. Default value is `false`.
`LabelString`	Character vector	RW	Full label of this state. Default value is `'?'`. See “Enter Multiline Labels in States and Transitions” on page 1-38.
`LoggingInfo.DataLogging`	Boolean	RW	Enable signal logging for this state. Default value is `false`.
`LoggingInfo.DecimateData`	Boolean	RW	Limit the amount of data logged by skipping samples. Uses the interval specified by the `LoggingInfo.Decimation` property of this state. Default value is `false`.
`LoggingInfo.Decimation`	Integer	RW	Decimation interval. Default value is `2`. This value means the chart logs every other sample of this state.
`LoggingInfo.LimitDataPoints`	Boolean	RW	Limit the number of data points to log. Uses the value specified by the `LoggingInfo.MaxPoints` property of this state. Default value is `false`.
`LoggingInfo.MaxPoints`	Integer	RW	Maximum number of data points to log. Default value is `5000`. This value means the chart logs the last 5000 data points generated by the simulation for this state.
`LoggingInfo.NameMode`	Enum	RW	Source of the signal name used to log this state. Options are: • `'Custom'` — Use the custom signal name specified by the `LoggingInfo.LoggingName` property. • `'SignalName'` — Use the name of the state (default).

Name	Type	Access	Description
`LoggingInfo.LoggingName`	Character vector	RW	Custom signal name used for logging this state.
`Machine`	`Machine`	RO	Machine that contains this state.
`Name`	Character vector	RW	Name of this state. Default value is `''`.
`OutputData`	`Data`	RO	Active state data object for this state. This property applies only when the `HasOutputData` property for this state is `true`. See “Monitor State Activity Through Active State Data”.
`OutputMonitoringMode`	Enum	RW	Indicates the monitoring mode for the active state output data. Options are `'ChildActivity'`, `'LeafStateActivity'`, or `'SelfActivity'` (default). This property applies only when the `HasOutputData` property for this state is `true`.
`Path`	Character vector	RO	Location of the parent of this state in the model hierarchy.
`Position`	Numeric vector	RW	Position and size of this state in the chart. Numeric vector `[left top width height]`. Default value is `[0 0 90 60]`.
`Subviewer`	`Chart`, `State`, or `Box`	RO	Chart or subchart where you can graphically view this state.
`Tag`	Any type	RW	Tag for this state. Holds data of any type. Default value is `[]`.
`TestPoint`	Boolean	RW	Set this state as a Stateflow test point. Default value is `false`. See “Monitor Test Points in Stateflow Charts”.
`Type`	Enum	RO	Type of decomposition for this state. Options are `'AND'` (parallel) or `'OR'` (exclusive). The state inherits this property from the `Decomposition` property of its parent.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `State` object type.
`defaultTransitions`	Return the default transitions at the top level of containment of this state.
`delete`	Delete this state.
`dialog`	Open the State properties dialog box.
`disp`	Display all properties and values for this state.
`find`	Find all objects inside this state that match the specified criteria.
`fitToView`	Zoom in on this state in the chart.
`get`	Return the value of the specified property for this state.
`help`	Display all properties and descriptions for this state.
`highlight`	Highlight this state in the chart.
`innerTransitions`	Return an array of the transitions that originate in this state and terminate on a contained object.
`isCommented`	Return a Boolean value that indicates if this state is explicitly or implicitly commented out.
`methods`	Display all methods for this state.
`outerTransitions`	Return an array of the transitions that exit the outer edge of this state and terminate on an object outside the containment of this state.
`set`	Set the value of the specified property for this state.
`sinkedTransitions`	Return all inner and outer transitions whose destination is this state.
`sourcedTransitions`	Return all inner and outer transitions whose source is this state.
`struct`	Return a MATLAB structure that contains all of the property values for this state.
`up`	Return a handle to the object that contains this state.
`view`	Zoom in and select this state. If the state is a subchart, display its contents.

Stateflow.StateTransitionTableChart

To create a Simulink model that contains an empty State Transition Table block, call the function `sfnew -STT`. For more information, see “State Transition Tables in Stateflow”.

Properties

Name	Type	Access	Description
`ActionLanguage`	Enum	RW	Action language used to program the state transition table. Options are `C` or `MATLAB` (default). See “Differences Between MATLAB and C as Action Language Syntax”.
`ChartColor`	Numeric vector	RW	Background color of the automatically generated chart for this state transition table. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[1 0.9608 0.8824]`.
`ChartUpdate`	Enum	RW	Activation method of this state transition table. Options are `'INHERITED'` (default), `'DISCRETE'`, or `'CONTINUOUS'`. See “Update Method”.
`Debug.Breakpoints.OnEntry`	Boolean	RW	Set the `On Chart Entry` breakpoint for this state transition table. Default value is `false`.
`Description`	Character vector	RW	Description of this state transition table. Default value is `''`.
`Dirty`	Boolean	RW	Indicates if this state transition table has changed since being opened or saved. Default value is `false`.
`Document`	Character vector	RW	Document link for this state transition table. Default value is `''`.
`Editor`	`Editor`	RO	`Editor` object for this state transition table.

Name	Type	Access	Description
`EmlDefaultFimath`	Character vector	RW	Default `fimath` properties for this state transition table. Options are: • `'Same as MATLAB Default'` — Use the same `fimath` properties as the current default `fimath` (default). • `'Other:UserSpecified'` — Use the `InputFimath` property to specify the default `fimath` object. This property applies only to state transition tables that use MATLAB as the action language.
`EnableBitOps`	Boolean	RW	Use C-bit operations in state and transition actions in this state transition table. Default value is `false`. This property applies only to state transition tables that use C as the action language. See “Supported Operations for Chart Data”.
`EnableNonTerminalStates`	Boolean	RW	Use super step semantics for this state transition table. Default value is `false`. See “Super Step Semantics”.
`EnableZeroCrossings`	Boolean	RW	Use zero-crossing detection on state transitions in this state transition table. Default value is `true`. This property applies only when the `ChartUpdate` property for this state transition table is set to `'CONTINUOUS'`. See “Disable Zero-Crossing Detection”.
`ErrorColor`	Numeric vector	RW	Color for errors in this state transition table. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[1 0 0]`.
`ExecuteAtInitialization`	Boolean	RW	Initialize the state configuration of this state transition table at time zero instead of at the first input event. Default value is `false`. See “Execution of a Chart at Initialization”.

Name	Type	Access	Description
`HasOutputData`	Boolean	RW	Create an active state data output port for this state transition table. Default value is `false`. See “Monitor State Activity Through Active State Data”.
`Iced`	Boolean	RO	Equivalent to property `Locked`. Used internally to prevent changes in this state transition table during simulation.
`Id`	Integer	RO	Unique identifier that distinguishes this state transition table from other objects in the model.
`InitializeOutput`	Boolean	RW	Apply the initial value of the output data every time this state transition table wakes up. Default value is `false`. See “Initialize Outputs Every Time Chart Wakes Up”.
`InputFimath`	Character vector	RW	Specify the `embedded.fimath` object associated with inputs from Simulink blocks. You can: • Enter an expression that constructs a `fimath` object. • Enter the variable name for a `fimath` object in the MATLAB or model workspace. This property applies only when the `EmlDefaultFimath` property for this state transition table is `'Other:UserSpecified'`.
`JunctionColor`	Numeric vector	RW	Color for junctions in the automatically generated chart for this state transition table. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0.6824 0.3294 0]`.
`Locked`	Boolean	RW	Prevent changes in this state transition table. Default value is `false`.
`Machine`	`Machine`	RO	Machine that contains this state transition table.
`Name`	Character vector	RW	Name of this state transition table. Default value is `'State Transition Table'`.

Name	Type	Access	Description
`NonTerminalMaxCounts`	Character vector	RW	Maximum number of transitions this state transition table can take in one super step. Default value is `1000`. This property applies only when the `EnableNonTerminalStates` property for this state transition table is `true`. See “Super Step Semantics”.
`NonTerminalUnstableBehavior`	Enum	RW	Behavior during simulation if this state transition table exceeds the maximum number of transitions specified in the `NonTerminalMaxCounts` property before reaching a stable state. Options are: • `'PROCEED'` — The state transition table goes to sleep with the last active state configuration (default). • `'THROW ERROR'` — The state transition table generates an error. This property applies only when the `EnableNonTerminalStates` property for this state transition table is `true`. See “Super Step Semantics”.
`OutputData`	`Data`	RO	Active state data object for this state transition table. This property applies only when the `HasOutputData` property for this state transition table is `true`. See “Monitor State Activity Through Active State Data”.
`OutputMonitoringMode`	Character vector	RW	Indicates the monitoring mode for the active state output data. Options are `'ChildActivity'` (default) or `'LeafStateActivity'`. This property applies only when the `HasOutputData` property for this state transition table is `true`. See “Monitor State Activity Through Active State Data”.
`Path`	Character vector	RO	Location of this state transition table in the model hierarchy.

Name	Type	Access	Description
`SampleTime`	Character vector	RW	Sample time for activating this state transition table. Default value is `'-1'`. This property applies only when the `ChartUpdate` property for this state transition table is `'DISCRETE'`.
`SaturateOnIntegerOverflow`	Boolean	RW	Specify the behavior of integer overflows in this state transition table. Options are: • `true` — The state transition table saturates integer overflows (default). • `false` — The state transition table wraps integer overflows. For more information, see “Handle Integer Overflow for Chart Data”.
`StateColor`	Numeric vector	RW	Color for state boxes in the automatically generated chart for this state transition table. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0 0 0]`.
`StateFont.Angle`	Enum	RW	Font angle for the state labels in the automatically generated chart for this state transition table. Options are `'ITALIC'` or `'NORMAL'` (default).
`StateFont.Name`	Character vector	RW	Font for the state labels in the automatically generated chart for this state transition table. Default value is `'Helvetica'`.
`StateFont.Size`	Integer	RW	Font size for the state labels in the automatically generated chart for this state transition table. Default value is `12`.
`StateFont.Weight`	Enum	RW	Font weight for the state labels in the automatically generated chart for this state transition table. Options are `'BOLD'` or `'NORMAL'` (default).

Name	Type	Access	Description
`StateLabelColor`	Numeric vector	RW	Color for state labels in the automatically generated chart for this state transition table. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0 0 0]`.
`StateMachineType`	Enum	RW	Type of state machine semantics. Options are `'Classic'` (default), `'Mealy'`, or `'Moore'`. See “Overview of Mealy and Moore Machines”.
`StatesWhenEnabling`	Enum	RW	Specify behavior of states when a function-call input event reenables this state transition table. Options are: • `'held'` — The state transition table maintains most recent values of the states (default). • `'reset'` — The state transition table reverts to the initial conditions of the states. This property applies only when the state transition table contains function-call input events. See “Control States in Charts Enabled by Function-Call Input Events”.
`StrongDataTypingWithSimulink`	Boolean	RW	Use strong data typing when this state transition table interfaces with Simulink input and output signals. Default value is `true`. This property applies only to state transition tables that use C as the action language. See “Use Strong Data Typing with Simulink I/O”.
`SupportVariableSizing`	Boolean	RW	Support input and output data that vary in dimension during simulation in this state transition table. Default value is `true`. See “Declare Variable-Size Data in Stateflow Charts”.
`Tag`	Any type	RW	Tag for this state transition table. Holds data of any type. Default value is `[]`.

Name	Type	Access	Description
`TransitionColor`	Numeric vector	RW	Color for transitions in the automatically generated chart for this state transition table. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0.2902 0.3294 0.6039]`.
`TransitionFont.Angle`	Enum	RW	Font angle for transition labels in the automatically generated chart for this state transition table. Options are `'ITALIC'` or `'NORMAL'` (default).
`TransitionFont.Name`	Character vector	RW	Font for transition labels in the automatically generated chart for this state transition table. Default value is `'Helvetica'`.
`TransitionFont.Size`	Integer	RW	Font size for transition labels in the automatically generated chart for this state transition table. Default value is `12`.
`TransitionFont.Weight`	Enum	RW	Font weight for transition labels in the automatically generated chart for this state transition table. Options are `'BOLD'` or `'NORMAL'` (default).
`TransitionLabelColor`	Numeric vector	RW	Color for transition labels in the automatically generated chart for this state transition table. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0.2902 0.3294 0.6039]`.

Name	Type	Access	Description
`TreatAsFi`	Enum	RW	Treat inherited fixed-point and integer signals as Fixed-Point Designer `fi` objects. Options are: • `'Fixed-point'` — The state transition table treats all fixed-point inputs as `fi` objects (default). • `'Fixed-point & Integer'` — The state transition table treats all fixed-point and integer inputs as `fi` objects. This property applies only to state transition tables that use MATLAB as the action language.
`Visible`	Boolean	RW	Indicates if the editor is displaying this state transition table.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `StateTransitionTableChart` object type.
`dialog`	Open the State Transition Table properties dialog box.
`disp`	Display all properties and values for this state transition table.
`find`	Find all objects inside this state transition table that match the specified criteria.
`get`	Return the value of the specified property for this state transition table.
`help`	Display all properties and descriptions for this state transition table.
`methods`	Display all methods for this state transition table.
`parse`	Parse this state transition table.
`set`	Set the value of the specified property for this state transition table.
`struct`	Return a MATLAB structure that contains all of the property values for this state transition table.
`view`	Display the contents of this state transition table.

Stateflow.Transition

To create a transition in a parent chart, state, box, or function, use the constructor method `Stateflow.Transition`. For example, if `ch` is a handle to a `Chart` object, enter:

```
tr = Stateflow.Transition(ch);
```

For more information, see “Transition Between Operating Modes”.

Properties

Name	Type	Access	Description
`ArrowSize`	Double	RW	Size of the arrow for this transition. Default value is `10`.
`Chart`	`Chart`	RO	Chart that contains this transition.
`Debug.Breakpoints.WhenTested`	Boolean	RW	Set the `When Transition is Tested` breakpoint for this transition. Default value is `false`.
`Debug.Breakpoints.WhenValid`	Boolean	RW	Set the `When Transition is Valid` breakpoint for this transition. Default value is `false`.
`Description`	Character vector	RW	Description of this transition. Default value is `''`.
`Destination`	`State`, `Box`, or `Junction`	RW	Destination state, box, or junction for this transition. Default value is `[]`.
`DestinationEndPoint`	Numeric vector	RW	Position of the transition endpoint at its destination. Numeric vector `[x y]` of coordinates relative to the upper left corner of the chart. Default value is `[40 40]`.
`DestinationOClock`	Double	RW	Location of the transition endpoint at its destination. Numeric value between 0 and 12 that specifies a clock position. Default value is `0`.
`Document`	Character vector	RW	Document link for this transition. Default value is `''`.

Name	Type	Access	Description
`ExecutionOrder`	Integer	RW	Order in which this transition executes when its source is active. This property applies only when the `UserSpecifiedStateTransitionExecutionOrder` property of the chart that contains the transition is `true`. See “Transition Evaluation Order”.
`FontSize`	Double	RW	Font size for the label on this transition. The `TransitionFont.Size` property of the chart that contains the state sets the initial value of this property.
`Id`	Integer	RO	Unique identifier that distinguishes this transition from other objects in the model.
`IsExplicitlyCommented`	Boolean	RW	Explicitly comment out this transition. Default value is `false`. Equivalent to right-clicking the transition and selecting **Comment Out**.
`IsImplicitlyCommented`	Boolean	RO	Indicates if this transition is implicitly commented out. A transition is implicitly commented out when you comment out a superstate in its hierarchy or a state or junction to which the transition is attached.
`LabelPosition`	Numeric vector	RW	Position and size of this label on this transition in the chart. Numeric vector `[left top width height]`. Default value is `[0 0 8 14]`.
`LabelString`	Character vector	RW	Full label on this transition. Default value is `'?'`. See “Enter Multiline Labels in States and Transitions” on page 1-38.
`Machine`	`Machine`	RO	Machine that contains this transition.
`MidPoint`	Numeric vector	RW	Coordinates of the midpoint of the transition. Numeric vector `[x y]` of coordinates relative to the upper left corner of the chart. Default value is `[21 21]`.
`Path`	Character vector	RO	Location of the parent of this transition in the model hierarchy.

Name	Type	Access	Description
`Source`	`State`, `Box`, or `Junction`	RW	Source state, box, or junction of this transition. Default value is `[]`.
`SourceEndPoint`	Double	RW	Position of the transition endpoint at its source. Numeric vector `[x y]` of coordinates relative to the upper left corner of the chart. Default value is `[2 2]`.
`SourceOClock`	Double	RW	Location of the transition endpoint at its source. Numeric value between 0 and 12 that specifies a clock position. Default value is `0`.
`Subviewer`	`Chart`, `State`, `Box`, or `Function`	RO	Chart or subchart where you can graphically view this transition.
`Tag`	Any type	RW	Tag for this transition. Holds data of any type. Default value is `[]`.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `Transition` object type.
`delete`	Delete this transition.
`dialog`	Open the Transition properties dialog box.
`disp`	Display all properties and values for this transition.
`fitToView`	Zoom in on this transition in the chart.
`get`	Return the value of the specified property for this transition.
`help`	Display all properties and descriptions for this transition.
`highlight`	Highlight this transition in the chart.
`isCommented`	Return a Boolean value that indicates if this transition is explicitly or implicitly commented out.
`methods`	Display all methods for this transition.

Name	Description
`set`	Set the value of the specified property for this transition.
`struct`	Return a MATLAB structure that contains all of the property values for this transition.
`up`	Return a handle to the object that contains this transition.
`view`	Zoom in and select this transition.

Stateflow.TruthTable

To create a truth table function in a parent chart, state, box, or function, use the constructor method `Stateflow.TruthTable`. For example, if `ch` is a handle to a `Chart` object, enter:

```
f = Stateflow.TruthTable(ch);
```

For more information, see “Reuse Combinatorial Logic by Defining Truth Tables”.

Properties

Name	Type	Access	Description
`ActionTable`	Cell array of character vectors	RW	Action table for this truth table function. Default value is `[]`.
`BadIntersection`	Boolean	RO	Indicates if this truth table function graphically intersects a box, state, or function.
`Chart`	`Chart`	RO	Chart that contains this truth table function.
`ConditionTable`	Cell array of character vectors	RW	Condition table for this truth table function. Default value is `[]`.
`Debug.Breakpoints.OnDuring`	Boolean	RW	Set the `During Function Call` breakpoint for this truth table function. Default value is `false`.
`Description`	Character vector	RW	Description of this truth table function. Default value is `''`.

Name	Type	Access	Description
`Document`	Character vector	RW	Document link for this truth table function. Default value is `''`.
`FontSize`	Double	RW	Font size for the label of this truth table function. The `StateFont.Size` property of the chart that contains the function sets the initial value of this property.
`Id`	Integer	RO	Unique identifier that distinguishes this truth table function from other objects in the model.
`InlineOption`	Character vector	RW	Specify how this truth table function appears in generated code. Options are: • `'Inline'` — Calls to the truth table function are replaced by code. • `'Function'` — The truth table function is implemented as a separate C function. • `'Auto'` — An internal calculation determines the appearance of function calls in generated code (default). For more information, see “Inline State Functions in Generated Code” (Simulink Coder).
`IsExplicitlyCommented`	Boolean	RW	Explicitly comment out this truth table function. Default value is `false`. Equivalent to right-clicking the function and selecting **Comment Out**.
`IsImplicitlyCommented`	Boolean	RO	Indicates if this truth table function is implicitly commented out. A function is implicitly commented out when you comment out a superstate in its hierarchy.
`LabelString`	Character vector	RW	Full label of this truth table function. Label syntax is `'`*`return = Name(arguments)`*`'`. Default value is `'()'`.

Name	Type	Access	Description
`Language`	Enum	RW	Action language used to program the truth table function. Options are `C` or `MATLAB` (default). See “Differences Between MATLAB and C as Action Language Syntax”.
`Machine`	`Machine`	RO	Machine that contains this truth table function.
`Name`	Character vector	RW	Name of this truth table function. Default value is `''`.
`OverSpecDiagnostic`	Character vector	RW	Level of diagnostic action when this truth table function is overspecified. Options are `'Error'` (default), `'Warning'`, or `'None'`. See “Correct Overspecified and Underspecified Truth Tables”.
`Path`	Character vector	RO	Location of the parent of this truth table function in the model hierarchy.
`Position`	Numeric vector	RW	Position and size of this truth table function in the chart. Numeric vector `[left top width height]`. Default value is `[0 0 90 60]`.
`Subviewer`	`Chart`, `State`, `Box`, or `Function`	RO	Chart or subchart where you can graphically view this truth table function.
`Tag`	Any type	RW	Tag for this truth table function. Holds data of any type. Default value is `[]`.
`UnderSpecDiagnostic`	Character vector	RW	Level of diagnostic action when this truth table function is underspecified. Options are `'Error'` (default), `'Warning'`, or `'None'`. See “Correct Overspecified and Underspecified Truth Tables”.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `TruthTable` object type.
`delete`	Delete this function.
`dialog`	Open the Truth Table properties dialog box.

Name	Description
`disp`	Display all properties and values for this function.
`find`	Find all objects inside this function that match the specified criteria.
`fitToView`	Zoom in on this function in the chart.
`get`	Return the value of the specified property for this function.
`help`	Display all properties and descriptions for this function.
`highlight`	Highlight this function in the chart.
`isCommented`	Return a Boolean value that indicates if this function is explicitly or implicitly commented out.
`methods`	Display all methods for this function.
`set`	Set the value of the specified property for this function.
`struct`	Return a MATLAB structure that contains all of the property values for this function.
`up`	Return a handle to the object that contains this function.
`view`	Display the contents of this function.

Stateflow.TruthTableChart

To create a Simulink model that contains an empty Truth Table block, call the function `sfnew -TT`. For more information, see “Reuse Combinatorial Logic by Defining Truth Tables”.

Properties

Name	Type	Access	Description
`ActionTable`	Cell array of character vectors	RW	Action table for this truth table. Default value is `[]`.
`ChartUpdate`	Enum	RW	Activation method of this truth table. Options are `'INHERITED'` (default), `'DISCRETE'`, or `'CONTINUOUS'`. See “Update Method”.

Name	Type	Access	Description
`ConditionTable`	Cell array of character vectors	RW	Condition table for this truth table. Default value is `[]`.
`Description`	Character vector	RW	Description of this truth table. Default value is `''`.
`Dirty`	Boolean	RW	Indicates if this truth table has changed since being opened or saved. Default value is `false`.
`Document`	Character vector	RW	Document link for this truth table. Default value is `''`.
`EmlDefaultFimath`	Character vector	RW	Default `fimath` properties for this truth table. Options are: • `'Same as MATLAB Default'` — Use the same `fimath` properties as the current default `fimath` (default). • `'Other:UserSpecified'` — Use the `InputFimath` property to specify the default `fimath` object.
`Iced`	Boolean	RO	Equivalent to property `Locked`. Used internally to prevent changes in this truth table during simulation.
`Id`	Integer	RO	Unique identifier that distinguishes this truth table from other objects in the model.
`InputFimath`	Character vector	RW	Specify the `embedded.fimath` object associated with inputs from Simulink blocks. You can: • Enter an expression that constructs a `fimath` object. • Enter the variable name for a `fimath` object in the MATLAB or model workspace. This property applies only when the `EmlDefaultFimath` property for this truth table is `'Other:UserSpecified'`.

Name	Type	Access	Description
`Locked`	Boolean	RW	Prevent changes in this truth table. Default value is `false`.
`Machine`	`Machine`	RO	Machine that contains this truth table.
`Name`	Character vector	RW	Name of this truth table. Default value is `'Truth Table'`.
`OverSpecDiagnostic`	Enum	RW	Level of diagnostic action when this truth table is overspecified. Options are `'Error'` (default), `'Warning'`, or `'None'`. See “Correct Overspecified and Underspecified Truth Tables”.
`Path`	Character vector	RO	Location of this truth table in the model hierarchy.
`SampleTime`	Character vector	RW	Sample time for activating this truth table. Default value is `'-1'`. This property applies only when the `ChartUpdate` property for this truth table is `'DISCRETE'`.
`SaturateOnIntegerOverflow`	Boolean	RW	Specify behavior of integer overflows in this truth table. Options are: • `true` — The truth table saturates integer overflows (default). • `false` — The truth table wraps integer overflows. For more information, see “Handle Integer Overflow for Chart Data”.

Name	Type	Access	Description
`StatesWhenEnabling`	Enum	RW	Specify behavior of states when a function-call input event reenables this truth table. Options are: • `'held'` — The truth table maintains the most recent values of the states (default). • `'reset'` — The truth table reverts to the initial conditions of the states. This property applies only when the truth table contains function-call input events. See “Control States in Charts Enabled by Function-Call Input Events”.
`SupportVariableSizing`	Boolean	RW	Support input and output data that vary in dimension during simulation in this truth table. Default value is `true`. See “Declare Variable-Size Data in Stateflow Charts”.
`Tag`	Any type	RW	Tag for this truth table. Holds data of any type. Default value is `[]`.
`TreatAsFi`	Enum	RW	Treat inherited fixed-point and integer signals as Fixed-Point Designer `fi` objects. Options are: • `'Fixed-point'` — The truth table treats all fixed-point inputs as `fi` objects (default). • `'Fixed-point & Integer'` — The truth table treats all fixed-point and integer inputs as `fi` objects.
`UnderSpecDiagnostic`	Enum	RW	Level of diagnostic action when this truth table is underspecified. Options are `'Error'` (default), `'Warning'`, or `'None'`. See “Correct Overspecified and Underspecified Truth Tables”.

Methods

Name	Description
`classhandle`	Return a read-only handle to the schema class of the `TruthTableChart` object type.
`dialog`	Open the Truth Table properties dialog box.
`disp`	Display all properties and values for this truth table.
`find`	Find all objects inside this truth table that match the specified criteria.
`get`	Return the value of the specified property for this truth table.
`help`	Display all properties and descriptions for this truth table.
`methods`	Display all methods for this truth table.
`parse`	Parse this truth table.
`set`	Set the value of the specified property for this truth table.
`struct`	Return a MATLAB structure that contains all of the property values for this truth table.
`view`	Display the contents of this truth table.

See Also

`sfclipboard` | `sfnew` | `sfroot`

More About

- “Create Charts by Using the Stateflow API” on page 1-7
- “Create and Destroy Stateflow Objects” on page 1-19
- “Access Properties and Methods of Stateflow Objects” on page 1-14
- “Copy and Paste Stateflow Objects” on page 1-30
- “Modify the Graphical Properties of Your Chart” on page 1-36

3

API Object Properties and Methods

Properties and Methods Sorted By Application

The following reference tables for Stateflow API properties and methods have these columns:

- **Name** — The name of the property or method. To access or set a property value or to call a method, use its name in dot notation along with a Stateflow object. Properties with multiple levels of hierarchy (such as the `LoggingInfo` and `Props` properties of data objects) must be set individually. For more information, see “Access Properties and Methods of Stateflow Objects” on page 1-14.
- **Type** — The data type for the property. Some property types are other Stateflow API objects. For example, the `Machine` property of an object is the `Stateflow.Machine` object that contains the object.
- **Access** — An access type for the property.
  - RW (read/write): You can access or set the value of these properties by using the Stateflow API.
  - RO (read-only): These properties are set by the Stateflow software.
- **Description** — A description of the property or method.
- **Objects** — The types of objects that have this property or method. The object types are listed as: Annotation (A on page 2-4), Atomic Box (AB on page 2-7), Atomic Subchart (AS on page 2-10), Box (B on page 2-13), Chart (C on page 2-15), Clipboard (CB on page 2-23), Data (D on page 2-23), Event (E on page 2-33), Editor (ED on page 2-30), Graphical Function (GF on page 2-35), Junction (J on page 2-38), Machine (M on page 2-40), MATLAB Function (MLF on page 2-30), Message (MSG on page 2-42), Root (R on page 2-3), State (S on page 2-53), Simulink Based State (SBS on page 2-46), Simulink Function (SLF on page 2-51), State Transition Table (STT on page 2-58), Transition (T on page 2-66), Truth Table (TT on page 2-72), and Truth Table Function (TTF on page 2-69).

Access

Properties

Property	Type	Access	Description	Objects
`Chart`	`Chart`	RO	Chart that contains this object.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 GF on page 2-35 J on page 2-38 MLF on page 2-30 S on page 2-53 SBS on page 2-46 SLF on page 2-51 T on page 2-66 TTF on page 2-69
`Editor`	`Editor`	RO	`Editor` object for this chart or state transition table.	C on page 2-15 STT on page 2-58

Property	Type	Access	Description	Objects
`Machine`	`Machine`	RO	Machine that contains this object.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 D on page 2-23 E on page 2-33 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF on page 2-69

Methods

Method	Description	Objects
`classhandle`	Return a read-only handle to the schema class of this object type.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 D on page 2-23 E on page 2-33 ED on page 2-30 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 R on page 2-3 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF

Method	Description	Objects
		on page 2-69
`defaultTransitions`	Return the default transitions at the top level of containment of this object.	B on page 2-13 C on page 2-15 GF on page 2-35 S on page 2-53

Method	Description	Objects
`disp`	Display all properties and values for this object.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 D on page 2-23 E on page 2-33 ED on page 2-30 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF on page 2-69

Method	Description	Objects
`find`	Find all objects inside this object that match the specified criteria.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 CB on page 2-23 D on page 2-23 E on page 2-33 ED on page 2-30 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 R on page 2-3 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF

Method	Description	Objects
		on page 2-69

Method	Description	Objects
get	Return the value of the specified property for this object.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 CB on page 2-23 D on page 2-23 E on page 2-33 ED on page 2-30 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 R on page 2-3 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF

Method	Description	Objects
		on page 2-69

Method	Description	Objects
`help`	Display all properties and descriptions for this object.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 CB on page 2-23 D on page 2-23 E on page 2-33 ED on page 2-30 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF on page 2-69

Method	Description	Objects
`innerTransitions`	Return an array of the transitions that originate in this object and terminate on a contained object.	B on page 2-13 S on page 2-53

Method	Description	Objects
`methods`	Display all methods for this object.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 CB on page 2-23 D on page 2-23 E on page 2-33 ED on page 2-30 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF on page 2-69

Method	Description	Objects
`outerTransitions`	Return an array of the transitions that exit the outer edge of this object and terminate on an object outside the containment of this object.	B on page 2-13 S on page 2-53

Method	Description	Objects
`set`	Set the value of the specified property for this object.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 CB on page 2-23 D on page 2-23 E on page 2-33 ED on page 2-30 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 R on page 2-3 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF

Method	Description	Objects
		on page 2-69
sinkedTransitions	Return all inner and outer transitions whose destination is this object.	B on page 2-13 J on page 2-38 S on page 2-53
sourcedTransitions	Return all inner and outer transitions whose source is this object.	B on page 2-13 J on page 2-38 S on page 2-53

Method	Description	Objects
`struct`	Return a MATLAB structure that contains all of the property values for this object.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 D on page 2-23 E on page 2-33 ED on page 2-30 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 R on page 2-3 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF on page 2-69

Method	Description	Objects
up	Return parent of this object.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 D on page 2-23 E on page 2-33 GF on page 2-35 J on page 2-38 MLF on page 2-30 MSG on page 2-42 S on page 2-53 SBS on page 2-46 SLF on page 2-51 T on page 2-66 TTF on page 2-69

Creation and Deletion

Method	Description	Objects
copy	Copy the specified objects to the clipboard.	CB on page 2-23

Method	Description	Objects
`delete`	Delete this object.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 D on page 2-23 E on page 2-33 GF on page 2-35 J on page 2-38 MLF on page 2-30 MSG on page 2-42 S on page 2-53 SBS on page 2-46 SLF on page 2-51 T on page 2-66 TTF on page 2-69
`pasteTo`	Paste the contents of the clipboard to the specified container object.	CB on page 2-23
`setImage`	Insert an image into an annotation.	A on page 2-4
`Stateflow.Annotation`	Create an annotation in a parent chart, state, box, or function.	N/A
`Stateflow.AtomicBox`	Create an atomic box in a parent chart, state, box, or function.	N/A
`Stateflow.AtomicSubchart`	Create an atomic subchart in a parent chart, state, or box.	N/A

Method	Description	Objects
`Stateflow.Box`	Create a box in a parent chart, state, box, or function.	N/A
`Stateflow.Data`	Create a data in a parent machine, chart, state, box, or function.	N/A
`Stateflow.EMFunction`	Create a MATLAB function in a parent chart, state, box, or function.	N/A
`Stateflow.Event`	Create an event in a parent chart, state, or box.	N/A
`Stateflow.Function`	Create a graphical function in a parent chart, state, box, or function.	N/A
`Stateflow.Junction`	Create a junction in a parent chart, state, box, or function.	N/A
`Stateflow.Message`	Create a message in a parent chart, state, or box.	N/A
`Stateflow.SimulinkBasedState`	Create a Simulink based state in a parent chart, state, or box.	N/A
`Stateflow.SLFunction`	Create a Simulink function in a parent chart, state, box, or function.	N/A
`Stateflow.State`	Create a state in a parent chart, state, or box.	N/A
`Stateflow.Transition`	Create a transition in a parent chart, state, box, or function.	N/A
`Stateflow.TruthTable`	Create a truth table function in a parent chart, state, box, or function.	N/A

Debugging

Properties

Property	Type	Access	Description	Objects
`Debug.Animation.Delay`	Double	RW	Delay value to slow down the animation for charts in this machine. Default value is `0`.	M on page 2-40
`Debug.Animation.Enabled`	Boolean	RW	Enable animation for charts in this machine. Default value is `true`.	M on page 2-40

Property	Type	Access	Description	Objects
`Debug.Breakpoints.EndBroadcast`	Boolean	RW	Set the `End of Broadcast` breakpoint of this event. Default value is `false`. This property applies only to local events.	E on page 2-33
`Debug.Breakpoints.OnDuring`	Boolean	RW	Set the `During State` breakpoint for this state or the `During Function Call` breakpoint for this function. Default value is `false`.	AS on page 2-10 GF on page 2-35 S on page 2-53 SBS on page 2-46 TTF on page 2-69
`Debug.Breakpoints.OnEntry`	Boolean	RW	Set the `On State Entry` breakpoint for this state or the `On Chart Entry` breakpoint for this chart. Default value is `false`.	AS on page 2-10 C on page 2-15 S on page 2-53 SBS on page 2-46 STT on page 2-58
`Debug.Breakpoints.OnExit`	Boolean	RW	Set the `On State Exit` breakpoint for this state. Default value is `false`.	AS on page 2-10 S on page 2-53 SBS on page 2-46
`Debug.Breakpoints.StartBroadcast`	Boolean	RW	Set the `Start of Broadcast` breakpoint of this event. Default value is `false`. This property applies only to local or input events.	E on page 2-33
`Debug.Breakpoints.WhenTested`	Boolean	RW	Set the `When Transition is Tested` breakpoint for this transition. Default value is `false`.	T on page 2-66

Property	Type	Access	Description	Objects
`Debug.Breakpoints.WhenValid`	Boolean	RW	Set the `When Transition is Valid` breakpoint for this transition. Default value is `false`.	T on page 2-66
`TestPoint`	Boolean	RW	Set this state or data object as a Stateflow test point. Default value is `false`. See “Monitor Test Points in Stateflow Charts”.	AS on page 2-10 D on page 2-23 S on page 2-53 SBS on page 2-46
`IsExplicitlyCommented`	Boolean	RW	Explicitly comment out this object. Default value is `false`. Equivalent to right-clicking the object and selecting **Comment Out**.	AB on page 2-7 AS on page 2-10 B on page 2-13 GF on page 2-35 J on page 2-38 MLF on page 2-30 S on page 2-53 SBS on page 2-46 SLF on page 2-51 T on page 2-66 TTF on page 2-69

Property	Type	Access	Description	Objects
`IsImplicitlyCommented`	Boolean	RO	Indicates if this object is implicitly commented out. An object is implicitly commented out when you comment out a superstate in its hierarchy.	AB on page 2-7 AS on page 2-10 B on page 2-13 GF on page 2-35 J on page 2-38 MLF on page 2-30 S on page 2-53 SBS on page 2-46 SLF on page 2-51 T on page 2-66 TTF on page 2-69

Methods

Method	Description	Objects
`isCommented`	Return a Boolean value that indicates if this object is explicitly or implicitly commented out.	AB on page 2-7 AS on page 2-10 B on page 2-13 GF on page 2-35 J on page 2-38 MLF on page 2-30 S on page 2-53 SBS on page 2-46 SLF on page 2-51 T on page 2-66 TTF on page 2-69
`parse`	Parse this chart or all the charts in this machine.	C on page 2-15 M on page 2-40 STT on page 2-58 TT on page 2-72

Display Control

Properties

Property	Type	Access	Description	Objects
`Subviewer`	`Chart`, `State`, or `Box`	RO	Chart or subchart where you can graphically view this object.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 GF on page 2-35 J on page 2-38 MLF on page 2-30 S on page 2-53 SBS on page 2-46 SLF on page 2-51 T on page 2-66 TTF on page 2-69
`Visible`	Boolean	RW	Indicates if the editor is displaying this chart or state transition table.	C on page 2-15 STT on page 2-58
`ZoomFactor`	Double	RW	Magnification level of this chart in the editor. A value of 1 corresponds to a magnification of 100%.	ED on page 2-30

Methods

Method	Description	Objects
`dialog`	Open the properties dialog box for this object.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 D on page 2-23 E on page 2-33 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF on page 2-69

Method	Description	Objects
`fitToView`	Zoom in on this object in the editor.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 GF on page 2-35 J on page 2-38 MLF on page 2-30 S on page 2-53 SBS on page 2-46 SLF on page 2-51 T on page 2-66 TTF on page 2-69

Method	Description	Objects
`highlight`	Highlight this object in the chart.	AB on page 2-7 AS on page 2-10 B on page 2-13 GF on page 2-35 J on page 2-38 MLF on page 2-30 S on page 2-53 SBS on page 2-46 SLF on page 2-51 T on page 2-66 TTF on page 2-69

Method	Description	Objects
`view`	Zoom in and select this object. If appropriate, display its contents in the Stateflow Editor, the Model Explorer, the MATLAB Editor.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 D on page 2-23 E on page 2-33 GF on page 2-35 J on page 2-38 MLF on page 2-30 MSG on page 2-42 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF on page 2-69
`zoomIn`	Zoom in on the Stateflow chart that contains this `Editor` object.	ED on page 2-30
`zoomOut`	Zoom out on the Stateflow chart that contains this `Editor` object.	ED on page 2-30

Graphical Appearance

Color Properties

Property	Type	Access	Description	Objects
`AutoBackgroundColor`	Boolean	RW	Use the automatic background color for this annotation. Default value is `true`.	A on page 2-4
`AutoForegroundColor`	Boolean	RW	Use the automatic foreground text color for this annotation. Default value is `true`.	A on page 2-4
`BackgroundColor`	Numeric vector	RW	Background color for this annotation. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[1 1 1]`.	A on page 2-4
`ChartColor`	Numeric vector	RW	Background color for this chart or for the automatically generated chart for this state transition table. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[1 0.9608 0.8824]`.	C on page 2-15 STT on page 2-58
`ErrorColor`	Numeric vector	RW	Color for errors in this chart or state transition table. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[1 0 0]`.	C on page 2-15 STT on page 2-58
`ForegroundColor`	Numeric vector	RW	Foreground color for this annotation. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0 0 0]`.	A on page 2-4

Property	Type	Access	Description	Objects
`JunctionColor`	Numeric vector	RW	Color for junctions in this chart or in the automatically generated chart for this state transition table. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0.6824 0.3294 0]`.	C on page 2-15 STT on page 2-58
`StateColor`	Numeric vector	RW	Color for state boxes in this chart or in the automatically generated chart for this state transition table. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0 0 0]`.	C on page 2-15 STT on page 2-58
`StateLabelColor`	Numeric vector	RW	Color for state labels in this chart or in the automatically generated chart for this state transition table. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0 0 0]`.	C on page 2-15 STT on page 2-58
`TransitionColor`	Numeric vector	RW	Color for transitions in this chart or in the automatically generated chart for this state transition table. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0.2902 0.3294 0.6039]`.	C on page 2-15 STT on page 2-58
`TransitionLabelColor`	Numeric vector	RW	Color for transition labels in this chart or in the automatically generated chart for this state transition table. Numeric vector `[r g b]` that specifies the red, green, and blue values of the color normalized to the range 0.0 to 1.0. Default value is `[0.2902 0.3294 0.6039]`.	C on page 2-15 STT on page 2-58

Drawing Properties

Property	Type	Access	Description	Objects
`ArrowSize`	Double	RW	For transitions, size of the transition arrow. Default value is `10`. For other objects, size of incoming transition arrows. Default value is `8`.	AB on page 2-7 AS on page 2-10 B on page 2-13 J on page 2-38 S on page 2-53 SBS on page 2-46 T on page 2-66
`DropShadow`	Boolean	RW	Display a drop shadow. Default value is `false`.	A on page 2-4

Font Properties

Property	Type	Access	Description	Objects
`Font.Angle`	Enum	RW	Font angle for the text in this annotation. Options are `'ITALIC'` or `'NORMAL'` (default).	A on page 2-4
`Font.Name`	Character vector	RO	Font for the text in this annotation. The `StateFont.Name` property of the chart that contains the annotation sets the value of this property.	A on page 2-4
`Font.Size`	Double	RW	Font size for the text in this annotation. The `StateFont.Size` property of the chart that contains the annotation sets the initial value of this property.	A on page 2-4
`Font.Weight`	Enum	RW	Font weight for the text in this annotation. Options are `'BOLD'` or `'NORMAL'` (default).	A on page 2-4

Property	Type	Access	Description	Objects
`FontSize`	Double	RW	Font size for the label of this object. The `StateFont.Size` property of the chart that contains the object sets the initial value of this property.	AB on page 2-7 AS on page 2-10 B on page 2-13 GF on page 2-35 MLF on page 2-30 S on page 2-53 SBS on page 2-46 SLF on page 2-51 T on page 2-66 TTF on page 2-69
`StateFont.Angle`	Enum	RW	Font angle for the state labels in this chart or in the automatically generated chart for this state transition table. Options are `'ITALIC'` or `'NORMAL'` (default).	C on page 2-15 STT on page 2-58
`StateFont.Name`	Character vector	RW	Font for the state labels in this chart or in the automatically generated chart for this state transition table. Default value is `'Helvetica'`.	C on page 2-15 STT on page 2-58
`StateFont.Size`	Integer	RW	Font size for the state labels in this chart or in the automatically generated chart for this state transition table. Default value is `12`.	C on page 2-15 STT on page 2-58
`StateFont.Weight`	Enum	RW	Font weight for the state labels in this chart or in the automatically generated chart for this state transition table. Options are `'BOLD'` or `'NORMAL'` (default).	C on page 2-15 STT on page 2-58

Property	Type	Access	Description	Objects
`TransitionFont.Angle`	Enum	RW	Font angle for transition labels in this chart or in the automatically generated chart for this state transition table. Options are `'ITALIC'` or `'NORMAL'` (default).	C on page 2-15 STT on page 2-58
`TransitionFont.Name`	Character vector	RW	Font for transition labels in this chart or in the automatically generated chart for this state transition table. Default value is `'Helvetica'`.	C on page 2-15 STT on page 2-58
`TransitionFont.Size`	Integer	RW	Font size for transition labels in this chart or in the automatically generated chart for this state transition table. Default value is `12`.	C on page 2-15 STT on page 2-58
`TransitionFont.Weight`	Enum	RW	Font weight for transition labels in this chart or in the automatically generated chart for this state transition table. Options are `'BOLD'` or `'NORMAL'` (default).	C on page 2-15 STT on page 2-58

Position Properties

Property	Type	Access	Description	Objects
`BadIntersection`	Boolean	RO	Indicates if this object graphically intersects a box, state, or function.	AB on page 2-7 AS on page 2-10 B on page 2-13 GF on page 2-35 MLF on page 2-30 S on page 2-53 SLF on page 2-51 SBS on page 2-46 TTF on page 2-69
`Destination`	`State`, `Box`, or `Junction`	RW	Destination state, box, or junction for this transition. Default value is `[]`.	T on page 2-66
`DestinationEndPoint`	Numeric vector	RW	Position of the transition endpoint at its destination. Numeric vector `[x y]` of coordinates relative to the upper left corner of the chart. Default value is `[40 40]`.	T on page 2-66
`DestinationOClock`	Double	RW	Location of the transition endpoint at its destination. Numeric value between 0 and 12 that specifies a clock position. Default value is `0`.	T on page 2-66
`LabelPosition`	Numeric vector	RW	Position and size of this label on this transition in the chart. Numeric vector `[left top width height]`. Default value is `[0 0 8 14]`.	T on page 2-66

Property	Type	Access	Description	Objects
`MidPoint`	Numeric vector	RW	Coordinates of the midpoint of the transition. Numeric vector `[x y]` of coordinates relative to the upper left corner of the chart. Default value is `[21 21]`.	T on page 2-66
`Position`	Numeric vector	RW	Position and size of this object in the chart. Numeric vector `[left top width height]`. Default value depends on the object type.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 GF on page 2-35 MLF on page 2-30 S on page 2-53 SBS on page 2-46 SLF on page 2-51 TTF on page 2-69
`Position.Center`	Numeric vector	RW	Position of the center of this junction. Numeric vector `[x y]` of coordinates relative to the upper left corner of the parent chart or state. Default value is `[7 7]`.	J on page 2-38
`Position.Radius`	Integer	RW	Radius of this junction. Default value is 7.	J on page 2-38
`Source`	`State`, `Box`, or `Junction`	RW	Source state, box, or junction of this transition. Default value is `[]`.	T on page 2-66

Property	Type	Access	Description	Objects
`SourceEndPoint`	Double	RW	Position of the transition endpoint at its source. Numeric vector `[x y]` of coordinates relative to the upper left corner of the chart. Default value is `[2 2]`.	T on page 2-66
`SourceOClock`	Double	RW	Location of the transition endpoint at its source. Numeric value between 0 and 12 that specifies a clock position. Default value is `0`.	T on page 2-66
`WindowPosition`	Numeric vector	RW	Position and size of the Stateflow editor window. Numeric vector `[left top width height]`.	ED on page 2-30

Text Properties

Property	Type	Access	Description	Objects
`Alignment`	Enum	RW	Alignment of the text in this annotation. Options are `'CENTER'`, `'LEFT'` (default), or `'RIGHT'`.	A on page 2-4
`FixedHeight`	Boolean	RW	Fix the height of the annotation box. Options are: • `true` — Fixes the height of the annotation box and hides content that is longer than the box. • `false` — Resizes the annotation box vertically as you add content (default).	A on page 2-4
`FixedWidth`	Boolean	RW	Fix the width of the annotation box. Options are: • `true` — Fixes the width of the annotation box and wraps text that is longer than the box. • `false` — Resizes the annotation box horizontally as you add content (default).	A on page 2-4

Property	Type	Access	Description	Objects
`InternalMargins`	Numeric vector	RW	Space from the bounding box of the text to the borders of this annotation. Numeric vector `[left top right bottom]`. Default value is `[0 0 0 0]`.	A on page 2-4
`Interpretation`	Enum	RW	Specify how to interpret the contents of the `Text` property in this annotation. Options are `'OFF'` (default), `'RICH'`, or `'TEX'`.	A on page 2-4
`LabelString`	Character vector	RW	Full label for this object. Default value depends on the object type.	AB on page 2-7 AS on page 2-10 B on page 2-13 GF on page 2-35 MLF on page 2-30 S on page 2-53 SBS on page 2-46 SLF on page 2-51 T on page 2-66 TTF on page 2-69
`PlainText`	Character vector	RO	Text for this annotation without formatting.	A on page 2-4
`Text`	Character vector	RW	Text for this annotation. Default value is `'?'`.	A on page 2-4

Identifiers

Property	Type	Access	Description	Objects
`Description`	Character vector	RW	Description of this object. Default value is `''`.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 D on page 2-23 E on page 2-33 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF on page 2-69

Property	Type	Access	Description	Objects
`Document`	Character vector	RW	Document link for this object. Default value is `''`.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 D on page 2-23 E on page 2-33 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF on page 2-69
`FullFileName`	Character vector	RO	Full path name of file that contains the Simulink model for this machine. Default value is `''`.	M on page 2-40

Property	Type	Access	Description	Objects
`Id`	Integer	RO	Unique identifier that distinguishes this object from other objects in the model.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 D on page 2-23 E on page 2-33 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF on page 2-69

Property	Type	Access	Description	Objects
`Name`	Character vector	RO for `Machine` objects RW for all other objects	Name of this object. Default value depends on the object type.	AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 D on page 2-23 E on page 2-33 GF on page 2-35 M on page 2-40 MLF on page 2-30 MSG on page 2-42 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 TT on page 2-72 TTF on page 2-69

Property	Type	Access	Description	Objects
`Path`	Character vector	RO	Location of the parent of this object in the model hierarchy.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 D on page 2-23 E on page 2-33 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF on page 2-69

Property	Type	Access	Description	Objects
Tag	Any Type	RW	Tag for this object. Holds data of any type. Default value is `[]`.	A on page 2-4 AB on page 2-7 AS on page 2-10 B on page 2-13 C on page 2-15 D on page 2-23 E on page 2-33 GF on page 2-35 J on page 2-38 M on page 2-40 MLF on page 2-30 MSG on page 2-42 S on page 2-53 SBS on page 2-46 SLF on page 2-51 STT on page 2-58 T on page 2-66 TT on page 2-72 TTF on page 2-69

Interface with Simulink

Property	Type	Access	Description	Objects
`ChartUpdate`	Enum	RW	Activation method for this chart. Options are `'CONTINUOUS'`, `'DISCRETE'`, or `'INHERITED'` (default). See “Update Method”.	C on page 2-15 STT on page 2-58 TT on page 2-72
`ClickFcn`	Character vector	RW	MATLAB code to be executed when a user single-clicks this note. Stateflow stores the code entered in this field with the chart. See “Associate a Click Function with an Annotation” (Simulink) for more information.	A on page 2-4
`DeleteFcn`	Character vector	RW	MATLAB code to be executed before deleting this note. See “Annotation Callback Functions” (Simulink).	A on page 2-4
`EmlDefaultFimath`	Enum	RW	Default `fimath` properties for this chart. Options are: • `'Same as MATLAB Default'` — Use the same `fimath` properties as the current default `fimath` (default). • `'Other:UserSpecified'` — Use the `InputFimath` property to specify the default `fimath` object. This property applies only to charts that use MATLAB as the action language.	C on page 2-15 STT on page 2-58 TT on page 2-72
`ExecuteAtInitialization`	Boolean	RW	Initialize the state configuration of this chart at time zero instead of at the first input event. Default value is `false`. See “Execution of a Chart at Initialization”.	C on page 2-15 STT on page 2-58

Property	Type	Access	Description	Objects
`ExportChartFunctions`	Boolean	RW	Export graphical functions at the chart level to other blocks in the Simulink model. Default value is `false`. See “Export Stateflow Functions for Reuse”.	C on page 2-15
`HasOutputData`	Boolean	RW	Create an active state data output port for this chart or state. Default value is `false`. See “Monitor State Activity Through Active State Data”.	AS on page 2-10 C on page 2-15 S on page 2-53 SBS on page 2-46 STT on page 2-58
`InitializeOutput`	Boolean	RW	Apply the initial value of the output data every time this chart wakes up. Default value is `false`. See “Initialize Outputs Every Time Chart Wakes Up”.	C on page 2-15 STT on page 2-58
`InputFimath`	Character vector	RW	Specify the `embedded.fimath` object associated with inputs from Simulink blocks. You can: • Enter an expression that constructs a `fimath` object. • Enter the variable name for a `fimath` object in the MATLAB or model workspace. This property applies only when the `EmlDefaultFimath` property for this chart is `'Other:UserSpecified'`.	C on page 2-15 STT on page 2-58 TT on page 2-72
`IsLink`	Boolean	RO	Indicates if this object is a library link.	AB on page 2-7 AS on page 2-10 SBS on page 2-46

Property	Type	Access	Description	Objects
`LoadFcn`	Character vector	RW	MATLAB code to be executed when the model containing this note is loaded. See "Annotation Callback Functions" (Simulink).	A on page 2-4
`OutputData`	`Data`	RO	Active state data object for this chart or state. This property applies only when the `HasOutputData` property for this object is `true`. See "Monitor State Activity Through Active State Data".	AS on page 2-10 C on page 2-15 S on page 2-53 SBS on page 2-46 STT on page 2-58
`OutputMonitoringMode`	Character vector	RW	Indicates the monitoring mode for the active state output data. • For charts and state transition tables, options are `'ChildActivity'` (default) or `'LeafStateActivity'`. • For states, options are `'ChildActivity'`, `'LeafStateActivity'`, or `'SelfActivity'` (default). • For atomic subcharts and Simulink based states, the only option is `'SelfActivity'` (read-only). This property applies only when the `HasOutputData` property for this object is `true`. See "Monitor State Activity Through Active State Data".	AS on page 2-10 C on page 2-15 S on page 2-53 SBS on page 2-46 STT on page 2-58
`Port`	Integer	RW	Port index for this data object, event, or message. This property applies only to input and output data, events, and messages.	D on page 2-23 E on page 2-33 MSG on page 2-42

Property	Type	Access	Description	Objects
`SampleTime`	Character vector	RW	Sample time for activating this chart. Default value is `'-1'`. This property applies only when the `ChartUpdate` property for this chart is `'DISCRETE'`.	C on page 2-15 STT on page 2-58 TT on page 2-72
`StatesWhenEnabling`	Enum	RW	Specify the behavior of states when a function-call input event reenables this chart. Options are: • `'held'` — The chart maintains the most recent values of the states (default). • `'reset'` — The chart reverts to the initial conditions of the states. This property applies only when the chart contains function-call input events. See “Control States in Charts Enabled by Function-Call Input Events”.	C on page 2-15 STT on page 2-58 TT on page 2-72
`StrongDataTypingWithSimulink`	Boolean	RW	Use strong data typing when this chart interfaces with Simulink input and output signals. Default value is `true`. This property applies only to state transition tables that use C as the action language. See “Use Strong Data Typing with Simulink I/O”.	C on page 2-15 STT on page 2-58

Property	Type	Access	Description	Objects
`TreatAsFi`	Enum	RW	Treat inherited fixed-point and integer signals as Fixed-Point Designer `fi` objects. Options are: • `'Fixed-point'` — The chart treats all fixed-point inputs as `fi` objects (default). • `'Fixed-point & Integer'` — The chart treats all fixed-point and integer inputs as `fi` objects. This property applies only to charts that use MATLAB as the action language.	C on page 2-15 STT on page 2-58 TT on page 2-72
`UseDisplayTextAsClickCallback`	Boolean	RW	Use the contents of the `Text` property as the click function for this annotation. Default value is `false`. See "Add an Annotation Callback" (Simulink).	A on page 2-4

Logging

Property	Type	Access	Description	Objects
`LoggingInfo.DataLogging`	Boolean	RW	Enable signal logging for this state or data object. Default value is `false`.	D on page 2-23 S on page 2-53 SBS on page 2-46
`LoggingInfo.DecimateData`	Boolean	RW	Limit the amount of data logged by skipping samples. Uses the interval specified by the `LoggingInfo.Decimation` property of this state or data object. Default value is `false`.	D on page 2-23 S on page 2-53 SBS on page 2-46

Property	Type	Access	Description	Objects
`LoggingInfo.Decimation`	Integer	RW	Decimation interval. Default value is 2. This value means the chart logs every other sample of this state or data object.	D on page 2-23 S on page 2-53 SBS on page 2-46
`LoggingInfo.LimitDataPoints`	Boolean	RW	Limit the number of data points to log. Uses the value specified by the `LoggingInfo.MaxPoints` property of this state or data object. Default value is `false`.	D on page 2-23 S on page 2-53 SBS on page 2-46
`LoggingInfo.MaxPoints`	Integer	RW	Maximum number of data points to log. Default value is `5000`. This value means the chart logs the last 5000 data points generated by the simulation for this state or data object.	D on page 2-23 S on page 2-53 SBS on page 2-46
`LoggingInfo.NameMode`	Enum	RW	Source of the signal name used to log this state or data object. Options are: • `'Custom'` — Use the custom signal name specified by the `LoggingInfo.LoggingName` property. • `'SignalName'` — Use the name of the state or data object (default).	D on page 2-23 S on page 2-53 SBS on page 2-46
`LoggingInfo.LoggingName`	Character vector	RW	Custom signal name used for logging this state or data object.	D on page 2-23 S on page 2-53 SBS on page 2-46

Specification

Chart and Machine

Property	Type	Access	Description	Objects
`ActionLanguage`	Enum	RW	Action language used to program this chart. Options are `'C'` or `'MATLAB'` (default). See “Differences Between MATLAB and C as Action Language Syntax”.	C on page 2-15 STT on page 2-58
`Created`	Character vector	RO	Date of creation of this machine.	M on page 2-40
`Creator`	Character vector	RW	Creator of this machine. Default value is `'Unknown'`.	M on page 2-40
`Dirty`	Boolean	RW	Indicates if this chart or the Simulink model for this machine has changed since being opened or saved.	C on page 2-15 M on page 2-40 STT on page 2-58 TT on page 2-72
`EnableBitOps`	Boolean	RW	Use C-bit operations in state and transition actions in this chart. Default value is `false`. This property applies only to charts that use C as the action language. See “Supported Operations for Chart Data”.	C on page 2-15 STT on page 2-58
`EnableNonTerminalStates`	Boolean	RW	Use super step semantics for this chart. Default value is `false`. See “Super Step Semantics”.	C on page 2-15 STT on page 2-58
`EnableZeroCrossings`	Boolean	RW	Use zero-crossing detection on state transitions in this chart. Default value is `true`. This property applies only when the `ChartUpdate` property for this chart is set to `'CONTINUOUS'`. See “Disable Zero-Crossing Detection”.	C on page 2-15 STT on page 2-58

Property	Type	Access	Description	Objects
`Iced`	Boolean	RO	Equivalent to property `Locked`. Used internally to prevent changes in this chart or machine during simulation.	C on page 2-15 M on page 2-40 STT on page 2-58 TT on page 2-72
`IsLibrary`	Boolean	RO	Indicates if the Simulink model for this machine builds a library and not an application. Default value is `false`.	M on page 2-40
`Locked`	Boolean	RW	Prevents changes in this chart or in the Simulink model for this machine. Default value is `false`.	C on page 2-15 M on page 2-40 STT on page 2-58 TT on page 2-72
`Modified`	Character vector	RW	Comment text for recording modifications to the Simulink model that defines this machine. Default value is `''`.	M on page 2-40
`NonTerminalMaxCounts`	Integer	RW	Maximum number of transitions this chart can take in one super step. Default value is `1000`. This property applies only when the `EnableNonTerminalStates` property for this chart is `true`. See “Super Step Semantics”.	C on page 2-15 STT on page 2-58

Property	Type	Access	Description	Objects
NonTerminalUnstableBehavior	Enum	RW	Behavior during simulation if this chart exceeds the maximum number of transitions specified in the NonTerminalMaxCounts property before reaching a stable state. Options are: • 'PROCEED' — The chart goes to sleep with the last active state configuration (default). • 'THROW ERROR' — The chart generates an error. This property applies only when the EnableNonTerminalStates property for this chart is true. See “Super Step Semantics”.	C on page 2-15 STT on page 2-58
StateMachineType	Enum	RW	Type of state machine semantics. Options are 'Classic' (default), 'Mealy', or 'Moore'. See “Overview of Mealy and Moore Machines”.	C on page 2-15 STT on page 2-58
Version	Character vector	RW	Comment text for recording the version of the Simulink model that defines this machine. Default value is 'none'.	M on page 2-40

Data, Events, and Messages

Property	Type	Access	Description	Objects
CompiledSize	Character vector	RO	Size of this data object as determined by the compiler.	D on page 2-23 MSG on page 2-42
CompiledType	Character vector	RO	Type of this data object as determined by the compiler.	D on page 2-23 MSG on page 2-42

Property	Type	Access	Description	Objects
`DataType`	Enum	RW	Type of this data object or message data. • If the `Props.Type.Method` property of this data object is `'Built-in'`, you can specify this property with one of these options: `'double'` (default), `'single'`, `'int8'`, `'int16'`, `'int32'`, `'int64'` (C charts only), `'uint8'`, `'uint16'`, `'uint32'`, `'uint64'` (C charts only), `'boolean'`, `'ml'`, or `'string'` (C charts only). • Otherwise, the `Props.Type` properties of this data object determine the value of this property.	D on page 2-23 MSG on page 2-42
`InitializeMethod`	Enum	RW	Method for initializing the value of this data object. Options depend on the scope of the data: • For local and output data, use `'Expression'` (default) or `'Parameter'`. • For constant data, use `'Expression'` (read-only). • For data store memory, input, and parameter data, use `'Not Needed'` (read-only).	D on page 2-23

Property	Type	Access	Description	Objects
`MessagePriorityOrder`	Enum	RW	Type of priority queue for this message. Options are: • `'Ascending'` — Messages are received in ascending order of the message data value (default). • `'Descending'` — Messages are received in descending order of the message data value. This property applies only when the `QueueType` property of this message is `Priority`.	MSG on page 2-42
`Port`	Integer	RW	Port index for this data object, event, or message. This property applies only to input and output data, events, and messages.	D on page 2-23 E on page 2-33 MSG on page 2-42
`Props.Array.FirstIndex`	Character vector	RW	Index for the first element of this data object or message data. Default value is `0`. This property applies only to array data in charts that use C as the action language.	D on page 2-23 MSG on page 2-42
`Props.Array.IsDynamic`	Boolean	RW	Allow the size of this data object to change at run time. Default value is `false`. Equivalent to the **Variable Size** check box in the Data properties dialog box. See “Declare Variable-Size Data in Stateflow Charts”.	D on page 2-23
`Props.Array.Size`	Character vector	RW	Size of this data object or message data. Default value is `'0'`. See “Specify Size of Stateflow Data”.	D on page 2-23 MSG on page 2-42
`Props.Complexity`	Enum	RW	Enable this data object or message data to take complex values. Options are `'On'` or `'Off'` (default). See “Complex Data in Stateflow Charts”.	D on page 2-23 MSG on page 2-42

Property	Type	Access	Description	Objects
`Props.Frame`	Enum	RW	Enable this data object or message data to accept frame-based signals. Options are: • `'Frame based'` — The data object supports frame-based signals. • `'Sample based'` — The data object supports sample-based signals (default).	D on page 2-23 MSG on page 2-42
`Props.InitialValue`	Character vector	RW	Initial value of this data object or message data. Default value is `''`.	D on page 2-23 MSG on page 2-42
`Props.Range.Maximum`	Character vector	RW	Maximum value for this data object. Default value is `''`.	D on page 2-23
`Props.Range.Minimum`	Character vector	RW	Minimum value for this data object. Default value is `''`.	D on page 2-23
`Props.ResolveToSignalObject`	Boolean	RW	Specify if this data object resolves to a `Simulink.Signal` object that you define in the model or base workspace. Default value is `false`. See “Resolve Data Properties from Simulink Signal Objects”.	D on page 2-23
`Props.Type.BusObject`	Character vector	RW	Name of the `Simulink.Bus` object that defines this data object or message data. This property applies only when the `Props.Type.Method` property of this data object is `'Bus Object'`. See “Access Bus Signals Through Stateflow Structures”.	D on page 2-23 MSG on page 2-42
`Props.Type.EnumType`	Character vector	RW	Name of the enumerated type that defines this data object or message data. This property applies only when the `Props.Type.Method` property of this data object is `'Enumerated'`. See “Reference Values by Name by Using Enumerated Data”.	D on page 2-23 MSG on page 2-42

Property	Type	Access	Description	Objects
`Props.Type.Expression`	Character vector	RW	Expression that evaluates to a data type for this data object or message data. This property applies only when the `Props.Type.Method` property of this data object is `'Expression'`. See “Specify Data Properties by Using MATLAB Expressions”.	D on page 2-23 MSG on page 2-42
`Props.Type.Fixpt.Bias`	Character vector	RW	Bias for this data object. This property applies only to fixed-point data with the `Props.Type.Fixpt.ScalingMode` property set to `'Slope and bias'`. See “Fixed-Point Data in Stateflow Charts”.	D on page 2-23
`Props.Type.Fixpt.FractionLength`	Character vector	RW	Location of the binary point for this data object. This property applies only to fixed-point data when the `Props.Type.Fixpt.ScalingMode` property is `'Binary point'`. See “Fixed-Point Data in Stateflow Charts”.	D on page 2-23
`Props.Type.Fixpt.Lock`	Boolean	RW	Prevents replacement of the fixed-point type of this data object with an autoscaled type chosen by the Fixed-Point Tool. Default value is `false`. See “Autoscaling Using the Fixed-Point Tool” (Fixed-Point Designer).	D on page 2-23
`Props.Type.Fixpt.ScalingMode`	Enum	RW	Method for scaling this data object. Options are `'Binary point'`, `'Slope and bias'`, or `'None'` (default). This property applies to fixed-point data. See “Fixed-Point Data in Stateflow Charts”.	D on page 2-23
`Props.Type.Fixpt.Slope`	Character vector	RW	Slope for this data object. This property applies only to fixed-point data when the `Props.Type.Fixpt.ScalingMode` property set to `'Slope and bias'`. See “Fixed-Point Data in Stateflow Charts”.	D on page 2-23

Property	Type	Access	Description	Objects

`Props.Type.Method`	Enum	RW	Method for setting the type of this data object or message data.
			• For data objects, options depend on the scope:
			  • For local, input, output, or parameter data, use `'Built-in'`, `'Fixed point'`, `'Enumerated'`, `'Expression'`, `'Inherited'` (default), or `'Bus Object'`.
			  • For constant data, use `'Built-in'` (default), `'Fixed point'`, or `'Expression'`.
			  • For data store memory data, use `'Inherited'` (read-only).
			• For message data, options are `'Inherited'`, `'Built-in'`, `'Enumerated'`, `'Expression'`, or `'Bus Object'`.
			Equivalent to the **Mode** field of the Data Type Assistant in the Data properties dialog box. See “Specify Type of Stateflow Data”.
				D on page 2-23 MSG on page 2-42

`Props.Type.Signed`	Boolean	RW	Specify if this data object is signed. Default value is `true`. This property applies only to fixed-point data. See “Fixed-Point Data in Stateflow Charts”.	D on page 2-23
`Props.Type.WordLength`	Character vector	RW	Bit size of the word that holds the quantized integer of this data object. This property applies only to fixed-point data. See “Fixed-Point Data in Stateflow Charts”.	D on page 2-23
`Props.Unit.Name`	Character vector	RW	Units of measurement for this data object. Default value is `''`. See “Specify Units for Stateflow Data”.	D on page 2-23

Property	Type	Access	Description	Objects
`QueueCapacity`	Integer	RW	Length of the internal queue for this message. Default value is `10`. This property applies only to input and local messages. For more information, see “Message Queue Properties”.	MSG on page 2-42
`QueueOverflowDiagnostic`	Enum	RW	Level of diagnostic action when the number of incoming messages exceeds the queue capacity for this message. Options are `'Error'` (default), `'Warning'`, or `'None'`. This property applies only to input and local messages. For more information, see “Message Queue Properties”.	MSG on page 2-42
`QueueType`	Enum	RW	Indicates the order in which messages are removed from the receiving queue. Options are: • `'FIFO'` — First in, first out (default). • `'LIFO'` — Last in, first out. • `'Priority'` — Remove messages according to the value in the data field. To specify the order, use the `MessagePriorityOrder` property for the message. This property applies only to input and local messages. For more information, see “Message Queue Properties”.	MSG on page 2-42
`SaturateOnIntegerOverflow`	Boolean	RW	Specify the behavior of integer overflows in this chart. Options are: • `true` — The chart saturates integer overflows (default). • `false` — The chart wraps integer overflows. For more information, see “Handle Integer Overflow for Chart Data”.	C on page 2-15 STT on page 2-58 TT on page 2-72

Property	Type	Access	Description	Objects
`SaveToWorkspace`	Boolean	RW	Assign the value of this data object to a variable of the same name in the MATLAB base workspace at the end of the simulation. Default value is `false`. See “Save Final Value to Base Workspace”.	D on page 2-23
`Scope`	Enum	RW	Scope of this data object, event, or message. • For data objects, options are `'Local'` (default), `'Constant'`, `'Parameter'`, `'Input'`, `'Output'`, `'Data Store Memory'`, `'Temporary'`, `'Imported'`, or `'Exported'`. • For events, options are `'Input'`, `'Local'` (default), or `'Output'`. • For messages, options are `'Input'`, `'Local'`, or `'Output'` (default).	D on page 2-23 E on page 2-33 MSG on page 2-42
`SupportVariableSizing`	Boolean	RW	Support input and output data that vary in dimension when simulating this chart. Default value is `true`. See “Declare Variable-Size Data in Stateflow Charts”.	C on page 2-15 STT on page 2-58 TT on page 2-72
`Trigger`	Enum	RW	Type of trigger associated with this event. Options depend on the scope of the event: • For input events, use `'Rising'`, `'Falling'`, `'Either'`, or `'Function call'` (default). • For output events, use `'Either'` or `'Function call'` (default). This property does not apply to local events.	E on page 2-33

Property	Type	Access	Description	Objects
Tunable	Boolean	RW	Indicates that the value of this data object can be modified during simulation. Default is true. This property applies only to parameter data.	D on page 2-23
UseInternalQueue	Boolean	RW	Indicates that the Stateflow chart maintains an internal receiving queue for this input message. Default value is true. This property applies only to input messages. For more information, see "Message Queue Properties".	MSG on page 2-42

Graphical Objects

Property	Type	Access	Description	Objects
ActionTable	Cell array of character vectors	RW	Action table for this truth table. Default value is [].	TT on page 2-72 TTF on page 2-69
ConditionTable	Cell array of character vectors	RW	Condition table for this truth table. Default value is [].	TT on page 2-72 TTF on page 2-69
ContentPreviewEnabled	Boolean	RW	Display a preview of the contents of this Simulink based state at the Stateflow level. Default value is true.	SBS on page 2-46
Decomposition	Enum	RW	State decomposition at the top level of containment in this object. Options are 'EXCLUSIVE_OR' (default) and 'PARALLEL_AND'.	C on page 2-15 S on page 2-53
ExecutionOrder	Integer	RW	Order in which this state wakes up in parallel (AND) decomposition or this transition executes when its source is active. This property applies only when the UserSpecifiedStateTransitionExecutionOrder property of the chart that contains the object is true.	AS on page 2-10 S on page 2-53 SBS on page 2-46 T on page 2-66

Property	Type	Access	Description	Objects
`InlineOption`	Character vector	RW	Specify how this function or state appears in generated code. Options are: • `'Inline'` — Calls to function are replaced by code. • `'Function'` — function is implemented as a separate C function. • `'Auto'` — An internal calculation determines the appearance of function calls in generated code (default). For more information, see “Inline State Functions in Generated Code” (Simulink Coder).	GF on page 2-35 MLF on page 2-30 S on page 2-53 TTF on page 2-69
`IsGrouped`	Boolean	RW	Specify if this object is a group. Default value is `false`. See “Copy by Grouping” on page 1-31.	B on page 2-13 GF on page 2-35 S on page 2-53
`IsSubchart`	Boolean	RW	Specify if this object is a subchart. Default value is `false`.	B on page 2-13 GF on page 2-35 S on page 2-53
`Language`	Enum	RW	Action language used to program the truth table function. Options are `C` or `MATLAB` (default). See “Differences Between MATLAB and C as Action Language Syntax”.	TTF on page 2-69
`OverSpecDiagnostic`	Enum	RW	Level of diagnostic action when this truth table is overspecified. Options are `'Error'` (default), `'Warning'`, or `'None'`. See “Correct Overspecified and Underspecified Truth Tables”.	TT on page 2-72 TTF on page 2-69

Property	Type	Access	Description	Objects

`Script`	Character vector	RW	Code for this MATLAB function. To specify the value of this property, create a character vector by calling the `sprintf` function. For example, if `f` is a handle to this function, enter:
			`str = sprintf('function y=f(x) \n y=x+1;');`
			`f.Script = str;`
				MLF on page 2-30

`Type`	Enum	RW for junctions RO for states	Type for this junction or state. • For junctions, options are `'CONNECTIVE'` (default) or `'HISTORY'`. • For states, options are `'AND'` (parallel) or `'OR'` (exclusive). The state inherits this property from the `Decomposition` property of its parent.	J on page 2-38 S on page 2-53 SBS on page 2-46
`UnderSpecDiagnostic`	Enum	RW	Level of diagnostic action when this truth table is underspecified. Options are `'Error'` (default), `'Warning'`, or `'None'`. See “Correct Overspecified and Underspecified Truth Tables”.	TT on page 2-72 TTF on page 2-69

4

API Method Reference

classhandle

Provide handle to schema class of object type

Syntax

```
handle = thisObject.classhandle
```

Description

The `classhandle` method returns a read-only handle to the schema class of this object type. You can use the `classhandle` method to provide information about the structure of each object type.

Arguments

`thisObject`	The object for which to return a handle. Can be any Stateflow object.

Returns

`handle`	Handle to schema class of this object.

Examples

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'})
```

Introduced before R2006a

copy

Copy specified array of objects to clipboard

Syntax

```
cbObj.copy(objArray)
```

Description

The `copy` method copies the specified objects to the clipboard. Objects to copy are specified through a single argument array of objects.

Later, complete the copy operation by invoking the `pasteTo` method.

Arguments

`cbObj`	The Clipboard object to copy to.
`objArray`	Array of Stateflow objects to copy. These objects must conform to the following: • The objects copied must be all graphical (states, boxes, functions, transitions, junctions) or all nongraphical (data, events). • If all objects are graphical, they must all be seen in the same subviewer.

Returns

None

Examples

See “Copy and Paste Stateflow Objects” on page 1-30.

Introduced before R2006a

defaultTransitions

Return default transitions in object at top level of containment

Syntax

```
defaultTransitions = thisObject.defaultTransitions
```

Description

The `defaultTransitions` method returns the default transitions in this object at the top level of containment.

Arguments

`thisObject`	The object for which to return default transitions. Can be an object of type Chart, State, Box, or Function.

Returns

`defaultTransitions`	Array of default transitions in this object at the top level of containment.

Examples

If state `A` contains state `A1`, and state `A1` contains state `A11`, and states `A1` and `A11` have default transitions attached to them, the `defaultTransitions` method of state `A` returns the default transition attached to state `A1`.

Introduced before R2006a

delete

Delete object

Syntax

```
thisObject.delete
```

Description

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

Arguments

`thisObject`	The object to delete. Can be an object of type State, Box, Function, Truth Table, Annotation, Transition, Junction, Data, Event.

Returns

None

Examples

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

Introduced before R2006a

dialog

Open properties dialog box of object

Syntax

```
thisObject.dialog
```

Description

The `dialog` method opens the Properties dialog box of its object.

Arguments

`thisObject`	The object for which to open the Properties dialog box.

Returns

None

Examples

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

Introduced before R2006a

disp

Display properties and settings for object

Syntax

```
thisObject.disp
```

Description

The `disp` method displays the properties and settings for this object. This is true for all but objects of type Root and Clipboard.

Arguments

`thisObject`	The object for which to display properties and settings.

Returns

None

Examples

If a state `A` is represented by the State object `sA`, the command `sA.disp` displays the property names and their settings for state `A`.

Introduced before R2006a

find

Return specified objects

Syntax

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

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`, and so on) 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 by using the `-depth` specifier. However, the zeroth level of containment, that is, the searched object itself, is always included if it satisfies the search criteria.

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

Arguments

`thisObject`	The object for which to find contained objects. Can be an object of type Root, Machine, State, Box, Function, or Truth Table.
`'-and'`	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. **Note** If no logical operator is specified, `-and` is assumed.

`'-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 name (e.g., `'Stateflow.State'`, `'Stateflow.Transition'`, etc.) or as a handle to the class. For more information, see `classhandle`.
`'-depth'`	Integer depth to search, which can be 0, 1, 2,..., infinite. The default search depth is infinite.

`'-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 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 name (e.g., `'Stateflow.State'`, `'Stateflow.Transition'`, etc.) or as a handle to the object type. For more information, see `classhandle`.
`'-method'`	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.
`'`*`property`*`'`	The specifier takes on the name of the property. Value of the specified property for the objects you want to find.
`'-property'`	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.

`'-regexp'`	No value follows this specifier. Instead, this specifier indicates that the value of the following specifier-value pair contains a regular expression. For more information, see `regexp`.

Returns

`objArray`	Array of objects found matching the criteria specified.

Examples

If a Chart object `c` represents a Stateflow chart, to produce an array of all the states in the chart, use this command :

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

To produce an array of all objects whose `Name` property is `'A'`, use this command:

```
objects = c.find('Name','A')
```

To produce an array of all states whose `Name` property starts with the letter `A`, use this command:

```
states = c.find('-isa','Stateflow.State','-and','-regexp','Name','^A')
```

See Also

`classhandle` | `regexp`

Introduced before R2006a

fitToView

Zoom in on graphical Stateflow object

Syntax

```
thisObject.fitToView
```

Description

The `fitToView` method zooms in on this Stateflow object and highlights it in the editor.

Arguments

`thisObject`	The object on which to zoom.

Returns

None

Examples

If `myState` is a State object, the command `myState.fitToView` zooms in on that state and highlights it in the editor.

See Also

`view` | `zoomIn and zoomOut`

Introduced in R2008a

get

Return MATLAB structure containing property settings of object or 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

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

Returns

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

Examples

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
```

Introduced before R2006a

help

Display list of properties for object with accompanying descriptions

Syntax

```
thisObject.help
```

Description

The `help` 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

Examples

If `j` is an API handle to a Stateflow junction, the command `j.help` returns a list of the property names and descriptions for a Stateflow API object of type Junction.

Introduced before R2006a

highlight

Highlight graphical object in chart

Syntax

```
thisObject.highlight
```

Description

This method highlights one of the following objects in a chart:

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

Arguments

`thisObject`	The object you want to highlight.

Returns

None

Examples

The following example shows how to highlight a state in a chart.

```
sf_car;
rt = sfroot;
ss_state = rt.find('-isa','Stateflow.State','Name','steady_state');
ss_state.highlight;
```

See Also

`view` | `zoomIn and zoomOut`

Introduced in R2012a

innerTransitions

Return inner transitions that originate with chart or state and terminate on 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.

Examples

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.

Introduced before R2006a

isCommented

Determine if object is commented out

Syntax

```
isCommented(thisObject)
```

Description

Returns a Boolean indicating if `thisObject` is explicitly or implicitly commented out.

Arguments

`thisObject`	The object which you determine if it is commented out.

Returns

If the object is explicitly or implicitly commented out, returns the Boolean value `true`. Otherwise, returns `false`.

Introduced in R2016a

methods

List methods belonging to 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 do not apply to chart use, and are not documented. Unsupported methods include: `areChildrenOrdered`, `evalDialogParams`, `getChildren`, `getCurrentDialogPrompts`, `getDialogInterface`, `getDialogProxy`, `getDialogSchema`, `getDisplayClass`, `getDisplayIcon`, `getDisplayLabel`, `getFullName`, `getHierarchicalChildren`, `getInstanceProperties`, `getParent`, `getPreferredProperties`, `isHierarchical`, `isLibrary`, `isLinked`, `isMasked`, `isModelReference`, `isTunableProperty`, `isValidProperty`.

Arguments

`thisObject`	Object for which to list methods. Can be of any Stateflow object type.

Returns

None

Examples

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

Introduced before R2006a

outerTransitions

Return array of outer transitions for 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

`thisObject`	The object for which to find outer transitions. Can be of object type State or Box.

Returns

`transitions`	An array of transitions exiting the outer edge of this state.

Examples

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`.

Introduced before R2006a

parse

Parse single chart or all charts in model

Syntax

```
thisChart.parse
```

```
thisMachine.parse
```

Description

For Chart objects, the `parse` method parses this chart.

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

Arguments

`thisChart`	The chart to parse.
`thisMachine`	The machine containing charts to parse.

Returns

None

Examples

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

Introduced before R2006a

pasteTo

Paste objects in clipboard to specified container object

Syntax

```
clipboard.pasteTo(newContainer)
```

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

`newContainer`	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, annotations, transitions, junctions), this object must be a chart or subchart.

Returns

None

Examples

See the section “Copy and Paste Stateflow Objects” on page 1-30.

Introduced before R2006a

set

Set properties with 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

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

Returns

None

Examples

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])
```

Introduced before R2006a

setImage

Insert image from clipboard or image file into an annotation

Syntax

```
thisAnnotation.setImage(path)

thisAnnotation.setImage('clipboard')

thisAnnotation.setImage('')
```

Description

`thisAnnotation.setImage(path)` inserts a image from the file specified with the `path` argument.

`thisAnnotation.setImage('clipboard')` inserts an image from the clipboard.

`thisAnnotation.setImage('')` sets the annotation to be a text annotation.

Arguments

`path`	Specifies the full path to the image file.
`'clipboard'`	Specify inserting an image from the clipboard.
`''`	An empty value that sets the annotation to be a text annotation.

Returns

None

Examples

If annotation `A` is represented by Annotation object `sA`, the MATLAB command statement `sA.setImage('myfolder/annotation_images/converter.png')` inserts the `converter.png` image in annotation A.

See Also

`Stateflow.Annotation`

Introduced in R2014a

sinkedTransitions

Return transitions that have object as destination

Syntax

```
transitions = thisObject.sinkedTransitions
```

Description

The `sinkedTransitions` method returns all inner and outer transitions that have this object as their destination.

Arguments

`thisObject`	Destination object of the returned transitions. Can be of type State, Box, or Junction.

Returns

`transitions`	Array of all transitions whose destination is this object.

Examples

The following example shows how to find all transitions whose destination is the state named `steady_state`.

```
sf_car;
rt = sfroot;
ss_state = rt.find('-isa','Stateflow.State','Name','steady_state');
sinked_trans = ss_state.sinkedTransitions;
```

Introduced in R2012a

sourcedTransitions

Return transitions that have object as source

Syntax

```
transitions = thisObject.sourcedTransitions
```

Description

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

Arguments

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

Returns

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

Examples

The following example shows how to find all transitions whose source is the state named `steady_state`.

```
sf_car;
rt = sfroot;
ss_state = rt.find('-isa','Stateflow.State','Name','steady_state');
sourced_trans = ss_state.sourcedTransitions;
```

Introduced before R2006a

Stateflow.Annotation

Create annotation

Syntax

```
annotation_new = Stateflow.Annotation(parent)
```

Description

The `Stateflow.Annotation` method is a constructor method for creating an annotation in a parent chart, state, box, or graphical function. This method returns a handle to the new `Annotation` object.

Arguments

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

Returns

`annotation_new`	Handle to the Annotation object for the newly created annotation

Examples

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

```
annotation_new = Stateflow.Annotation(sA)
```

The new annotation appears in the upper left corner of state `A` in the chart, but is invisible because it has no text content. `annotation_new` is a handle to the Annotation

object for the new annotation, that you use to set its text content with a command like the following:

```
annotation_new.Text = 'This is an annotation'
```

Introduced before R2006a

Stateflow.AtomicBox

Create atomic box

Syntax

```
atomic_box_new = Stateflow.AtomicBox(parent)
```

Description

The `Stateflow.AtomicBox` method is a constructor method for creating an atomic box in a parent chart, state, box, or graphical function. This method returns a handle to the new `AtomicBox` object.

Arguments

`parent`	Handle to the object for the parent chart, state, box, or function that contains the new atomic box

Returns

`atomic_box_new`	Handle to Atomic Box object for newly created atomic box

Examples

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

```
atomic_box_new = Stateflow.AtomicBox(sA)
```

The new atomic box appears in the upper left corner of state `A` in the chart. `atomic_box_new` is a handle to the new Atomic Box object that you can use to rename the atomic box, set its properties, and execute its methods.

Introduced in R2012b

Stateflow.AtomicSubchart

Create atomic subchart

Syntax

```
atomic_subchart_new = Stateflow.AtomicSubchart(parent)
```

Description

The `Stateflow.AtomicSubchart` method is a constructor method for creating an atomic subchart in a parent chart, state, or box. This method returns a handle to the new `AtomicSubchart` object. For more information on atomic subcharts, see “Create Reusable Subcomponents by Using Atomic Subcharts”.

Arguments

`parent`	Handle to the object for the parent chart, state, or box that contains the new atomic subchart

Returns

`atomic_subchart_new`	Handle to Atomic Subchart object for newly created atomic subchart

Examples

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

```
atomic_subchart_new = Stateflow.AtomicSubchart(sA)
```

The new atomic subchart appears in the upper left corner of state `A` in the chart. `atomic_subchart_new` is a handle to the new Atomic Subchart object that you can use to rename the atomic subchart, set its properties, and execute its methods.

Introduced in R2010b

Stateflow.Box

Create box

Syntax

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

Description

The `Stateflow.Box` method is a constructor method for creating a box in a parent chart, state, box, or graphical function. This method returns a handle to the new `Box` object.

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

Examples

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 corner inside state `A`. `box_new` is a handle to a Box object for the new box.

Introduced before R2006a

Stateflow.Data

Create data

Syntax

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

Description

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

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

Examples

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.

Introduced before R2006a

Stateflow.EMFunction

Create MATLAB function

Syntax

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

Description

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

Arguments

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

Returns

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

Examples

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

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

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

Introduced before R2006a

Stateflow.Event

Create event

Syntax

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

Description

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

Arguments

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

Returns

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

Examples

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.

Introduced before R2006a

Stateflow.Function

Create graphical function

Syntax

```
function_new = Stateflow.Function(parent)
```

Description

The `Stateflow.Function` method is a constructor method for creating a graphical function in a parent chart, state, box, or graphical function. This method returns a handle to the new `Function` object.

Arguments

`parent`	Handle to parent chart, state, box, or function of the new graphical function

Returns

`function_new`	Handle to a Function object for the new graphical function

Examples

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

```
function_new = Stateflow.Function(sA)
```

The new graphical function is unnamed and appears in the upper left corner inside state `A` in the chart. `function_new` is a handle to the Function object for the new graphical function that you use to rename the function, set its properties, and execute its methods.

Stateflow.Junction

Create 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 graphical function. This method returns a handle to the new `Junction` object.

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

Examples

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 chart. `junc_new` is a handle to the Junction object for the new junction that you use to set its properties, and execute its methods.

Introduced before R2006a

Stateflow.Message

Create message

Syntax

```
message_new = Stateflow.Message(parent)
```

Description

The `Stateflow.Message` method is a constructor method for creating a message in a parent chart, state, or box. This method returns a handle to the new `Message` object.

Arguments

`parent`	Handle to parent chart, state, or box of new message

Returns

`message_new`	Handle to the Message object for the new message

Examples

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

```
message_new = Stateflow.Message(sA)
```

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

Introduced in R2015b

Stateflow.SimulinkBasedState

Create Simulink based state

Syntax

```
Simulink_based_state_new = Stateflow.SimulinkBasedState(parent)
```

Description

The `Stateflow.SimulinkBasedState` method is a constructor method for creating a Simulink based state in a parent chart, state, or box. This method returns a handle to the new `SimulinkBasedState` object. For more information on Simulink based state, see “Simulink Subsystems as States”.

Arguments

`parent`	Handle to the object for the parent chart, state, or box that contains the new atomic subchart

Returns

`Simulink_based_state_new`	Handle to Simulink based state object for newly created Simulink based state

Examples

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

```
Simulink_based_state_new = Stateflow.SimulinkBasedState(sA)
```

The new Simulink based state appears in the upper left corner of state A in the chart. `Simulink_based_state_new` is a handle to the new Simulink based state object that you can use to rename the Simulink based state, set its properties, and execute its methods.

Introduced in R2017b

Stateflow.SLFunction

Create Simulink function

Syntax

```
Simulink_function_new = Stateflow.SLFunction(parent)
```

Description

The `Stateflow.SLFunction` method is a constructor method for creating a Simulink function in a parent chart, state, box, or graphical function. This method returns a handle to the new `SLFunction` object.

Arguments

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

Returns

`sl_function`	Handle to the newly created Simulink Function object

Examples

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

```
sl_function = Stateflow.SLFunction(sA)
```

The new Simulink function appears in the upper left corner of state `A` in the chart. `sl_function` is a handle to the new Simulink function that you can use to rename the function, set its properties, and execute its methods.

Introduced in R2008b

Stateflow.State

Create state

Syntax

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

Description

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

Arguments

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

Returns

`state_new`	Handle to State object for newly created state

Examples

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 corner of state `A` in the chart. `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.

Introduced before R2006a

Stateflow.Transition

Create transition

Syntax

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

Description

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

Arguments

`parent`	Handle to parent chart, state, box, or function of new transition

Returns

`transition_new`	Handle to Transition object for the new transition

Examples

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. `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.

Introduced before R2006a

Stateflow.TruthTable

Create truth table function

Syntax

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

Description

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

Arguments

`parent`	Handle to parent chart, state, box, or function of new truth table

Returns

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

Examples

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 chart. `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.

Introduced before R2006a

struct

Return MATLAB structure containing property settings of 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 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

Examples

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`.

Introduced before R2006a

up

Return parent of object

Syntax

```
parentObject = thisObject.up
```

Description

The up method returns a handle to the parent of this object.

Arguments

thisObject	Object for which to return parent (containing) object

Returns

parentObject	Object containing thisObject

Examples

Assume that a chart has two states, A and B, and state A contains state B. If the object sB represents the state B, then the command

```
p = sB.up
```

returns a handle p to the parent of B, which is state A.

Introduced before R2006a

view

Make object visible for editing

Syntax

```
thisObject.view
```

Description

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

- For Chart objects, the `view` method opens the chart, if it is not already open, and brings it to the foreground.
- For State, Box, Function, Annotation, Junction, and Transition objects, the `view` method does the following:
  - **a** Opens the chart containing the object if it is not already open.
  - **b** Highlights the object.
  - **c** Zooms the object's editor window to the level of full expanse of the object's containing state or chart.
  - **d** Brings the editor window for this object to the foreground.
- For Atomic Subchart and Atomic Box objects, the `view` method shows the contents of the object.
- For Truth Table objects, the `view` method opens the Truth Table Editor for this truth table.
- For MATLAB Function objects, the `view` method opens the editor for this function.
- For Simulink Function objects, the `view` method shows the contents of the function-call subsystem.
- For Event and Data objects, the `view` method opens the Model Explorer.

Arguments

`thisObject`	Object for which to display editing environment.

Returns

None

Introduced before R2006a

zoomIn and zoomOut

Zoom in or out on Stateflow chart

Syntax

```
thisEditor.zoomIn

thisEditor.zoomOut
```

Description

The methods `zoomIn` and `zoomOut` cause the editor for a 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 editor for the chart.

Arguments

`thisEditor`	Editor object on which to zoom in or out.

Returns

None

Examples

If the Editor object `ed` represents the editor for a chart with the zoom level at 100%, the command `ed.zoomIn` raises the zoom level to 120%.

Introduced before R2006a