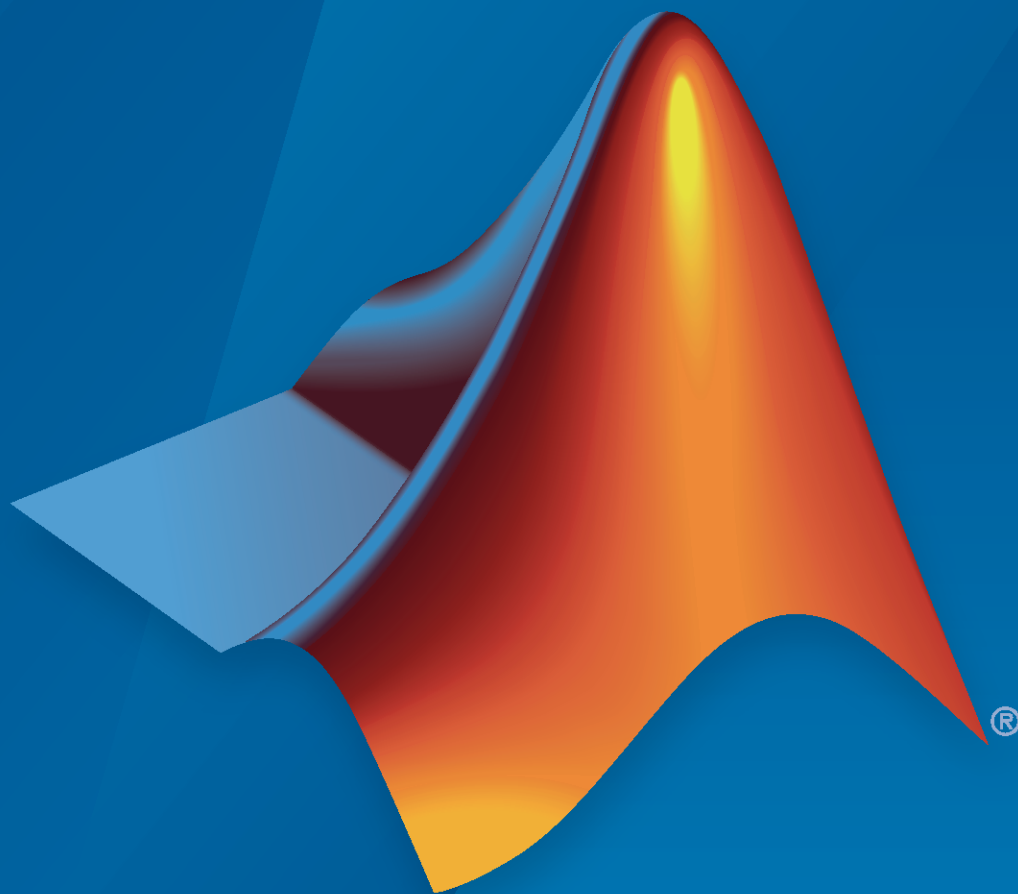


Stateflow®

API



MATLAB® & SIMULINK®

R2023a



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

© COPYRIGHT 2004-2023 by The MathWorks, Inc.

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

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

Trademarks

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

Patents

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

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)
March 2020	Online only	Revised for Version 10.2 (Release 2020a)
September 2020	Online only	Revised for Version 10.3 (Release 2020b)
March 2021	Online only	Revised for Version 10.4 (Release 2021a)
September 2021	Online only	Revised for Version 10.5 (Release 2021b)
March 2022	Online only	Revised for Version 10.6 (Release 2022a)
September 2022	Online only	Revised for Version 10.7 (Release 2022b)
March 2023	Online only	Revised for Version 10.8 (Release 2023a)

Overview of the Stateflow API	1-2
Hierarchy of Stateflow API Objects	1-2
Access Stateflow API Objects	1-4
Modify Properties of API Objects	1-4
Call API Object Functions	1-5
Access Objects in Your Stateflow Chart	1-6
Find Objects in a Chart	1-6
Navigate the Stateflow Hierarchy	1-7
Retrieve Recently Selected Objects	1-9
Modify Properties and Call Functions of Stateflow Objects	1-11
Call Object Functions	1-11
Access Properties by Using Dot Notation	1-11
Get and Set the Values of Multiple Properties	1-12
Create and Delete Stateflow Objects	1-13
Create Stateflow Objects	1-13
Delete Stateflow Objects	1-14
Specify Labels in States and Transitions Programmatically	1-16
Enter Labels on Transitions	1-16
Enter Multiline Labels in States	1-17
Create Charts by Using the Stateflow API	1-19
Create Charts by Using a MATLAB Script	1-24
Refactor Charts Programmatically	1-26
Summary of Stateflow API Objects and Properties	1-36
Stateflow.Annotation	1-36
Stateflow.AtomicBox	1-39
Stateflow.AtomicSubchart	1-40
Stateflow.Box	1-43
Stateflow.Chart	1-45
Stateflow.Clipboard	1-50
Stateflow.Data	1-50
Stateflow.EMChart	1-56
Stateflow.EMFunction	1-59
Stateflow.Editor	1-62
Stateflow.Event	1-62
Stateflow.Function	1-63
Stateflow.Junction	1-65

Stateflow.Machine	1-66
Stateflow.Message	1-68
Stateflow.Port	1-73
Stateflow.SLFunction	1-75
Stateflow.SimulinkBasedState	1-76
Stateflow.State	1-79
Stateflow.StateTransitionTableChart	1-83
Stateflow.Transition	1-88
Stateflow.TruthTable	1-90
Stateflow.TruthTableChart	1-93

API Object Reference

2

API Object Function Reference

3

Using the Stateflow API

- “Overview of the Stateflow API” on page 1-2
- “Access Objects in Your Stateflow Chart” on page 1-6
- “Modify Properties and Call Functions of Stateflow Objects” on page 1-11
- “Create and Delete Stateflow Objects” on page 1-13
- “Specify Labels in States and Transitions Programmatically” on page 1-16
- “Create Charts by Using the Stateflow API” on page 1-19
- “Create Charts by Using a MATLAB Script” on page 1-24
- “Refactor Charts Programmatically” on page 1-26
- “Summary of Stateflow API Objects and Properties” on page 1-36

Overview of the Stateflow API

In this section...
“Hierarchy of Stateflow API Objects” on page 1-2
“Access Stateflow API Objects” on page 1-4
“Modify Properties of API Objects” on page 1-4
“Call API Object Functions” on page 1-5

The Stateflow application programming interface (API) allows you to create or change Stateflow charts from the MATLAB Command Window. By placing Stateflow API commands in a MATLAB function or script, you can:

- Automate your chart modification operations by executing several editing steps in a single command.
- Eliminate repetitive chart creation steps by producing a "base" Stateflow chart that you can reuse as a template for your applications.
- Produce a specialized report of your model.

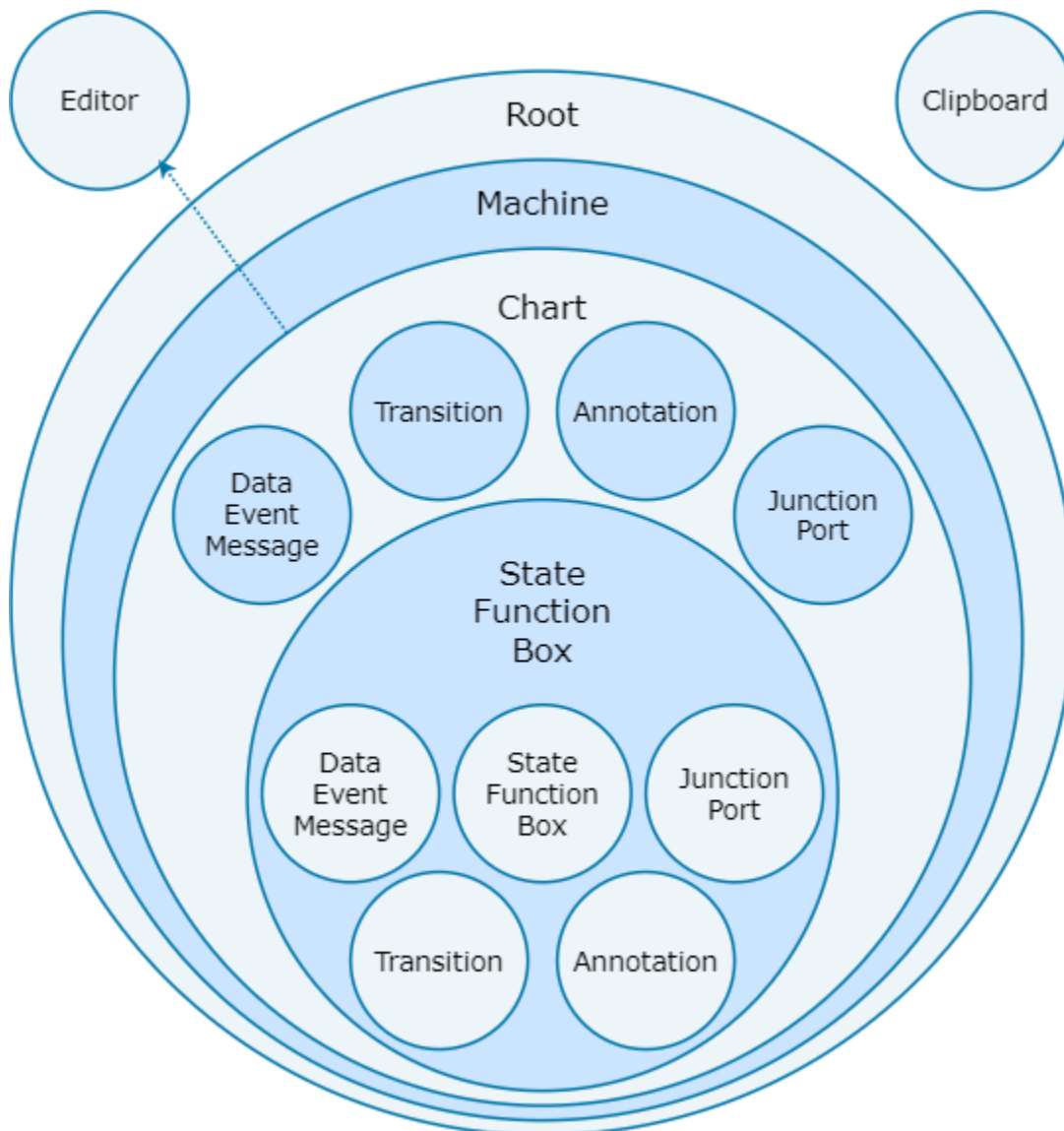
The Stateflow API consists of objects that represent the graphical and nongraphical objects of a Stateflow chart. For example, the API objects `Stateflow.State` and `Stateflow.Transition` represent states and transitions in a Stateflow chart. When you modify the properties of an API object or call one of its object functions, 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 affect the corresponding API object.

Note You cannot undo any operation in the Stateflow Editor that you perform by using the Stateflow API. If you perform an editing operation through the API, the **Undo** and **Redo** buttons in the quick access toolbar are disabled.

Hierarchy of Stateflow API Objects

Stateflow API objects are organized in a containment hierarchy. 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. The Stateflow API hierarchy follows the same rules of containment as the Stateflow 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 hierarchy of objects in the Stateflow API.



The hierarchy consists of four levels of containment:

- **Root** — The `Simulink.Root` object is the parent of all Stateflow API objects. It is a placeholder at the top of the Stateflow API hierarchy that distinguishes Stateflow objects from other objects in a Simulink® model. You automatically create the `Simulink.Root` object when you add a Stateflow chart, a State Transition Table block, a Truth Table block, or a MATLAB Function block to a Simulink model, or when you load a model that contains one of these blocks.
- **Machine** — From a Stateflow perspective, `Stateflow.Machine` objects are equivalent to Simulink models. A `Stateflow.Machine` object contains objects that represent the Stateflow charts, State Transition Table blocks, Truth Table blocks, and MATLAB Function blocks in a model.
- **Chart** — `Stateflow.Chart`, `Stateflow.StateTransitionTableChart`, `Stateflow.TruthTableChart`, and `Stateflow.EMChart` objects represent Stateflow charts, State Transition Table blocks, Truth Table blocks, and MATLAB Function blocks, respectively. Objects in this level of the hierarchy can contain objects that represent states, functions, boxes, data, events, messages, transitions, junctions, entry and exit ports, and annotations.

- **States, Functions, and Boxes** — This level of the hierarchy includes `Stateflow.State`, `Stateflow.Function`, and `Stateflow.Box` objects that represent states, functions, and boxes, respectively. These objects can contain other objects that represent states, functions, boxes, data, events, messages, transitions, junctions, entry and exit ports, and annotations. Levels of nesting can continue indefinitely.

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

- **Editor** — `Stateflow.Editor` objects provide access to the graphical aspects of charts and state transition tables. For each `Stateflow.Chart` or `Stateflow.StateTransitionTableChart` object, there is a `Stateflow.Editor` object that you can use to control the position, size, and magnification level of the Stateflow Editor. For more information, see “Zoom in on Stateflow Chart” on page 2-64, “Zoom out on Stateflow Chart” on page 2-65, and “Set Zoom Factor” on page 2-65.
- **Clipboard** — The `Stateflow.Clipboard` object has two functions, `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 by Grouping” on page 2-28 and “Copy and Paste Array of Objects” on page 2-30.

Access Stateflow API Objects

To use the Stateflow API, you begin by accessing the `Simulink.Root` object, which is the parent of all objects in the Stateflow API. You use the `Simulink.Root` object to access the other API objects in your model. For example:

- 1 Create a Simulink model with an empty Stateflow chart by calling the function `sfnew`.

```
sfnew
```

- 2 Use the function `sfroot` to access the `Simulink.Root` object.

```
rt = sfroot;
```

- 3 Call the `find` function to access the `Stateflow.Chart` object that corresponds to the chart in your model.

```
ch = find(rt, "-isa", "Stateflow.Chart");
```

- 4 Call the `Stateflow.State` function to add a state to the chart. This function returns an `Stateflow.State` object that corresponds to the new state.

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

- 5 Display the new state in the Stateflow Editor.

```
view(st)
```

For more information, see “Access Objects in Your Stateflow Chart” on page 1-6 and “Create Charts by Using the Stateflow API” on page 1-19.

Modify Properties of API Objects

API objects have properties that correspond to the values you set in the Stateflow Editor. For example, to use the editor to change the position of a state, you click and drag the state. With the Stateflow API, you change the position of a state by modifying the `Position` property of the corresponding `Stateflow.State` object:

```
st.Position = [10 20 100 80];
```

For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Call API Object Functions

API objects have functions that correspond to actions in the Stateflow Editor. For example, to use the editor to open the Properties dialog box for a transition, you right-click the transition and select **Properties**. With the Stateflow API, you open this dialog box by calling the `dialog` function of the corresponding `Stateflow.Transition` object:

```
dialog(tr);
```

For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

See Also

Functions

`find` | `dialog` | `sfnew` | `sfroot` | `view`

Objects

`Stateflow.Box` | `Stateflow.Chart` | `Stateflow.Clipboard` | `Stateflow.Editor` | `Stateflow.EMChart` | `Stateflow.Function` | `Stateflow.Machine` | `Stateflow.State` | `Stateflow.StateTransitionTableChart` | `Stateflow.Transition` | `Stateflow.TruthTableChart`

More About

- “Create Charts by Using the Stateflow API” on page 1-19
- “Create Charts by Using a MATLAB Script” on page 1-24
- “Access Objects in Your Stateflow Chart” on page 1-6
- “Modify Properties and Call Functions of Stateflow Objects” on page 1-11

Access Objects in Your Stateflow Chart

The objects in the Stateflow API represent the graphical and nongraphical objects of a Stateflow chart. For example, the API objects `Stateflow.State` and `Stateflow.Transition` represent states and transitions in a Stateflow chart. For more information, see “Overview of the Stateflow API” on page 1-2.

Find Objects in a Chart

With the `find` function, you can locate an API object by specifying search criteria. You can combine criteria such as:

- The type of object
- The name of a property or function
- A property name and value

For example, this command searches the `Simulink.Root` object and returns every `Stateflow.State` object with the name `On`:

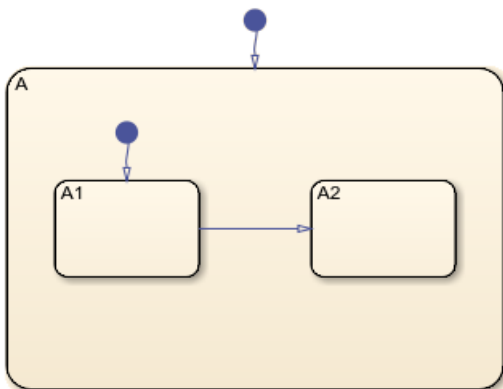
```
onState = find(sfroot, "-isa", "Stateflow.State", Name="On")
```

If more than one object meets the search criteria, `find` returns an array of qualifying objects. For example, if more than one chart is open, this command returns an array of `Stateflow.Chart` objects:

```
chartArray = find(sfroot, "-isa", "Stateflow.Chart")
```

Find Objects at Specific Levels of Containment

By default, the `find` function finds objects at all depths of containment within an object. For example, suppose that `ch` is a `Stateflow.Chart` object that corresponds to this chart. The chart contains a parent state `A` with two child states, `A1` and `A2`. For more information on this example, see “Create Charts by Using a MATLAB Script” on page 1-24.



Calling the `find` function to find all the states in this chart returns an array with three `Stateflow.State` objects:

```
states = find(ch, "-isa", "Stateflow.State");
get(states, "Name")
```

```
ans =
    3x1 cell array
    {'A'}
    {'A1'}
    {'A2'}
```

To limit the maximum containment depth of a search, use the `"-depth"` argument as part of your search criteria. For example, to find the only `Stateflow.State` object at the first level of containment in `ch`, enter:

```
sA = find(ch, "-isa", "Stateflow.State", "-depth", 1);
sA.Name
ans =
    'A'
```

Similarly, you can call the `find` function to search for states in the first level of containment in the `Stateflow.State` object `sA`. In this case, the search includes the zeroth level of containment, which is the searched object itself:

```
states = find(sA, "-isa", "Stateflow.State", "-depth", 1);
get(states, "Name")
ans =
    3x1 cell array
    {'A'}
    {'A1'}
    {'A2'}
```

To exclude state A from the search results, call the MATLAB function `setdiff`:

```
childStates = setdiff(states, sA);
get(childStates, "Name")
ans =
    2x1 cell array
    {'A1'}
    {'A2'}
```

Navigate the Stateflow Hierarchy

After you access an API object, you can use the `getChildren` and `getParent` functions to navigate through the Stateflow hierarchy and identify the children that the object contains or the parent that contains the object.

Find Child Objects

To find the children of an API object, call the `getChildren` function. For instance, suppose that `ch` is the `Stateflow.Chart` object that corresponds to the chart in the previous example. Calling the `getChildren` function on `ch` returns an array that contains a `Stateflow.State` object and a `Stateflow.Transition` object.

```
children = getChildren(ch);  
arrayfun(@class,children,UniformOutput=false)
```

```
ans =
```

```
2×1 cell array
```

```
{'Stateflow.State' }  
{'Stateflow.Transition'}
```

The first element in the array is a `Stateflow.State` object that corresponds to state A.

```
state = children(1);  
state.Name
```

```
ans =
```

```
'A'
```

The second element in the array is a `Stateflow.Transition` object that corresponds to the default transition into state A.

```
children(2).Source
```

```
ans =
```

```
[]
```

```
children(2).Destination.Name
```

```
ans =
```

```
'A'
```

Similarly, calling the `getChildren` function on the state returns an array that contains two `Stateflow.State` objects and two `Stateflow.Transition` objects.

```
grandchildren = getChildren(state);  
arrayfun(@class,grandchildren,UniformOutput=false)
```

```
ans =
```

```
4×1 cell array
```

```
{'Stateflow.State' }  
{'Stateflow.State' }  
{'Stateflow.Transition'}  
{'Stateflow.Transition'}
```

The first and second elements in this array are `Stateflow.State` objects that correspond to the states A1 and A2.

```
grandchildren(1).Name
```

```
ans =
```

```
'A1'
```

```
grandchildren(2).Name
```

```
ans =
    'A2'
```

The third and fourth elements in `grandchildren` are `Stateflow.Transition` objects that correspond to the transitions into states A1 and between state A1 and A2, respectively.

```
grandchildren(3).Source
```

```
ans =
    []
```

```
grandchildren(3).Destination.Name
```

```
ans =
    'A1'
```

```
grandchildren(4).Source.Name
```

```
ans =
    'A1'
```

```
grandchildren(4).Destination.Name
```

```
ans =
    'A2'
```

Find Parent Object

To find the parent of an API object, call the `getParent` function. For instance, suppose that `sA1` is the `Stateflow.State` object that corresponds to state A1 in the previous example. Calling the `getParent` function on `sA1` returns the `Stateflow.State` object that corresponds to state A:

```
parent = getParent(sA1);
parent.Name
```

```
ans =
    'A'
```

Similarly, calling the `getParent` function on `parent` returns the `Stateflow.Chart` object that corresponds to the chart:

```
grandparent = getParent(parent);
grandparent.Name
```

```
ans =
    'Chart'
```

Retrieve Recently Selected Objects

You can retrieve the most recently selected objects in a chart by calling the `sfgco` function. This function returns a single object or an array of objects, depending on your selection.

For instance, suppose that you select the transition from state A1 to state A2 in the previous example. Calling `sfgco` returns the corresponding `Stateflow.Transition` object:

```
tr = sfgco;
str = str = "Transition from "+tr.Source.Name+" to "+tr.Destination.Name

str =

    "Transition from A1 to A2"
```

Similarly, if you simultaneously select the three states in the chart, calling `sfgco` returns an array of `Stateflow.State` objects.

```
states = sfgco;
get(states, "Name")

ans =

    3×1 cell array

    {'A'}
    {'A1'}
    {'A2'}
```

Note When you use `sfgco` to access multiple objects, the order of the objects in the array depends on the order in which you select the objects.

See Also

Functions

`find` | `getChildren` | `getParent` | `setdiff` | `sfgco` | `arrayfun` | `class`

Objects

`Stateflow.Chart` | `Stateflow.State` | `Stateflow.Transition`

More About

- “Overview of the Stateflow API” on page 1-2
- “Modify Properties and Call Functions of Stateflow Objects” on page 1-11
- “Create and Delete Stateflow Objects” on page 1-13
- “Create Charts by Using the Stateflow API” on page 1-19

Modify Properties and Call Functions of Stateflow Objects

In this section...

“Call Object Functions” on page 1-11

“Access Properties by Using Dot Notation” on page 1-11

“Get and Set the Values of Multiple Properties” on page 1-12

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. For example, to use the editor to change the position of a state, you click and drag the state. With the Stateflow API, you change the position of a state by modifying the `Position` property of the corresponding `Stateflow.State` object:

```
st.Position = [10 20 100 80];
```

Additionally, object functions provide services that correspond to actions in the Stateflow Editor. For example, to use the editor to open the Properties dialog box for a transition, you right-click the transition and select **Properties**. With the Stateflow API, you open this dialog box by calling the `dialog` function of the corresponding `Stateflow.Transition` object:

```
dialog(tr);
```

Call Object Functions

To call a function of an API object, use standard function-call notation. For example, to open the Chart properties dialog box, call the `dialog` function of the corresponding `Stateflow.Chart` object `ch`:

```
dialog(ch)
```

Access Properties by Using Dot Notation

To access a property of an API object, use dot notation. For example, to see the value of the `StateMachineType` property for the `Stateflow.Chart` object `ch`, enter:

```
ch.StateMachineType
```

Similarly, to change the action language of the chart, modify its `ActionLanguage` property:

```
ch.ActionLanguage = "MATLAB";
```

To access the subproperties of an API property, you can nest multiple property names in a single expression that uses dot notation. For example, you can set an entry breakpoint on a chart by changing the subproperty `Debug.Breakpoints.OnEntry` of the corresponding `Stateflow.Chart` object:

```
ch.Debug.Breakpoints.OnEntry = true;
```

When a property or function returns another API object, you can also access the properties and functions for the second object by using nested dot notation. For example, the `Machine` property of a `Stateflow.Chart` returns the `Stateflow.Machine` object that contains the corresponding chart. To access the `Name` property of this `Stateflow.Machine` object, enter the expression:

```
machineName = ch.Machine.Name;
```

Similarly, the `defaultTransitions` function returns an array of `Stateflow.Transition` objects that correspond to the default transitions in the chart. If the chart contains only one default transition, you can retrieve its label by entering:

```
label = defaultTransitions(ch).LabelString;
```

If the chart contains more than one default transition, you must first store the array and then use an array index to retrieve each label:

```
transitions = defaultTransitions(ch);  
label1 = transitions(1).LabelString;  
label2 = transitions(2).LabelString;
```

Get and Set the Values of Multiple Properties

You can access multiple properties of an API object in a single command by calling the `get` function. For example, to obtain the name and description for the `Stateflow.Chart` object `ch`, enter:

```
chartInfo = get(ch, {"Name", "Description"});
```

You can also use the `get` to access properties of multiple API objects. For example, this command returns a cell array that contains the names and descriptions of the `Stateflow.Chart` objects in the array `chartArray`:

```
chartInfo = get(chartArray, {"Name", "Description"});
```

Similarly, you can change the value of multiple properties by calling the `set` function. For example, to change the name and description of the `Stateflow.Chart` object `ch`, enter:

```
set(ch, {"Name", "Description"}, {"Rectifier", "Half-wave rectifier."})
```

To set the names and descriptions of the `Stateflow.Chart` objects in the array `chartArray`, enter:

```
set(chartArray, {"Name", "Description"}, chartInfo)
```

In this command, `chartInfo` must be an N -by-2 cell array, where N equals the number of charts in `chartArray`. The first column in `chartInfo` contains the new chart names, and the second column contains the new descriptions.

See Also

Functions

`defaultTransitions` | `dialog` | `fitToView`

Objects

`Stateflow.Chart` | `Stateflow.State` | `Stateflow.Transition`

More About

- “Overview of the Stateflow API” on page 1-2
- “Summary of Stateflow API Objects and Properties” on page 1-36

Create and Delete Stateflow Objects

The objects in the Stateflow API represent the graphical and nongraphical objects of a Stateflow chart. For example, the API objects `Stateflow.State` and `Stateflow.Transition` represent states and transitions in a Stateflow chart. For more information, see “Overview of the Stateflow API” on page 1-2.

Create Stateflow Objects

Stateflow API objects are organized in the containment hierarchy described in “Hierarchy of Stateflow API Objects” on page 1-2. To create a Stateflow object as the child of a parent object, you begin by accessing the parent object. Then use the parent object as the input argument to a function that creates the child object. For example, to add a new `Stateflow.State` object in a `Stateflow.Chart` object, follow these steps:

- 1 Access the parent object `ch` as described in “Access Objects in Your Stateflow Chart” on page 1-6.
- 2 Call the `Stateflow.State` function using the parent object `ch` as an argument.

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

- 3 Display the new state in the Stateflow Editor by calling the `view` function. Use the `Stateflow.State` object as the argument to the function.

```
view(st)
```

- 4 Make changes to the state by modifying the properties of the `Stateflow.State` object. For example, you can set the name and position of the state by modifying the `Name` and `Position` properties. To set the `Position` property, specify the new position as a four-element vector in which the first two values are the (x,y) coordinates of the upper-left corner of the state and the last two values are the width and height of the state.

```
st.Name = "A";
st.Position = [30 30 90 60];
```

You can also connect the new state to other states or junctions in your chart by creating a `Stateflow.Transition` object and setting its `Source` or `Destination` properties to `st`.

For an example of how to add states, transitions, and data objects to a chart, see “Create Charts by Using the Stateflow API” on page 1-19.

Graphical Object Containment

When you create a graphical object such as a state, function, box, junction, or annotation, it appears in the upper-left corner of its parent object. You can move the graphical object to a different location by modifying its `Position` property, as explained in the previous example.

When you create a transition, it appears in the upper-left corner of the chart or subchart where you can view the parent object. You can move the transition to a different location by setting its source and destination or by modifying its `SourceEndPoint`, `MidPoint`, and `DestinationEndPoint` properties.

A graphical object must be located inside the boundary of its parent. Repositioning a graphical object can change its parent or result in an undefined parent error. You can check for this condition by examining the value of the `BadIntersection` property of an object. This property is `true` if the

edges of the graphical object overlap with another graphical object. Set the position and size of objects so that they are separate from other objects.

You cannot move an object in a subcharted state, box, or graphical function to a different level of the chart hierarchy by changing its position. Instead, copy and paste the object from one parent object to another. Then delete the original object. For more information, see “Copy and Paste by Grouping” on page 2-28 and “Copy and Paste Array of Objects” on page 2-30.

Nongraphical Object Containment

When you create nongraphical objects such as data, events, or messages, they appear in the Model Explorer and in the **Symbols** pane at the hierarchical level of their parent object. You can also see the location of the parent object by inspecting the `Path` property of an object.

You cannot change the parent of a nongraphical object programmatically. Instead, use the Model Explorer. For more information, see “Use the Model Explorer with Stateflow Objects”.

Delete Stateflow Objects

You can delete most objects in a Stateflow chart by calling the function `delete`. For example, to delete a `Stateflow.State` object `st`, enter:

```
delete(st);
```

After you delete the state, the variable `st` still exists in the MATLAB workspace, but it is no longer associated with the state.

Deleting a graphical object also deletes the nongraphical children of the deleted object. The graphical children of the deleted object become children of the parent of the deleted object.

Note You cannot use the `delete` function to delete objects of these types:

- `Simulink.Root`
 - `Stateflow.Machine`
 - `Stateflow.Chart`
 - `Stateflow.EMChart`
 - `Stateflow.StateTransitionTableChart`
 - `Stateflow.TruthTableChart`
 - `Stateflow.Clipboard`
 - `Stateflow.Editor`
-

See Also

Functions

`delete` | `view`

Objects

`Stateflow.Chart` | `Stateflow.State` | `Stateflow.Transition`

More About

- “Overview of the Stateflow API” on page 1-2
- “Access Objects in Your Stateflow Chart” on page 1-6
- “Create Charts by Using the Stateflow API” on page 1-19

Specify Labels in States and Transitions Programmatically

When using the Stateflow API, specify the labels of states and transitions by assigning a character vector to the `LabelString` property.

To extract parts of the state or transition label, use the properties of the `Stateflow.State` and `Stateflow.Transition` objects listed in this table.

API Object	Property	Description
Stateflow.State	DuringAction	Text in the <code>during</code> action in this state. This property is not supported in Moore charts.
	EntryAction	Text in the <code>entry</code> action in this state. This property is not supported in Moore charts.
	ExitAction	Text in the <code>exit</code> action in this state. This property is not supported in Moore charts.
	MooreAction	Text in the action in this state. This property is supported only in Moore charts. For more information, see “Design Guidelines for Moore Charts”.
	Name	Name of this state.
	OnAction	Text in the <code>on</code> actions in this state, parsed as a cell array of this form: {'trigger1','action1',...,'triggerN','actionN'} This property is not supported in Moore charts.
Stateflow.Transition	Condition	Text in the condition on this transition.
	ConditionAction	Text in the condition action on this transition.
	TransitionAction	Text in the transition action on this transition.
	Trigger	Text in the trigger on this transition.

With the exception of `Name`, all of these properties are read-only. For more information on the syntax for state and transition labels, see “Define Actions in a State” and “Define Actions in a Transition”.

Enter Labels on Transitions

Suppose that `tr` is the `Stateflow.Transition` object that corresponds to a transition. You can assign a label that specifies a trigger, condition, and condition action on this transition by entering:

```
tr.LabelString = "trigger[guard]{action();}";
```



To extract the trigger, condition, and condition action specified by the transition label, enter:

```

trigger = tr.Trigger
trigger =
    'trigger'
cond = tr.Condition
cond =
    'guard'
action = tr.ConditionAction
action =
    'action()';

```

Enter Multiline Labels in States

There are two equivalent ways to enter multiline labels for states and transitions. For example, Suppose that `sA` is a `Stateflow.State` object that corresponds to a state. To enter a multiline label with entry and during actions, you can:

- Call the MATLAB function `sprintf` and use the escape sequence `\n` to insert newline characters:

```

str = sprintf("A\nen: action1();\ndu: action2();\nen,du: action3();");
sA.LabelString = str;

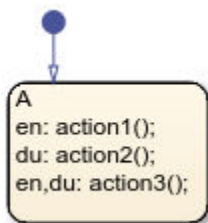
```

- Enter a concatenated text expression that uses the function `newline` to create newline characters:

```

str = "A" + newline + ...
      "en: action1();" + newline + ...
      "du: action2();" + newline + ...
      "en,du: action3();"
sA.LabelString = str;

```



To extract the state name, entry action, and during action specified by the state label, enter:

```

name = sA.Name
name =
    'A'
entry = sA.EntryAction
entry =

```

```
        ' action1();  
          action3();'  
  
during = sA.DuringAction  
  
during =  
  
        ' action2();  
          action3();'
```

See Also

Functions

`sprintf`

Objects

`Stateflow.State` | `Stateflow.Transition`

More About

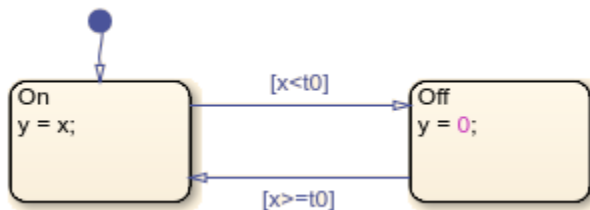
- “Overview of the Stateflow API” on page 1-2
- “Create Charts by Using the Stateflow API” on page 1-19
- “Represent Operating Modes by Using States”
- “Transition Between Operating Modes”

Create Charts by Using the Stateflow API

This example shows how to create a Stateflow® chart by using the Stateflow application programming interface (API). The Stateflow API is a tool to create or change Stateflow charts through MATLAB® commands. For more information, see “Overview of the Stateflow API” on page 1-2.

Create a Stateflow Chart

This Stateflow chart presents the logic underlying a half-wave rectifier. The chart contains two states labeled *On* and *Off*. In the *On* state, the chart output signal *y* is equal to the input *x*. In the *Off* state, the output signal is set to zero. When the input signal crosses some threshold *t0*, the chart transitions between these states. The actions in each state update the value of *y* at each time step of the simulation.



For more information on simulating this chart, see “Construct and Run a Stateflow Chart”.

1. Create a Simulink® model called `rectify` that contains an empty Stateflow Chart block.

```
sfnew rectify
```

2. Access the `Stateflow.Chart` object that corresponds to the chart in your model by calling the `find` function. Use the function `sfroot` to access the `Simulink.Root` object, which is the parent of all objects in the Stateflow API.

```
ch = find(sfroot, "-isa", "Stateflow.Chart", ...
    Path="rectify/Chart");
```

3. Open the chart in the Stateflow Editor by calling the `view` function.

```
view(ch);
```

4. Change the action language by modifying the `ActionLanguage` property of the chart.

```
ch.ActionLanguage = "C";
```

Add States

To create a Stateflow API object as the child of a parent object, use the parent object as the input argument to a function that creates the child object. For more information, see “Create and Delete Stateflow Objects” on page 1-13.

1. Call the `Stateflow.State` function to add a state to the chart.

```
s1 = Stateflow.State(ch);
```

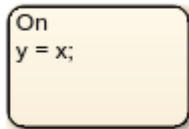
2. Adjust the position of the state by changing the `Position` property of the corresponding `State` object. Specify the new position as a four-element vector in which the first two values are the (x,y)

coordinates of the upper-left corner of the state and the last two values are the width and height of the state.

```
s1.Position = [30 30 90 60];
```

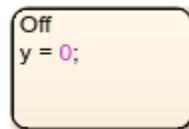
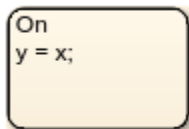
3. Specify the name and label for the state by changing the `LabelString` property, as described in “Specify Labels in States and Transitions Programmatically” on page 1-16.

```
s1.LabelString = "On"+newline+"y = x;";
```



4. Create a second state. Adjust its position and specify its name and label.

```
s2 = Stateflow.State(ch);  
s2.Position = [230 30 90 60];  
s2.LabelString = "Off"+newline+"y = 0;";
```



Add Transitions

When you add a transition, you specify its source and destination by modifying its `Source` and `Destination` properties. For a default transition, you specify a destination but no source.

1. Call the `Stateflow.Transition` function to add a transition to the chart.

```
t1 = Stateflow.Transition(ch);
```

2. Set the transition source and destination.

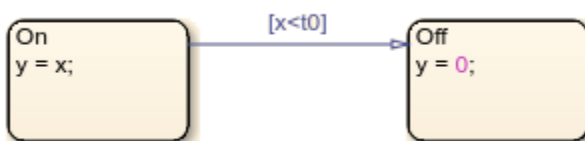
```
t1.Source = s1;  
t1.Destination = s2;
```

3. Adjust the position of the transition by modifying its `SourceClock` property.

```
t1.SourceClock = 2.1;
```

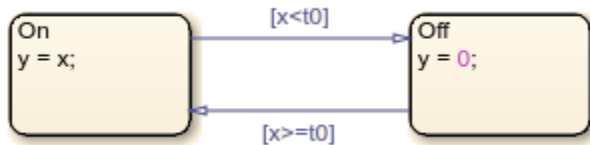
4. Specify the transition label and its position by changing the `LabelString` and `LabelPosition` properties.

```
t1.LabelString = "[x<t0]";  
t1.LabelPosition= [159 23 31 16];
```



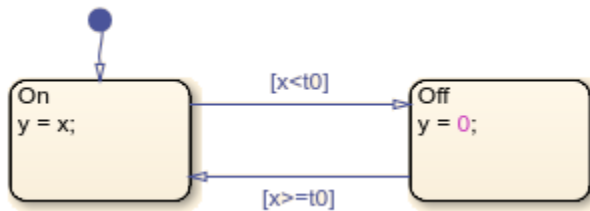
5. Create a second transition. Specify its source, destination, and label.

```
t2 = Stateflow.Transition(ch);
t2.Source = s2;
t2.Destination = s1;
t2.SourceClock = 8.1;
t2.LabelString = "[x>=t0]";
t2.LabelPosition = [155 81 38 16];
```



6. Add a default transition to the state On. To make a vertical transition, modify the values of the SourceEndpoint and Midpoint properties. For more information, see “Add a Default Transition” on page 2-197.

```
t0 = Stateflow.Transition(ch);
t0.Destination = s1;
t0.DestinationClock = 0;
t0.SourceEndpoint = t0.DestinationEndpoint-[0 30];
t0.Midpoint = t0.DestinationEndpoint-[0 15];
```



Add Data

Before you can simulate your chart, you must define each data symbol that you use in the chart and specify its scope and type.

1. Call the `Stateflow.Data` function to add a data object that represents the input to the chart.

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

2. Specify the name of the data object as `x` and its scope as `Input`.

```
x.Name = "x";
x.Scope = "Input";
```

3. To specify that the input `x` has type `double`, set its `Props.Type.Method` property to `Built-in`. The default built-in data type is `double`.

```
x.Props.Type.Method = "Built-in";
x.DataType
```

```
ans =
'double'
```

4. Add a data object that represents the output for the chart. Specify its name as `y` and its scope as `Output`.

```
y = Stateflow.Data(ch);  
y.Name = "y";  
y.Scope = "Output";
```

5. To specify that the output `y` has type `single`, set its `Props.Type.Method` property to `Built-in` and its `Data Type` property to `single`.

```
y.Props.Type.Method = "Built-in";  
y.DataType = "single";  
y.DataType
```

```
ans =  
'single'
```

6. Add a data object that represents the transition threshold in the chart. Specify its name as `t0` and its scope as `Constant`. Set its initial value to `0`.

```
t0 = Stateflow.Data(ch);  
t0.Name = "t0";  
t0.Scope = "Constant";  
t0.Props.InitialValue = "0";
```

7. To specify that the threshold `t0` has a fixed-point data type, set its `Props.Type.Method` property to `Fixed-point`. Then specify the values of the `Props.Type` properties that apply to fixed-point data.

```
t0.Props.Type.Method = "Fixed point";  
t0.Props.Type.Signed = true;  
t0.Props.Type.WordLength = "5";  
t0.Props.Type.Fixpt.ScalingMode = "Binary point";  
t0.Props.Type.Fixpt.FractionLength = "2";  
t0.DataType
```

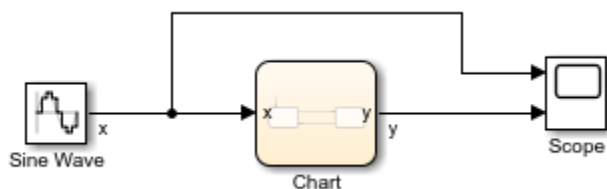
```
ans =  
'fixdt(1,5,2)'
```

Save and Simulate Your Chart

To save the model that contains your completed chart, call the `sfsave` function.

```
sfsave
```

To simulate the chart, connect it to other blocks in the Simulink model through input and output ports.



For more information, see “Simulate the Chart as a Simulink Block”.

See Also

Blocks

Chart

Functions

find | sfnew | sfroot | sfsave | view

Objects

Stateflow.State | Stateflow.Transition | Stateflow.Data

More About

- “Overview of the Stateflow API” on page 1-2
- “Create and Delete Stateflow Objects” on page 1-13
- “Modify Properties and Call Functions of Stateflow Objects” on page 1-11
- “Specify Labels in States and Transitions Programmatically” on page 1-16

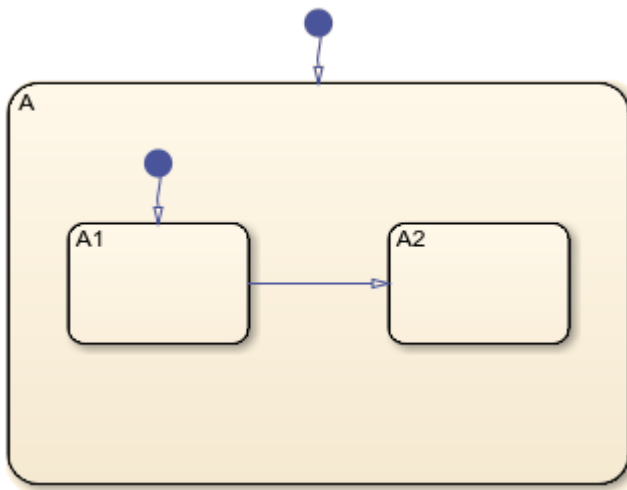
Create Charts by Using a MATLAB Script

This example shows how to include Stateflow® API commands in a MATLAB® function or script. Creating a script of API commands allows you to avoid repetitive chart creation steps and recreate the same model with a single command. For more information, see “Overview of the Stateflow API” on page 1-2.

Run the MATLAB Function

The function `makeMyModel`, which is defined at the bottom of this page on page 1-24, produces a "base" Stateflow chart that you can reuse as a template for your applications.

```
ch = makeMyModel;
view(ch)
```



Create Base Chart Function

This function creates a Stateflow chart and returns the corresponding `Stateflow.Chart` object.

```
function ch = makeMyModel
```

Create model and access new `Stateflow.Chart` object.

```
rt = sfroot;
prev_machines = find(rt,"-isa","Stateflow.Machine");
sfnew;
curr_machines = find(rt,"-isa","Stateflow.Machine");
m = setdiff(curr_machines,prev_machines);
ch = find(m,"-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 transition from A1 to A2.

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

Add default transition to state A.

```
dtA = Stateflow.Transition(ch);  
dtA.Destination = sA;  
dtA.DestinationOClock = 0;  
dtA.SourceEndPoint = dtA.DestinationEndpoint-[0 30];  
dtA.MidPoint = dtA.DestinationEndpoint-[0 15];
```

Add default transition to state A1.

```
dtA1 = Stateflow.Transition(ch);  
dtA1.Destination = sA1;  
dtA1.DestinationOClock = 0;  
dtA1.SourceEndPoint = dtA1.DestinationEndpoint-[0 30];  
dtA1.MidPoint = dtA1.DestinationEndpoint-[0 15];
```

end

See Also

Functions

view | sfroot | find | sfnew | setdiff

Objects

Stateflow.State | Stateflow.Transition

More About

- “Overview of the Stateflow API” on page 1-2
- “Create Charts by Using the Stateflow API” on page 1-19

Refactor Charts Programmatically

This example shows how to use the Stateflow® API to improve the legibility of Stateflow charts. You can programmatically detect and fix common stylistic patterns such as inconsistent state names, deeply nested states, and unnecessary junctions. For more information about using the Stateflow API, see “Overview of the Stateflow API” on page 1-2.

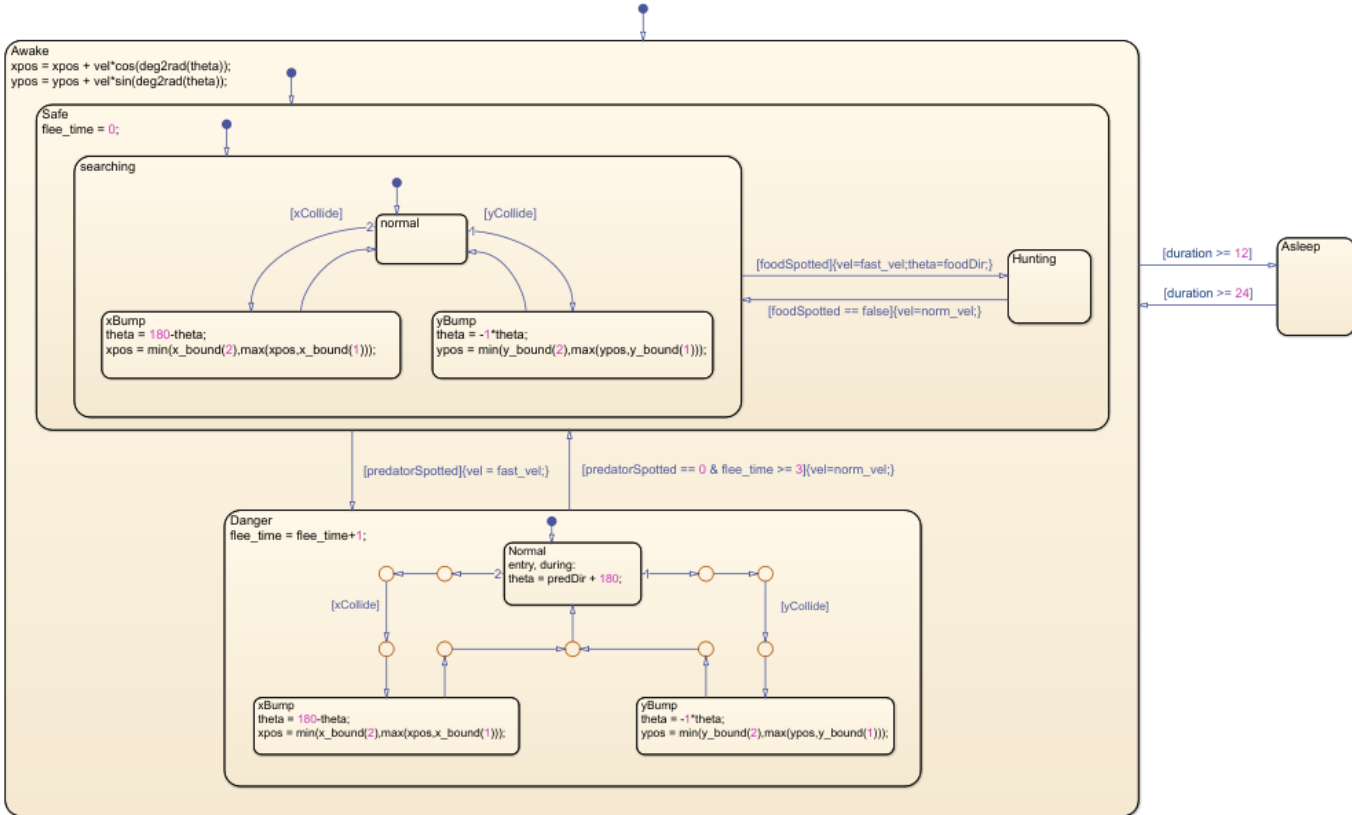
Open the Model

This model contains a Stateflow chart that emulates the behavior of an insect. In this model:

- The insect is confined to a box.
- The insect can only detect objects that are a short distance in front of the insect.
- The insect moves in a straight line until it hits a wall or detects a predator or prey.
- When the insect hits a wall, the insect bounces off the wall and continues in the opposite direction.
- When the insect sees a predator, the insect runs away from the predator at an increased speed.
- When the insect sees prey, the insect heads towards the prey at an increased speed.
- The insect stops to rest every twelve hours.

The chart uses an inconsistent naming scheme in which some state names start with a capital letter and some state names do not. Additionally, the chart has a deep hierarchy with more than three levels of nested states. Finally, the chart contains superfluous junctions that have only one incoming transition and one outgoing transition.

```
model = "sfInsectExample";  
open_system(model)
```

To access the Stateflow.Chart object for the chart, call the find function.

```
ch = find(sfroot, "-isa", "Stateflow.Chart", Name="BehavioralLogic");
```

Fix Inconsistent State Names

To improve chart readability, use the same naming convention for all states in the chart. For example, you can use a naming scheme in which all state names start with a capital letter. To search for states with names that do not follow this naming convention, call the find function with the optional argument `-regexp` and specify a regular expression. For more information, see “Regular Expressions”.

```
misnamedStates = find(ch, "-isa", "Stateflow.State", ...
    "-regexp", "Name", "^[a-z]\w*");
```

To inspect the search results, display the path from the model to each state by accessing the Path and Name properties of each Stateflow.State object.

```
numMisnamedStates = numel(misnamedStates);
misnamedStateNames = "";

for i = 1:numMisnamedStates
    state = misnamedStates(i);
    misnamedStateNames = misnamedStateNames + ...
        newline + " * " + state.Path + "/" + state.Name;
end
```

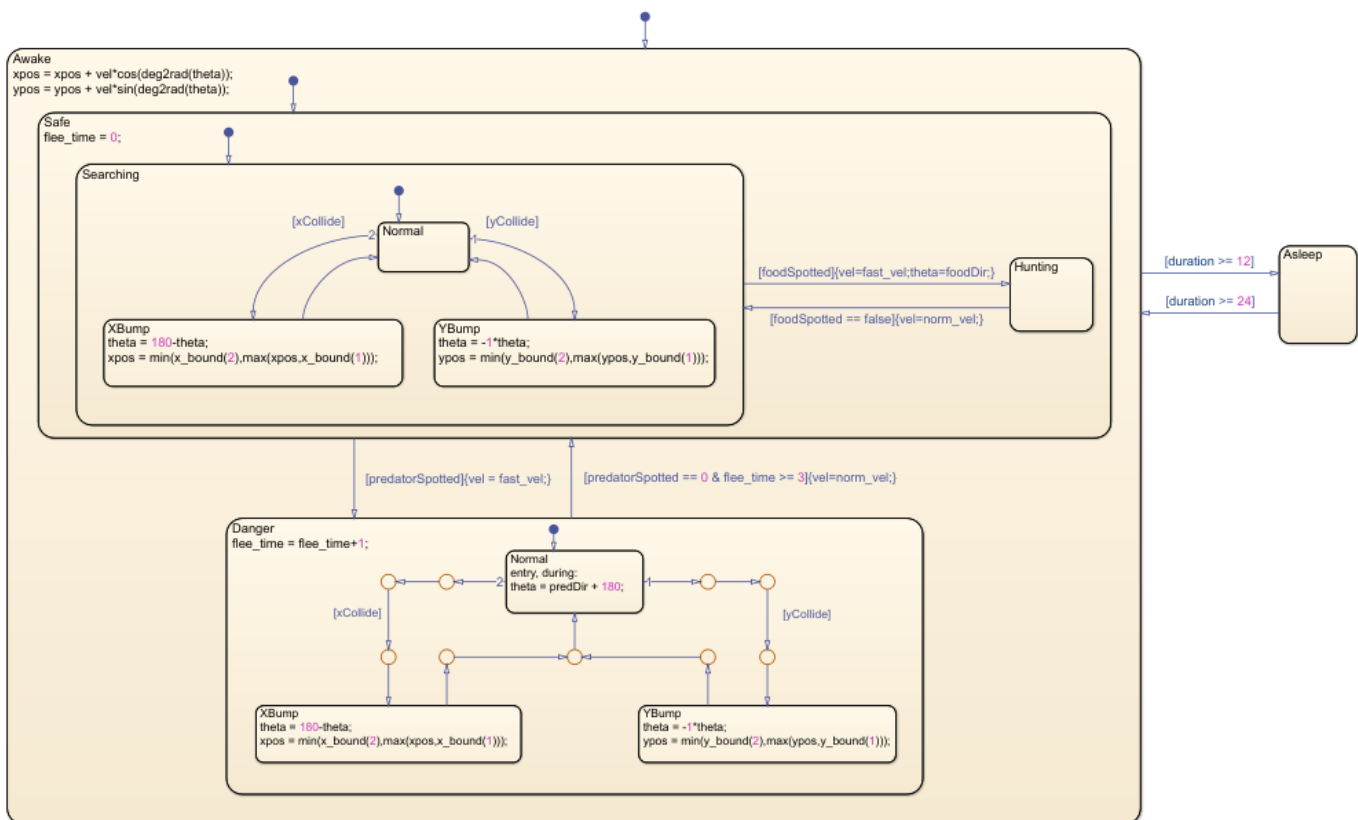
```
disp("The names of these states do not start with a capital letter:" + ...
    newline + misnamedStateNames)
```

The names of these states do not start with a capital letter:

```
* sfInsectExample/BehavioralLogic/Awake/Safe/searching
* sfInsectExample/BehavioralLogic/Awake/Danger/xBump
* sfInsectExample/BehavioralLogic/Awake/Danger/yBump
* sfInsectExample/BehavioralLogic/Awake/Safe/searching/normal
* sfInsectExample/BehavioralLogic/Awake/Safe/searching/xBump
* sfInsectExample/BehavioralLogic/Awake/Safe/searching/yBump
```

To replace the first letter of each identified state name with its uppercase equivalent, modify the Name property of each Stateflow.State object by calling the renameReferences function.

```
for i = 1:numMisnamedStates
    state = misnamedStates(i);
    newName = [upper(state.Name(1)) state.Name(2:end)];
    renameReferences(state,newName)
end
```



Simplify Deep Hierarchy of States

A chart with too many levels of nested states can be difficult to understand. If your chart requires a state hierarchy with more than three levels, you can reduce the complexity of your chart by creating subcharts. For more information, see “Encapsulate Modal Logic by Using Subcharts”.

To search your chart for deeply nested states, call `find` with the optional argument `-function` and specify a function handle that evaluates the helper function `getDepth`. This function computes the number of levels between a given state and the nearest ancestor chart or subchart. To view the code for this function, see [Get Depth of State](#) on page 1-32.

```
maxDepth = 3;
deepStates = find(ch, "-isa", "Stateflow.State", ...
    "-function", @(s)(getDepth(s) > maxDepth));

numDeepStates = numel(deepStates);
deepStateNames = "";

for i = 1:numDeepStates
    state = deepStates(i);
    deepStateNames = deepStateNames + ...
        newline + " * " + state.Path + "/" + state.Name;
end

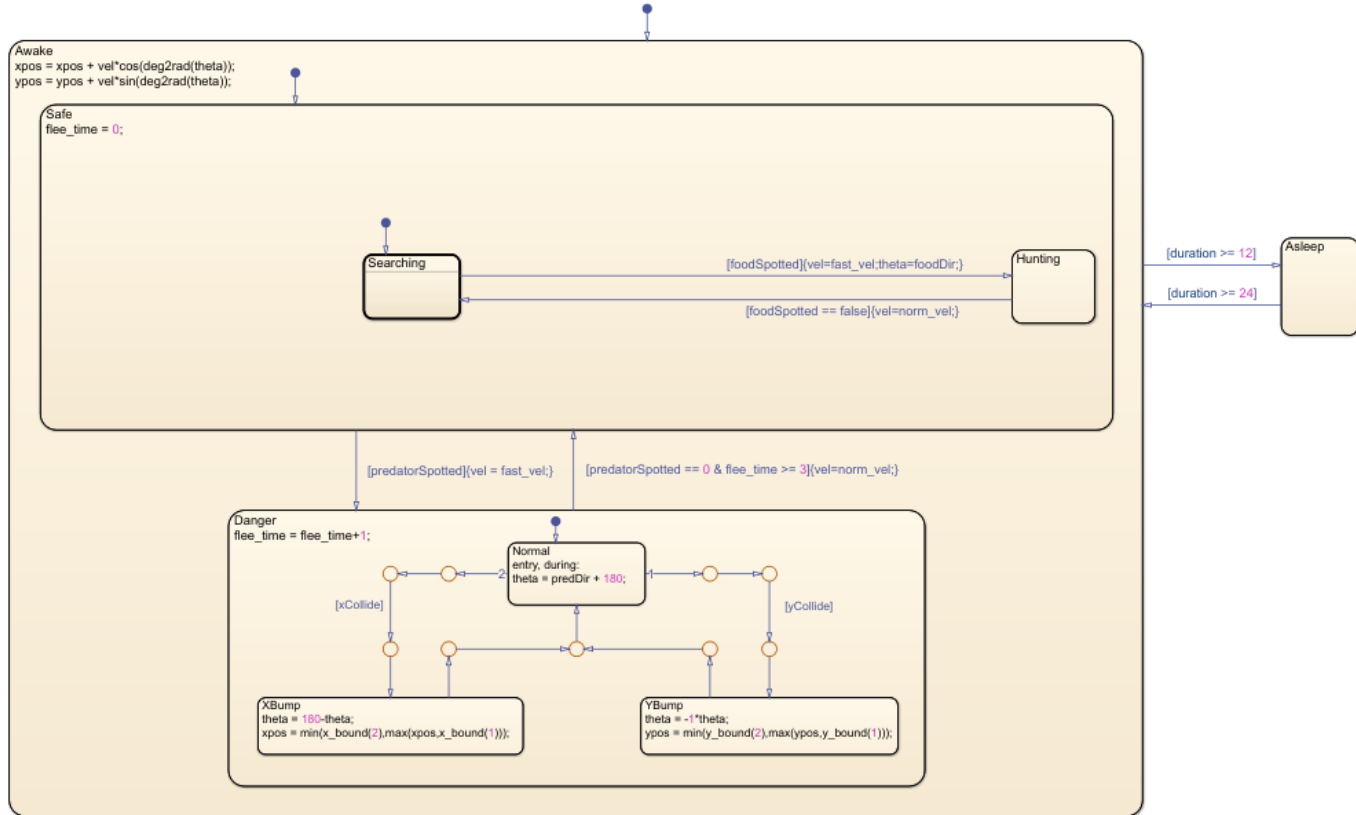
disp("These states occur at a depth greater than " + maxDepth + ":" + ...
    newline + deepStateNames)
```

These states occur at a depth greater than 3:

```
* sfInsectExample/BehavioralLogic/Awake/Safe/Searching/Normal
* sfInsectExample/BehavioralLogic/Awake/Safe/Searching/XBump
* sfInsectExample/BehavioralLogic/Awake/Safe/Searching/YBump
```

To simplify a chart with a deep hierarchy of states, convert states to subcharts by setting the `IsSubchart` property for the `Stateflow.State` objects to `true`. To automate this process for a large chart, the helper function `convertStatesToSubcharts` recursively finds nonleaf states at a given depth of the hierarchy and converts these states into subcharts. The function also disables the content preview and resizes the new subcharts. To view the code for this function, see [Convert States to Subcharts](#) on page 1-32.

```
convertStatesToSubcharts(ch,maxDepth)
```



Remove Superfluous Junctions

Long transition paths can increase the complexity of your charts. For example, to connect a source to a destination without any branching, using a single transition is simpler than using a sequence of transitions with multiple superfluous junctions.

To identify superfluous junctions, call `find` with the optional argument `-function` and specify a handle to the helper function `isSuperfluous`. This function checks whether a given junction satisfies these conditions:

- The junction has only one incoming transition and one outgoing transition.
- The outgoing transition continues in the same direction as the incoming transition.
- An action in the incoming transition does not precede a trigger or condition in the outgoing transition.
- Both transitions are not guarded by triggers.

To view the code for this function, see [Identify Superfluous Junctions](#) on page 1-33.

```
superfluousJunctions = find(ch,"-isa","Stateflow.Junction", ...
    "-function", @(j)(isSuperfluous(j)));
```

```
numSuperfluousJunctions = numel(superfluousJunctions);
superfluousJunctionIDs = "";
```

```
for i = 1:numSuperfluousJunctions
    junction = superfluousJunctions(i);
```

```

superfluousJunctionIDs = superfluousJunctionIDs + newline + ...
    " * Junction " + junction.SSIdNumber + " in " + junction.Path;
end

disp("These junctions are part of a transition path " + ...
    "that you can replace with a single transition:" + ...
    newline + superfluousJunctionIDs)

```

These junctions are part of a transition path that you can replace with a single transition:

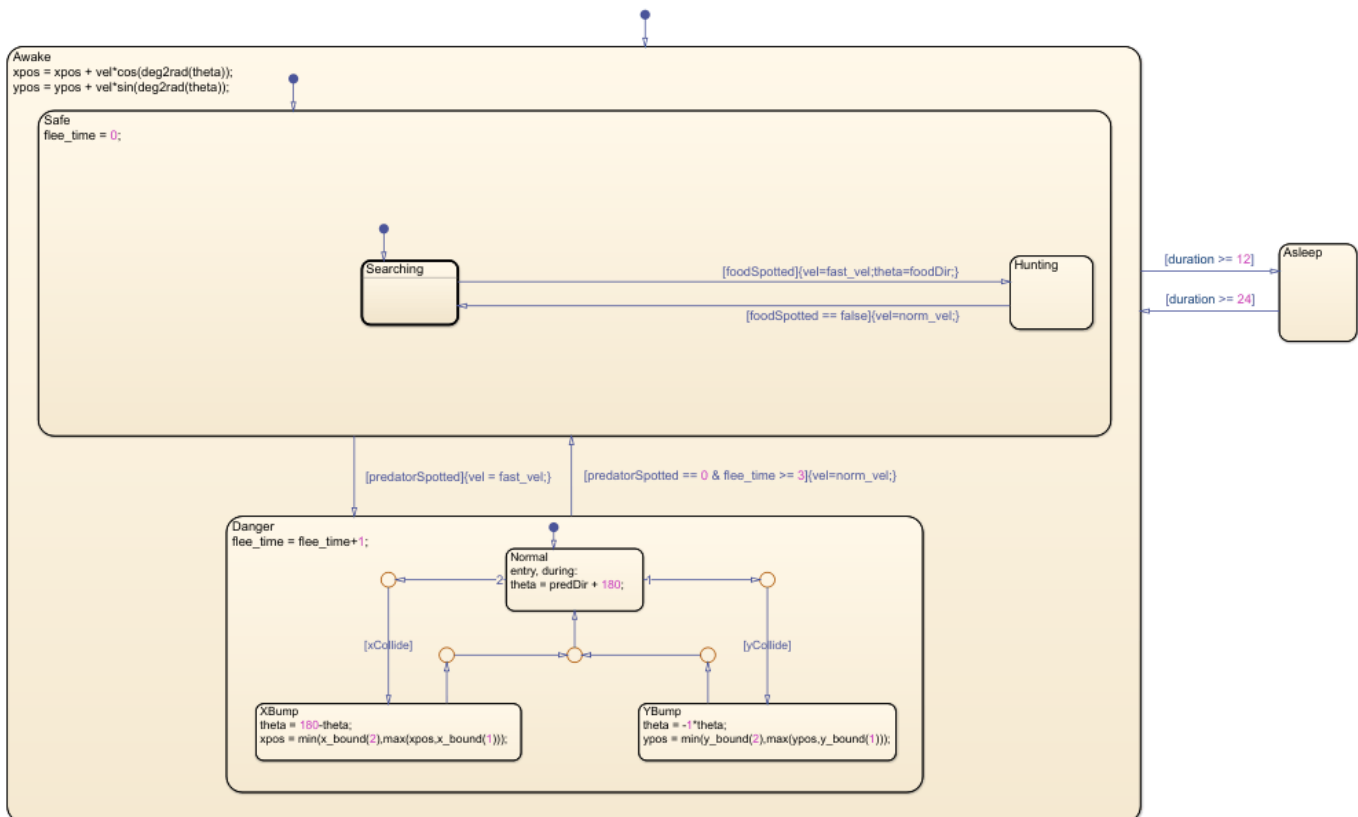
```

* Junction 161 in sfInsectExample/BehavioralLogic/Awake/Danger
* Junction 163 in sfInsectExample/BehavioralLogic/Awake/Danger
* Junction 153 in sfInsectExample/BehavioralLogic/Awake/Danger
* Junction 155 in sfInsectExample/BehavioralLogic/Awake/Danger

```

The helper function `removeJunctions` replaces the two transitions around each superfluous junction with a single transition. To preserve the geometry of the chart, the new transition starts at the same point as the original incoming transition and ends at the same point as the original outgoing transition. To view the code for this function, see [Remove Junctions](#) on page 1-33.

```
removeJunctions(superfluousJunctions)
```



Close the Model

Save the changes in a new model and close the model.

```
newModel = model+"Refactored";  
sfsave(model,newModel)  
close_system(newModel)
```

Helper Functions

Get Depth of State

This function returns the number of levels between a given state and the nearest ancestor chart or subchart.

```
function depth = getDepth(state)  
  
parent = getParent(state);  
  
if isa(parent,"Stateflow.Chart")  
    depth = 1;  
elseif parent.isSubchart  
    depth = 1;  
else  
    depth = getDepth(parent)+1;  
end  
end
```

Convert States to Subcharts

This function identifies nonleaf states at a given depth from the parent chart or subchart. The function converts these states into subcharts and disables the content preview of the new subcharts by modifying the `IsSubchart` and `ContentPreviewEnabled` properties for each `Stateflow.State` object. When possible, the function reduces the size of the subchart to a default of 90-by-60 while keeping the center of the subchart fixed. The process repeats recursively in each of the new subcharts.

```
function convertStatesToSubcharts(parent,maxDepth)  
  
statesToConvert = find(parent,"-isa","Stateflow.State", ...  
    "-function", @(s) (getDepth(s) == maxDepth), ...  
    "-function", @(s) (~isempty(getChildren(s))), ...  
    IsSubchart=false);  
  
for i = 1:numel(statesToConvert)  
    state = statesToConvert(i);  
    state.IsSubchart = true;  
    state.ContentPreviewEnabled = false;  
  
    reduceWidth(state,90)  
    reduceHeight(state,60)  
  
    convertStatesToSubcharts(state,maxDepth);  
end  
end  
  
function reduceWidth(state,newWidth)  
pos = state.Position;  
if pos(3) > newWidth  
    state.Position = [pos(1)+pos(3)/2-newWidth/2 pos(2) newWidth pos(4)];  
end  
end
```

```
function reduceHeight(state,newHeight)
pos = state.Position;
if pos(4) > 60
    state.Position = [pos(1) pos(2)+pos(4)/2-newHeight/2 pos(3) newHeight];
end
end
```

Identify Superfluous Junctions

This function checks whether a given junction satisfies these conditions:

- The junction has only one incoming transition and one outgoing transition.
- The outgoing transition continues in the same direction as the incoming transition.
- An action in the incoming transition does not precede a trigger or condition in the outgoing transition.
- Both transitions are not guarded by triggers.

```
function tf = isSuperfluous(junction)

transitionIn = sinkedTransitions(junction);
transitionOut = sourcedTransitions(junction);

tf = oneIncomingTransition && oneOutgoingTransition && ...
    transitionsContinueInSameDirection && ...
    noActionBeforeTriggerOrCondition && ...
    noDoubleTriggers;

function tf = oneIncomingTransition
    tf = (numel(transitionIn)==1);
end

function tf = oneOutgoingTransition
    tf = (numel(transitionOut)==1);
end

function tf = transitionsContinueInSameDirection
    tolerance = 1.5;
    theta = abs(transitionIn.DestinationOClock-transitionOut.SourceOClock);
    tf = (abs(theta-6) < tolerance);
end

function tf = noActionBeforeTriggerOrCondition
    tf = isempty(transitionIn.ConditionAction) || ...
        (isempty(transitionOut.Trigger) && isempty(transitionOut.Condition));
end

function tf = noDoubleTriggers
    tf = isempty(transitionIn.Trigger) || isempty(transitionOut.Trigger);
end
end
```

Remove Junctions

This function replaces the transitions around each superfluous junction with a single transition. To preserve the geometry of the chart, the new transition starts at the same point as the original

transition entering the junction and ends at the same point as the original transition exiting the junction. The function merges the labels strings of the original transitions according to these rules:

- Only one transition can have a nonempty trigger.
- Combine nonempty conditions by using the AND operator &&.
- Combine nonempty condition and transition actions by juxtaposition. If the first action does not end in a comma or semicolon, add a comma in between the actions.

For more information, see “Define Actions in a Transition”.

```
function removeJunctions(junctionsToRemove)

for i = 1:numel(junctionsToRemove)
    junction = junctionsToRemove(i);

    transitionIn = sinkedTransitions(junction);
    transitionOut = sourcedTransitions(junction);

    newSourceEndpoint = transitionIn.SourceEndpoint;
    newSourceOClock = transitionIn.SourceOClock;
    newDestinationEndpoint = transitionOut.DestinationEndpoint;
    newMidPoint = (newSourceEndpoint+newDestinationEndpoint)/2;
    newLabelString = mergeLabelStrings;

    transitionIn.Destination = transitionOut.Destination;
    transitionIn.SourceOClock = newSourceOClock;
    transitionIn.DestinationEndpoint = newDestinationEndpoint;
    transitionIn.MidPoint = newMidPoint;
    transitionIn.LabelString = newLabelString;
    transitionIn.LabelPosition = [newMidPoint 0 0];
    transitionIn.LabelPosition(1) = transitionIn.LabelPosition(1)-transitionIn.LabelPosition(3)/2;
    transitionIn.LabelPosition(2) = transitionIn.LabelPosition(2)-transitionIn.LabelPosition(4)/2;
    delete(junction)
    delete(transitionOut)
end

function label = mergeLabelStrings

    trigger1 = transitionIn.Trigger;
    trigger2 = transitionOut.Trigger;
    if isempty(trigger1)
        label = trigger2;
    elseif isempty(trigger2)
        label = trigger1;
    else
        error("Unable to merge transitions with multiple triggers " + ...
            trigger1 + " and " + trigger2 + ".")
    end

    condition1 = transitionIn.Condition;
    condition2 = transitionOut.Condition;
    if ~isempty(condition1) && ~isempty(condition2)
        label = label+"["+condition1+" && "+condition2+"]";
    elseif ~isempty(condition1)
        label = label+"["+condition1+"]";
    elseif ~isempty(condition2)
        label = label+"["+condition2+"]";
    end
end
```



```

end

action1 = transitionIn.ConditionAction;
action2 = transitionOut.ConditionAction;
if ~isempty(action1) && ~isempty(action2)
    if endsWith(action1, ";") || endsWith(action1, ",")
        label = label+"{"+action1+action2+"}";
    else
        label = label+"{"+action1+", "+action2+"}";
    end
elseif ~isempty(action1)
    label = label+"{"+action1+"}";
elseif ~isempty(action2)
    label = label+"{"+action2+"}";
end

transaction1 = transitionIn.TransitionAction;
transaction2 = transitionOut.TransitionAction;
if ~isempty(transaction1) && ~isempty(transaction2)
    if endsWith(transaction1, ";") || endsWith(action1, ",")
        label = label+"/{"+transaction1+transaction2+"}";
    else
        label = label+"/{"+transaction1+", "+transaction2+"}";
    end
elseif ~isempty(transaction1)
    label = label+"/{"+transaction1+"}";
elseif ~isempty(transaction2)
    label = label+"/{"+transaction2+"}";
end
end
end

```

See Also

Functions

find | renameReferences | getChildren | getParent | sinkedTransitions | sourcedTransitions

Objects

Stateflow.Chart | Stateflow.State | Stateflow.Transition | Stateflow.Junction

More About

- “Overview of the Stateflow API” on page 1-2
- “Access Objects in Your Stateflow Chart” on page 1-6
- “Encapsulate Modal Logic by Using Subcharts”
- “Summary of Stateflow API Objects and Properties” on page 1-36

Summary of Stateflow API Objects and Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. For example, to use the editor to change the position of a state, you click and drag the state. With the Stateflow API, you change the position of a state by modifying the `Position` property of the corresponding `Stateflow.State` object:

```
st.Position = [10 20 100 80];
```

For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

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

- **Property Name** — The name of the property. To access or set a property value, use its name in dot notation along with a Stateflow 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.

Stateflow.Annotation

Use `Stateflow.Annotation` objects to include descriptive comments in your chart. Annotations can contain any combination of:

- Text
- Images
- Equations using TeX commands
- Hyperlinks that open a website or perform MATLAB functions

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

Property Name	Access	Description
Alignment	RW	Alignment of the annotation text, specified as "LEFT", "CENTER", or "RIGHT".
AutoBackgroundColor	RW	Whether to use the default background color, specified as a numeric or logical 1 (true) or 0 (false). <ul style="list-style-type: none"> • <code>true</code> — Use the default color specified by the <code>ChartColor</code> property of the chart that contains the annotation. • <code>false</code> — Use the color specified by the <code>BackgroundColor</code> property of the annotation.

Property Name	Access	Description
AutoForegroundColor	RW	Whether to use the default foreground color, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). <ul style="list-style-type: none"> <code>true</code> — Use the default color specified by the <code>StateLabelColor</code> property of the chart that contains the annotation. <code>false</code> — Use the color specified by the <code>ForegroundColor</code> property of the annotation.
BackgroundColor	RW	Background color for the annotation, specified as a three-element numeric vector of the form <code>[red green blue]</code> that specifies the red, green, and blue values. Each element must be in the range between 0 and 1. This property applies only when the <code>AutoBackgroundColor</code> property is <code>false</code> .
Chart	RO	Chart that contains the annotation, specified as a <code>Stateflow.Chart</code> object.
ClickFcn	RW	Callback on click, specified as a string scalar or character vector. This callback contains MATLAB code to execute when you click the annotation.
DeleteFcn	RW	Callback at delete, specified as a string scalar or character vector. This callback contains MATLAB code to execute before you delete the annotation.
Description	RW	Description for the annotation, specified as a string scalar or character vector.
Document	RW	Document link for the annotation, specified as a string scalar or character vector.
DropShadow	RW	Whether to display a drop shadow around the annotation box, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>).
FixedHeight	RW	Whether to fix the height of the annotation box, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). <ul style="list-style-type: none"> <code>true</code> — Fixes the height of the annotation box and hides content that is longer than the box. <code>false</code> — Resizes the annotation box vertically as you add content.
FixedWidth	RW	Whether to fix the width of the annotation box, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). <ul style="list-style-type: none"> <code>true</code> — Fixes the width of the annotation box and wraps text that is longer than the box. <code>false</code> — Resizes the annotation box horizontally as you add content.


Property Name	Access	Description
Font	RW	Font for the annotation text, specified as a <code>Stateflow.NoteFont</code> object with these properties: <ul style="list-style-type: none"> • Name — Font name, specified as a character vector. This property is read-only. The <code>StateFont.Name</code> property of the chart that contains the annotation sets the value of this property. • Angle — Font angle, specified as "NORMAL" or "ITALIC". • Weight — Font weight, specified as "NORMAL" or "BOLD". • Size — Font size, specified as a scalar.
ForegroundColor	RW	Foreground color for the annotation, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1. This property applies only when the <code>AutoForegroundColor</code> property is false.
Id	RO	Unique identifier, specified as an integer scalar. Use this property to distinguish the annotation from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
InternalMargins	RW	Space between the text and the border of the annotation box, specified as a four-element numeric vector of the form [left top right bottom].
Interpretation	RW	Format of the annotation text, specified as "OFF", "RICH", or "TEX".
IsImage	RO	Whether the annotation contains an image, specified as a numeric or logical 1 (true) or 0 (false).
LoadFcn	RW	Callback at model load, specified as a string scalar or character vector. This callback contains MATLAB code to execute when you load the model that contains the annotation.
Machine	RO	Machine that contains the annotation, specified as a <code>Stateflow.Machine</code> object.
Path	RO	Location of the parent of the annotation in the model hierarchy, specified as a character vector.
PlainText	RO	Annotation text without formatting, specified as a character vector.
Position	RW	Position and size of annotation box, specified as a four-element numeric vector of the form [left top width height].
Subviewer	RO	Subviewer for the annotation, specified as a <code>Stateflow.Chart</code> , <code>Stateflow.State</code> , <code>Stateflow.Box</code> , or <code>Stateflow.Function</code> object. The subviewer is the chart or subchart where you can graphically view the annotation.
Tag	RW	User-defined tag for the annotation, specified as data of any type.
Text	RW	Text for the annotation, specified as a string scalar or character vector.
UseDisplayTextAsClickCallback	RW	Whether to use the annotation text as a callback, specified as a numeric or logical 1 (true) or 0 (false). When this property is enabled, the contents of the <code>Text</code> property is used as the callback when you click the annotation.

Stateflow.AtomicBox

Use `Stateflow.AtomicBox` objects to encapsulate graphical, truth table, MATLAB, and Simulink functions in a separate namespace. Atomic boxes allow for:

- Faster simulation after making small changes to a function in a chart with many states or levels of hierarchy
- Reuse of the same functions across multiple charts and models
- Ease of team development for people working on different parts of the same chart
- Manual inspection of generated code for a specific function in a chart

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

Property Name	Access	Description
BadIntersection	RO	Whether the atomic box graphically intersects a box, state, or function, specified as a numeric or logical 1 (true) or 0 (false).
Chart	RO	Chart that contains the atomic box, specified as a <code>Stateflow.Chart</code> object.
CommentText	RW	Comment text for the atomic box, specified as a string scalar or character vector. This property applies only when the <code>IsExplicitlyCommented</code> property is <code>true</code> . In the Stateflow Editor, when you point to the comment badge  on the atomic box, the text appears as a tooltip. When you set the <code>IsExplicitlyCommented</code> property to <code>false</code> , the value of <code>CommentText</code> reverts to "".
ContentPreviewEnabled	RW	Whether to display a preview of the atomic box contents, specified as a numeric or logical 1 (true) or 0 (false).
Description	RW	Description for the atomic box, specified as a string scalar or character vector.
Document	RW	Document link for the atomic box, specified as a string scalar or character vector.
FontSize	RW	Font size for the atomic box label, specified as a scalar. The <code>StateFont.Size</code> property of the chart that contains the atomic box sets the initial value of this property.
Id	RO	Unique identifier, specified as an integer scalar. Unlike <code>SSIdNumber</code> , the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
IsCommented	RO	Whether the atomic box is commented out, specified as a numeric or logical 1 (true) or 0 (false). This property is <code>true</code> when either <code>IsExplicitlyCommented</code> or <code>IsImplicitlyCommented</code> is <code>true</code> .
IsExplicitlyCommented	RW	Whether to comment out the atomic box, specified as a numeric or logical 1 (true) or 0 (false). Setting this property to <code>true</code> is equivalent to right-clicking the atomic box and selecting Comment Out . For more information, see “Comment Out Objects in a Stateflow Chart”.

Property Name	Access	Description
IsImplicitlyCommented	RO	Whether the atomic box is implicitly commented out, specified as a numeric or logical 1 (true) or 0 (false). The atomic box is implicitly commented out when you explicitly comment out an object that contains it. If the atomic box is contained in an atomic subchart or another atomic box, this property is false unless the explicitly commented object is also contained in the atomic subchart or atomic box.
IsLink	RO	Whether the atomic box is a library link, specified as a numeric or logical 1 (true) or 0 (false).
LabelString	RW	Label for the atomic box, specified as a string scalar or character vector.
Machine	RO	Machine that contains the atomic box, specified as a <code>Stateflow.Machine</code> object.
Name	RW	Name of the atomic box, specified as a string scalar or character vector.
Path	RO	Location of the parent of the atomic box in the model hierarchy, specified as a character vector.
Position	RW	Position and size of the atomic box, specified as a four-element numeric vector of the form [left top width height].
SSIdNumber	RO	Session-independent identifier, specified as an integer scalar. Use this property to distinguish the atomic box from other objects in the model.
Subchart	RO	Contents of the atomic box, specified as a <code>Stateflow.Chart</code> object. Use this object to add children, such as states and transitions, to the atomic box.
Subviewer	RO	Subviewer for the atomic box, specified as a <code>Stateflow.Chart</code> , <code>Stateflow.State</code> , <code>Stateflow.Box</code> , or <code>Stateflow.Function</code> object. The subviewer is the chart or subchart where you can graphically view the atomic box.
Tag	RW	User-defined tag for the atomic box, specified as data of any type.


Stateflow.AtomicSubchart

Use `Stateflow.AtomicSubchart` objects to create independent subcomponents in a Stateflow chart. Atomic subcharts allow for:

- Reuse of the same state or subchart across multiple charts and models
- Faster simulation after making small changes to a chart with many states or levels of hierarchy
- Ease of team development when multiple people are working on different parts of the same chart
- Manual inspection of generated code for a specific state or subchart in a chart

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

Property Name	Access	Description
ArrowSize	RW	Size of incoming transition arrows, specified as a scalar.
BadIntersection	RO	Whether the atomic subchart graphically intersects a box, state, or function, specified as a numeric or logical 1 (true) or 0 (false).

Property Name	Access	Description
Chart	RO	Chart that contains the atomic subchart, specified as a <code>Stateflow.Chart</code> object.
CommentText	RW	Comment text for the atomic subchart, specified as a string scalar or character vector. This property applies only when the <code>IsExplicitlyCommented</code> property is <code>true</code> . In the Stateflow Editor, when you point to the comment badge  on the atomic subchart, the text appears as a tooltip. When you set the <code>IsExplicitlyCommented</code> property to <code>false</code> , the value of <code>CommentText</code> reverts to "".
ContentPreviewEnabled	RW	Whether to display a preview of the atomic subchart contents, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>).
Debug	RW	Debugger properties for the state, atomic subchart, or Simulink based state, specified as a <code>Stateflow.StateDebug</code> object with these properties: <ul style="list-style-type: none"> • OnEntry — Whether to set the On State Entry breakpoint, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • OnDuring — Whether to set the During State breakpoint, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • OnExit — Whether to set the On State Exit breakpoint, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see “Set Breakpoints to Debug Charts”.
Description	RW	Description for the atomic subchart, specified as a string scalar or character vector.
Document	RW	Document link for the atomic subchart, specified as a string scalar or character vector.
ExecutionOrder	RW	Execution order for the atomic subchart in parallel (AND) decomposition, specified as an integer scalar. This property applies only when both of these conditions are satisfied: <ul style="list-style-type: none"> • The <code>Type</code> property of the atomic subchart is "AND". • The <code>UserSpecifiedStateTransitionExecutionOrder</code> property of the chart that contains the atomic subchart is <code>true</code>.
FontSize	RW	Font size for the atomic subchart label, specified as a scalar. The <code>StateFont.Size</code> property of the chart that contains the atomic subchart sets the initial value of this property.
HasOutputData	RW	Whether to create an active state data output port for the atomic subchart, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see “Monitor State Activity Through Active State Data”.
Id	RO	Unique identifier, specified as an integer scalar. Unlike <code>SSIdNumber</code> , the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
IsCommented	RO	Whether the atomic subchart is commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property is <code>true</code> when either <code>IsExplicitlyCommented</code> or <code>IsImplicitlyCommented</code> is <code>true</code> .


Property Name	Access	Description
IsExplicitlyCommented	RW	Whether to comment out the atomic subchart, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). Setting this property to <code>true</code> is equivalent to right-clicking the atomic subchart and selecting Comment Out . For more information, see “Comment Out Objects in a Stateflow Chart”.
IsImplicitlyCommented	RO	Whether the atomic subchart is implicitly commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). The atomic subchart is implicitly commented out when you explicitly comment out an object that contains it. If the atomic subchart is contained in another atomic subchart, this property is <code>false</code> unless the explicitly commented object is also contained in the atomic subchart.
IsLink	RO	Whether the atomic subchart is a library link, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>).
LabelString	RW	Label for the atomic subchart, specified as a string scalar or character vector.
LoggingInfo	RW	Signal logging properties for the atomic subchart, specified as a <code>Stateflow.SigLoggingInfo</code> object with these properties: <ul style="list-style-type: none"> • DataLogging — Whether to enable signal logging, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • DecimateData — Whether to limit the amount of logged data, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • Decimation — Decimation interval, specified as an integer scalar. This property applies only when the <code>DecimateData</code> property is <code>true</code>. • LimitDataPoints — Whether to limit the number of data points to log, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • MaxPoints — Maximum number of data points to log, specified as an integer scalar. This property applies only when the <code>LimitDataPoints</code> property is <code>true</code>. • NameMode — Source of the signal name, specified as “SignalName” or “Custom”. • LoggingName — Custom signal name, specified as a string scalar or character vector. This property applies only when the <code>NameMode</code> property is “Custom”. <p>Signal logging saves the self activity of the atomic subchart to the MATLAB workspace during simulation. For more information, see “Log Simulation Output for States and Data”.</p>
Machine	RO	Machine that contains the atomic subchart, specified as a <code>Stateflow.Machine</code> object.
Name	RW	Name of the atomic subchart, specified as a string scalar or character vector.
OutputData	RO	Active state data object for the atomic subchart, specified as a <code>Stateflow.Data</code> object. This property applies only when the <code>HasOutputData</code> property for the atomic subchart is <code>true</code> .

Property Name	Access	Description
OutputMonitoringMode	RW	Monitoring mode for the active state output data, specified as a string scalar or character vector. For atomic subcharts, the only option is "SelfActivity".
OutputPortName	RW	Name of the active state data object for the atomic subchart, specified as a string scalar or character vector. This property applies only when the HasOutputData property for the atomic subchart is true.
Path	RO	Location of the parent of the atomic subchart in the model hierarchy, specified as a character vector.
Position	RW	Position and size of the atomic subchart, specified as a four-element numeric vector of the form [left top width height].
SSIdNumber	RO	Session-independent identifier, specified as an integer scalar. Use this property to distinguish the atomic subchart from other objects in the model.
Subchart	RO	Contents of the atomic subchart, specified as a Stateflow.Chart object. Use this object to add children, such as states and transitions, to the atomic subchart. For more information, see "Add Exit Port and Junction to Atomic Subchart" on page 2-20.
Subviewer	RO	Subviewer for the atomic subchart, specified as a Stateflow.Chart, Stateflow.State, or Stateflow.Box object. The subviewer is the chart or subchart where you can graphically view the atomic subchart.
Tag	RW	User-defined tag for the atomic subchart, specified as data of any type.
TestPoint	RW	Whether to set the atomic subchart as a test point, specified as a numeric or logical 1 (true) or 0 (false). You can monitor testpoints with a floating scope during simulation. You can also log test point values to the MATLAB workspace. For more information, see "Monitor Test Points in Stateflow Charts".
Type	RO	Decomposition of sibling states, specified as 'AND' or 'OR'. The atomic subchart inherits this property from the Decomposition property of its parent state or chart.

Stateflow.Box

Use Stateflow.Box objects to organize objects such as functions and states in your chart. You can also use a box to encapsulate states and functions in a separate namespace. For more information, see "Group Chart Objects by Using Boxes".

Property Name	Access	Description
BadIntersection	RO	Whether the box graphically intersects a box, state, or function, specified as a numeric or logical 1 (true) or 0 (false).
Chart	RO	Chart that contains the box, specified as a Stateflow.Chart object.

Property Name	Access	Description
CommentText	RW	Comment text for the box, specified as a string scalar or character vector. This property applies only when the <code>IsExplicitlyCommented</code> property is <code>true</code> . In the Stateflow Editor, when you point to the comment badge  on the box, the text appears as a tooltip. When you set the <code>IsExplicitlyCommented</code> property to <code>false</code> , the value of <code>CommentText</code> reverts to "".
ContentPreviewEnabled	RW	Whether to display a preview of the box contents, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property applies only when the <code>IsSubchart</code> property is <code>true</code> .
Description	RW	Description for the box, specified as a string scalar or character vector.
Document	RW	Document link for the box, specified as a string scalar or character vector.
ExecutionOrder	RW	Execution order for the substates of the box in parallel (AND) decomposition, specified as an integer scalar. This property applies only when both of these conditions are satisfied: <ul style="list-style-type: none"> The <code>Decomposition</code> property of the parent state or chart is "PARALLEL_AND". The <code>UserSpecifiedStateTransitionExecutionOrder</code> property of the chart that contains the box is <code>true</code>.
FontSize	RW	Font size for the box label, specified as a scalar. The <code>StateFont.Size</code> property of the chart that contains the box sets the initial value of this property.
Id	RO	Unique identifier, specified as an integer scalar. Use this property to distinguish the box from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
IsCommented	RO	Whether the box is commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property is <code>true</code> when either <code>IsExplicitlyCommented</code> or <code>IsImplicitlyCommented</code> is <code>true</code> .
IsExplicitlyCommented	RW	Whether to comment out the box, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). Setting this property to <code>true</code> is equivalent to right-clicking the box and selecting Comment Out . For more information, see "Comment Out Objects in a Stateflow Chart".
IsGrouped	RW	Whether the box is a grouped box, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). When you copy and paste a grouped box, you copy not only the box but all of its contents. For more information, see "Copy and Paste by Grouping" on page 2-28.
IsImplicitlyCommented	RO	Whether the box is implicitly commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). The box is implicitly commented out when you explicitly comment out an object that contains it. If the box is contained in an atomic subchart or an atomic box, this property is <code>false</code> unless the explicitly commented object is also contained in the atomic subchart or atomic box.
IsSubchart	RW	Whether the box is a subchart, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>).

Property Name	Access	Description
LabelString	RW	Label for the box, specified as a string scalar or character vector.
Machine	RO	Machine that contains the box, specified as a <code>Stateflow.Machine</code> object.
Name	RW	Name of the box, specified as a string scalar or character vector.
Path	RO	Location of the parent of the box in the model hierarchy, specified as a character vector.
Position	RW	Position and size of the box, specified as a four-element numeric vector of the form [left top width height].
Subviewer	RO	Subviewer for the box, specified as a <code>Stateflow.Chart</code> , <code>Stateflow.State</code> , <code>Stateflow.Box</code> , or <code>Stateflow.Function</code> object. The subviewer is the chart or subchart where you can graphically view the box.
Tag	RW	User-defined tag for the box, specified as data of any type.

Stateflow.Chart

Use a `Stateflow.Chart` object to create a graphical representation of a finite state machine based on a state transition diagram. In a Chart block, states and transitions form the basic building blocks of a sequential logic system. States correspond to operating modes and transitions represent pathways between states. For more information, see “Model Finite State Machines by Using Stateflow Charts” and “Create Charts by Using the Stateflow API” on page 1-19.

Property Name	Access	Description
ActionLanguage	RW	Action language used to program the chart, specified as "MATLAB" or "C". For more information, see “Differences Between MATLAB and C as Action Language Syntax”.
AllowGlobalAccessToExportedFunctions	RW	Whether exported functions from the chart are globally visible in the Simulink model, specified as a numeric or logical 1 (true) or 0 (false). When this property is enabled, blocks throughout the model can call functions exported from the chart without using qualified notation. This property applies only when the <code>ExportChartFunctions</code> property for the chart is true.
ChartColor	RW	Background color for the chart, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.
ChartUpdate	RW	Activation method for the chart, specified as "CONTINUOUS", "DISCRETE", or "INHERITED". For more information, see “Update Method”.
Debug	RW	Debugger properties for the chart, specified as a <code>Stateflow.ChartDebug</code> object with this property: <ul style="list-style-type: none"> • Breakpoints.OnEntry — Whether to set the On Chart Entry breakpoint, specified as a numeric or logical 1 (true) or 0 (false). For more information, see “Set Breakpoints to Debug Charts”.

Property Name	Access	Description
Decomposition	RW	Decomposition of substates at the top level of containment in the chart, specified as "EXCLUSIVE_OR" or "PARALLEL_AND". For more information, see "Define Exclusive and Parallel Modes by Using State Decomposition".
Description	RW	Description for the chart, specified as a string scalar or character vector.
Dirty	RW	Whether the chart has changed after being opened or saved, specified as a numeric or logical 1 (true) or 0 (false).
DoNotAutogenerateEnum	RW	Whether to define the enumerated data type for the active state data output manually, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Define State Activity Enumeration Type".
Document	RW	Document link for the chart, specified as a string scalar or character vector.
Editor	RO	Editor for the chart, specified as a <code>Stateflow.Editor</code> object. You can use this object to control the position, size, and magnification level of the Stateflow Editor window.
Em1DefaultFimath	RW	Default fimath properties for the chart, specified as one of these values: <ul style="list-style-type: none"> "Same as MATLAB Default" — Use the same fimath properties as the current default fimath object. "Other:UserSpecified" — Use the <code>InputFimath</code> property to specify the default fimath object. <p>This property applies only when the <code>ActionLanguage</code> property of the chart is "MATLAB".</p>
EnableBitOps	RW	Whether to use bit operations in state and transition actions in the chart, specified as a numeric or logical 1 (true) or 0 (false). This property applies only to charts that use C as the action language. For more information, see "Enable C-bit operations".
EnableNonTerminalStates	RW	Whether to enable super step semantics for the chart, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Super Step Semantics".
EnableZeroCrossings	RW	Whether to enable zero-crossing detection on state transitions in the chart, specified as a numeric or logical 1 (true) or 0 (false). This property applies only when the <code>ChartUpdate</code> property for the chart is set to "CONTINUOUS". For more information, see "Disable Zero-Crossing Detection".
EnumTypeName	RW	Name of the enumerated data type for the active state data object for the chart, specified as a string scalar or character vector. For more information, see "Enum Name".
ExecuteAtInitialization	RW	Whether to initialize the state configuration of the chart at time zero instead of at the first input event, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Execution of a Chart at Initialization".
ExportChartFunctions	RW	Whether to export chart-level functions to other blocks in the Simulink model, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Export Stateflow Functions for Reuse".

Property Name	Access	Description
GeneratePreprocessorConditionals	RW	Whether the generated code includes a preprocessor conditional statement for the variant conditions in the chart, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property applies only when generating code with Embedded Coder®. For more information, see “Control Indicator Lamp Dimmer Using Variant Conditions”.
HasOutputData	RW	Whether to create an active state data output port for the chart, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see “Monitor State Activity Through Active State Data”.
Iced	RO	Whether the chart is locked, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property is equivalent to the property <code>Locked</code> , but is used internally to prevent changes in the chart during simulation.
Id	RO	Unique identifier, specified as an integer scalar. Use this property to distinguish the chart from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
InitializeOutput	RW	Whether to initialize the output data every time the chart wakes up, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see “Initialize outputs every time chart wakes up”.
InputFimath	RW	Default fimath object, specified as a string scalar or character vector. When the <code>EmlDefaultFimath</code> property for the chart is “Other:UserSpecified”, you can use this property to: <ul style="list-style-type: none"> • 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 to charts that use MATLAB as the action language.
JunctionColor	RW	Color for junctions in the chart, specified as a three-element numeric vector of the form <code>[red green blue]</code> that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.
Locked	RW	Whether the chart is locked, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). Enable this property to prevent changes in the chart.
Machine	RO	Machine that contains the chart, specified as a <code>Stateflow.Machine</code> object.
Name	RW	Name of the chart, specified as a string scalar or character vector.
NonTerminalMaxCounts	RW	Maximum number of transitions the chart can take in one super step, specified as an integer scalar. This property applies only when the <code>EnableNonTerminalStates</code> property for the chart is <code>true</code> .

Property Name	Access	Description
NonTerminalUnstableBehavior	RW	<p>Behavior if a super step for the chart exceeds the maximum number of transitions specified in the <code>NonTerminalMaxCounts</code> property before reaching a stable state, specified as one of these values:</p> <ul style="list-style-type: none"> "Proceed" — The chart goes to sleep with the last active state configuration. "Throw Error" — The chart generates an error. <p>This property applies only when the <code>EnableNonTerminalStates</code> property for the chart is <code>true</code>.</p>
OutputData	RO	Active state data object for the chart, specified as a <code>Stateflow.Data</code> object. This property applies only when the <code>HasOutputData</code> property for the chart is <code>true</code> .
OutputMonitoringMode	RW	Monitoring mode for the active state output data, specified as "ChildActivity" or "LeafStateActivity".
OutputPortName	RW	Name of the active state data object for the chart, specified as a string scalar or character vector. This property applies only when the <code>HasOutputData</code> property for the chart is <code>true</code> .
Path	RO	Location of the chart in the model hierarchy, specified as a character vector.
SampleTime	RW	Sample time for activating the chart, specified as a string scalar or character vector. This property applies only when the <code>ChartUpdate</code> property for the chart is "DISCRETE".
SaturateOnIntegerOverflow	RW	Whether the data in the chart saturates on integer overflow, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). When this property is disabled, the data in the chart wraps on integer overflow. For more information, see "Handle Integer Overflow for Chart Data".
StateColor	RW	Color for the boxes, functions, and states in the chart, specified as a three-element numeric vector of the form <code>[red green blue]</code> that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.
StateFont	RW	Font for the box, function, and state labels in the chart, specified as a <code>Stateflow.StateFont</code> object with these properties: <ul style="list-style-type: none"> Name — Font name, specified as a string scalar or character vector. This property also determines the font for annotations in the chart. Angle — Font angle, specified as "NORMAL" or "ITALIC". Weight — Font weight, specified as "NORMAL" or "BOLD". Size — Default font size for new boxes, functions, and states, specified as a scalar. This property also determines the default font size for new annotations in the chart.
StateLabelColor	RW	Color for the box, function, and state labels in the chart, specified as a three-element numeric vector of the form <code>[red green blue]</code> that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.

Property Name	Access	Description
StateMachineType	RW	State machine semantics implemented by the chart, specified as "Classic", "Mealy", or "Moore". For more information, see "Overview of Mealy and Moore Machines".
StatesWhenEnabling	RW	Behavior of the states when a function-call input event reenables the chart, specified as one of these values: <ul style="list-style-type: none"> " " — The chart does not contain function-call input events. "held" — The chart maintains the most recent values of the states. "reset" — The chart reverts to the initial conditions of the states. For more information, see "Control States in Charts Enabled by Function-Call Input Events".
SupportVariableSizing	RW	Whether the chart supports variable-size data, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Declare Variable-Size Data in Stateflow Charts".
Tag	RW	User-defined tag for the chart, specified as data of any type.
TransitionColor	RW	Color for transitions in the chart, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.
TransitionFont	RW	Font for the transition labels in the chart, specified as a Stateflow.TransFont object with these properties: <ul style="list-style-type: none"> Name — Font name, specified as a string scalar or character vector. Angle — Font angle, specified as "NORMAL" or "ITALIC". Weight — Font weight, specified as "NORMAL" or "BOLD". Size — Default font size for new transitions, specified as a scalar.
TransitionLabelColor	RW	Color for the transition labels in the chart, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.
TreatAsFi	RW	Inherited Simulink signals to treat as Fixed-Point Designer™ fi objects, specified as one of these values: <ul style="list-style-type: none"> "Fixed-point" — The chart treats all fixed-point inputs as fi objects. "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.

Property Name	Access	Description
TreatDimensionOfLengthOneAsFixedSize	RW	Whether chart output data with at least one dimension of length 1 are fixed size, specified as a numeric or logical 0 (<code>false</code>) or 1 (<code>true</code>). When this property is <code>true</code> , the object sets data that are variable size in the chart with a dimension of 1 to fixed size. When this property is <code>false</code> , data in the chart that has the Variable size property enabled are always variable size. Prior to R2023a, the object treats data with at least one dimension of length 1 as fixed size. This property only affects output data that have the Variable size property enabled. See “Variable size” (Simulink).
UserSpecifiedStateTransitionExecutionOrder	RW	Whether to use explicit ordering of parallel states and transitions, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property applies only to charts that use C as the action language. For more information, see “User-specified state/transition execution order”.
Visible	RW	Whether the Stateflow Editor window is displaying the chart, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>).

Stateflow.Clipboard

Use the `Stateflow.Clipboard` object to copy and paste graphical and nongraphical objects within the same chart, between charts in the same Simulink model, or between charts in different models.

`Stateflow.Clipboard` objects do not have any properties.

Stateflow.Data

Use `Stateflow.Data` objects to store values that are visible at a specific level of the Stateflow hierarchy. For more information, see “Add Stateflow Data” and “Set Data Properties”.

Property Name	Access	Description
CompiledSize	RO	Data size as determined by the compiler, specified as a character vector.
CompiledType	RO	Data type as determined by the compiler, specified as a character vector.

Property Name	Access	Description
DataType	RW	<p>Type of the data object, specified as a string scalar or character vector that depends on the <code>Props.Type.Method</code> property of the data object:</p> <ul style="list-style-type: none"> If the <code>Props.Type.Method</code> property of the data object is "Inherit", the value of this property is "Inherit: From definition in chart" for local data and "Inherit: Same as Simulink" for input, output, and parameter data. If the <code>Props.Type.Method</code> property of the data object is "Built-in", you can specify this property with one of these options: <ul style="list-style-type: none"> "double" "single" "int8" "int16" "int32" "int64" "uint8" "uint16" "uint32" "uint64" "boolean" "string" "ml" (Supported only in charts that use C as the action language) Otherwise, the <code>Props.Type</code> properties of the data object determine the value of this property. <p>For more information, see the section Add Data on page 1-0 in "Create Charts by Using the Stateflow API" on page 1-19.</p>
Debug	RW	<p>Debugger properties for the data object, specified as a <code>Stateflow.DataDebug</code> object with this property:</p> <ul style="list-style-type: none"> Watch — Whether to track the value of the data object in the Breakpoints and Watch window, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see "View Data in the Breakpoints and Watch Window".
Description	RW	Description for the data object, specified as a string scalar or character vector.
Document	RW	Document link for the data object, specified as a string scalar or character vector.
Id	RO	Unique identifier, specified as an integer scalar. Unlike <code>SSIdNumber</code> , the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Property Name	Access	Description
InitializeMethod	RW	<p>Method for initializing the value of the data object, specified as a string scalar or character vector that depends on the scope of the data:</p> <ul style="list-style-type: none"> • For local and output data, use "Expression" or "Parameter". • For constant data, use "Expression". • For input data, parameters, and data store memory, use "Not Needed". <p>To specify the initial value of the data object, use the <code>Props.InitialValue</code> property.</p> <p>This property is equivalent to the Initial Value drop-down list in the Model Explorer and the Data properties dialog box. For more information, see "Initial value".</p>
LoggingInfo	RW	<p>Signal logging properties for the data object, specified as a <code>Stateflow.SigLoggingInfo</code> object with these properties:</p> <ul style="list-style-type: none"> • DataLogging — Whether to enable signal logging, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • DecimateData — Whether to limit the amount of logged data, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • Decimation — Decimation interval, specified as an integer scalar. This property applies only when the <code>DecimateData</code> property is <code>true</code>. • LimitDataPoints — Whether to limit the number of data points to log, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • MaxPoints — Maximum number of data points to log, specified as an integer scalar. This property applies only when the <code>LimitDataPoints</code> property is <code>true</code>. • NameMode — Source of the signal name, specified as "SignalName" or "Custom". • LoggingName — Custom signal name, specified as a string scalar or character vector. This property applies only when the <code>NameMode</code> property is "Custom". <p>Signal logging saves the values of the data object to the MATLAB workspace during simulation. For more information, see "Log Simulation Output for States and Data".</p>
Machine	RO	Machine that contains the data object, specified as a <code>Stateflow.Machine</code> object.
Name	RW	Name of the data object, specified as a string scalar or character vector.
OutputState	RO	State or chart monitored by the data object, specified as an empty array or a <code>Stateflow.AtomicSubchart</code> , <code>Stateflow.Chart</code> , <code>Stateflow.SimulinkBasedState</code> , <code>Stateflow.State</code> , or <code>Stateflow.StateTransitionTableChart</code> object. For more information, see "Monitor State Activity Through Active State Data".

Property Name	Access	Description
Path	RO	Location of the parent of the data object in the model hierarchy, specified as a character vector.
Port	RW	Port index for the data object, specified as an integer scalar. This property applies only to input and output data. For more information, see “Port”.

Property Name	Access	Description
Props	RW	<p>Data specification properties, specified as a <code>Stateflow.DataProps</code> object with these properties:</p> <ul style="list-style-type: none"> • Type.Method — Method for setting the type of the data object, specified as a string scalar or character vector. <ul style="list-style-type: none"> • For local, input, output, or parameter data, use "Inherited", "Built-in", "Bus Object", "Enumerated", "Expression", or "Fixed point". • For constant data, use "Built-in", "Expression", or "Fixed point". • For data store memory data, use "Inherited". <p>This property is equivalent to the Mode field of the Data Type Assistant in the Model Explorer and the Data properties dialog box. For more information, see "Specify Type of Stateflow Data".</p> <ul style="list-style-type: none"> • Type.BusObject — Name of the <code>Simulink.Bus</code> object that defines the data object, specified as a string scalar or character vector. This property applies only when the <code>Type.Method</code> property of the data object is "Bus Object". For more information, see "Access Bus Signals Through Stateflow Structures". • Type.EnumType — Name of the enumerated type that defines the data object, specified as a string scalar or character vector. This property applies only when the <code>Type.Method</code> property of the data object is "Enumerated". For more information, see "Reference Values by Name by Using Enumerated Data". • Type.Expression — Expression that evaluates to the data type of the data object, specified as a string scalar or character vector. This property applies only when the <code>Type.Method</code> property of the data object is "Expression". For more information, see "Specify Data Properties by Using MATLAB Expressions". • Type.Signed — Signedness, specified as a numeric or logical 1 (true) or 0 (false). This property applies only when the <code>Type.Method</code> property of the data object is "Fixed point". For more information, see "Fixed-Point Data in Stateflow Charts". • Type.WordLength — Word length, in bits, specified as a string scalar or character vector. This property applies only when the <code>Type.Method</code> property of the data object is "Fixed point". For more information, see "Fixed-Point Data in Stateflow Charts". • Type.Fixpt.ScalingMode — Method for scaling the fixed-point data object, specified as "Binary point", "Slope and bias", or "None". This property applies only when the <code>Type.Method</code> property of the data object is "Fixed point". For more information, see "Fixed-Point Data in Stateflow Charts". • Type.Fixpt.FractionLength — Fraction length, in bits, specified as a string scalar or character vector. This property applies only when the <code>Type.Method</code> property is "Fixed point" and the <code>Type.Fixpt.ScalingMode</code> property is "Binary point".

Property Name	Access	Description
		<ul style="list-style-type: none"> • Type.Fixpt.Slope — Slope, specified as a string scalar or character vector. This property applies only when the <code>Type.Method</code> property is "Fixed point" and the <code>Type.Fixpt.ScalingMode</code> property is "Slope and bias". • Type.Fixpt.Bias — Bias, specified as a string scalar or character vector. This property applies only when the <code>Type.Method</code> property is "Fixed point" and the <code>Type.Fixpt.ScalingMode</code> property is "Slope and bias". • Type.Fixpt.Lock — Whether to prevent replacement of the fixed-point type with an autoscaled type chosen by the Fixed-Point Tool (Fixed-Point Designer), specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property applies only when the <code>Type.Method</code> property of the data object is "Fixed point". • Array.Size — Size of the data object, specified as a string scalar or character vector. For more information, see "Specify Size of Stateflow Data". • Array.IsDynamic — Whether the data object has variable size, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property is equivalent to the Variable Size check box in the Property Inspector, the Model Explorer, or the Data properties dialog box. For more information, see "Declare Variable-Size Data in Stateflow Charts". • Array.FirstIndex — Index for the first element of the array data object, specified as a string scalar or character vector. This property applies only to array data in charts that use C as the action language. For more information, see "Save final value to base workspace". • Complexity — Whether the data object accepts complex values, specified as "On" or "Off". For more information, see "Complex Data in Stateflow Charts". • InitialValue — Initial value, specified as a string scalar or character vector. For more information, see "Initial value". • Range.Minimum — Minimum value, specified as a string scalar or character vector. For more information, see "Limit range". • Range.Maximum — Maximum value, specified as a string scalar or character vector. For more information, see "Limit range". • ResolveToSignalObject — Whether the data object resolves to a <code>Simulink.Signal</code> object that you define in the model or base workspace, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see "Resolve Data Properties from Simulink Signal Objects". • Unit.Name — Unit of measurement, specified as a string scalar or character vector. This property applies only to input and output data. For more information, see "Specify Units for Stateflow Data".
SSIdNumber	RO	Session-independent identifier, specified as an integer scalar. Use this property to distinguish the data object from other objects in the model.

Property Name	Access	Description
SaveToWorkspace	RW	Whether to save the value of the data object to a variable of the same name in the MATLAB base workspace at the end of the simulation, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property applies only to data in charts that use C as the action language. For more information, see "Save final value to base workspace".
Scope	RW	Scope of the data object, specified as one of these values: <ul style="list-style-type: none"> "Local" "Input" "Output" "Constant" "Parameter" "Data Store Memory" "Temporary" "Imported" "Exported" For more information, see "Scope".
Tag	RW	User-defined tag for the data object, specified as data of any type.
TestPoint	RW	Whether to set the data object as a test point, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). You can monitor testpoints with a floating scope during simulation. You can also log test point values to the MATLAB workspace. For more information, see "Monitor Test Points in Stateflow Charts".
Tunable	RW	Whether the data object is a tunable parameter, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). Only tunable parameters can be modified during simulation. This property applies only to parameter data.
UpdateMethod	RW	Method for updating data object, specified as "Discrete" or "Continuous". This property applies only when the <code>ChartUpdate</code> property of the chart that contains the data is "CONTINUOUS". For more information, see "Continuous-Time Modeling in Stateflow".

Stateflow.EMChart

Use `Stateflow.EMChart` objects to configure MATLAB Function blocks through the Stateflow programmatic interface.

MATLAB Function blocks define custom functionality in Simulink models. Use these blocks when:

- You have an existing MATLAB function that models custom functionality, or it is easy for you to create such a function.
- Your model requires custom functionality that is not or cannot be captured in the Simulink graphical language.
- You find it easier to model custom functionality by using a MATLAB function than by using a Simulink block diagram.

- The custom functionality that you want to model does not include continuous or discrete dynamic states. To model dynamic states, use S-functions. See “Create and Configure MATLAB S-Functions” (Simulink).

For more information, see “Implement MATLAB Functions in Simulink with MATLAB Function Blocks” (Simulink).

Tip You can also configure the properties of a MATLAB Function block programmatically by using a `MATLABFunctionConfiguration` object. This object provides a direct interface to the properties of a MATLAB Function block. For more information, see “Configure MATLAB Function Blocks Programmatically” (Simulink).

Property Name	Access	Description
AllowDirectFeedthrough	RW	Whether the MATLAB Function block supports direct feedthrough semantics, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see “Allow direct feedthrough” (Simulink).
ChartUpdate	RW	Activation method for the MATLAB Function block, specified as "CONTINUOUS", "DISCRETE", or "INHERITED". For more information, see “Update method” (Simulink).
Description	RW	Description for the MATLAB Function block, specified as a string scalar or character vector.
Dirty	RW	Whether the MATLAB Function block has changed after being opened or saved, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>).
Document	RW	Document link for the MATLAB Function block, specified as a string scalar or character vector.
EmlDefaultFimath	RW	Default fimath properties for the MATLAB Function block, specified as one of these values: <ul style="list-style-type: none"> • "Same as MATLAB Default" — Use the same fimath properties as the current default fimath object. • "Other:UserSpecified" — Use the InputFimath property to specify the default fimath object.
Iced	RO	Whether the MATLAB Function block is locked, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property is equivalent to the property Locked, but is used internally to prevent changes in the MATLAB Function block during simulation.
Id	RO	Unique identifier, specified as an integer scalar. Use this property to distinguish the MATLAB Function block from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
InputFimath	RW	Default fimath object, specified as a string scalar or character vector. When the EmlDefaultFimath property for the MATLAB Function block is "Other:UserSpecified", you can use this property to: <ul style="list-style-type: none"> • Enter an expression that constructs a fimath object. • Enter the variable name for a fimath object in the MATLAB or model workspace.

Property Name	Access	Description
Inputs	RO	Input arguments of the MATLAB Function block, specified as an array of <code>Stateflow.Data</code> objects. The value of this property depends on the inputs defined in the <code>Script</code> property for the block.
Locked	RW	Whether the MATLAB Function block is locked, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). Enable this property to prevent changes in the MATLAB Function block.
Machine	RO	Machine that contains the MATLAB Function block, specified as a <code>Stateflow.Machine</code> object.
Name	RW	Name of the MATLAB Function block, specified as a string scalar or character vector.
Outputs	RO	Output arguments of the MATLAB Function block, specified as an array of <code>Stateflow.Data</code> objects. The value of this property depends on the outputs defined in the <code>Script</code> property for the block.
Path	RO	Location of the MATLAB Function block in the model hierarchy, specified as a character vector.
SampleTime	RW	Sample time for activating the MATLAB Function block, specified as a string scalar or character vector. This property applies only when the <code>ChartUpdate</code> property for the MATLAB function is "DISCRETE".
SaturateOnIntegerOverflow	RW	Whether the data in the MATLAB Function block saturates on integer overflow, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). When this property is disabled, the data in the function wraps on integer overflow. For more information, see "Saturate on integer overflow" (Simulink).
Script	RW	Code for the MATLAB Function block, specified as a string scalar or character vector. To enter multiple lines of code, you can: <ul style="list-style-type: none"> Call the MATLAB function <code>sprintf</code> and use <code>\n</code> to insert newline characters: <pre>str = sprintf("function y=f(x)\ny=x+1;\nend"); block.Script = str;</pre> Enter a concatenated text expression that uses the function <code>newline</code> to create newline characters: <pre>str = "function y=f(x)" + newline + ... "y=x+1;" + newline + ... "end"; block.Script = str;</pre>
SupportVariableSizing	RW	Whether the MATLAB Function block supports variable-size data, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see "Declare Variable-Size MATLAB Function Block Variables" (Simulink).
Tag	RW	User-defined tag for the MATLAB Function block, specified as data of any type.


Property Name	Access	Description
TreatAsFi	RW	Inherited Simulink signals to treat as Fixed-Point Designer <code>fi</code> objects, specified as one of these values: <ul style="list-style-type: none"> "Fixed-point" — The MATLAB Function block treats all fixed-point inputs as <code>fi</code> objects. "Fixed-point & Integer" — The MATLAB Function block treats all fixed-point and integer inputs as <code>fi</code> objects.
TreatDimensionOfLengthOneAsFixedSize	RW	Whether MATLAB Function block output variables with at least one dimension of length 1 are fixed size, specified as a numeric or logical <code>0</code> (<code>false</code>) or <code>1</code> (<code>true</code>). When this property is <code>true</code> , the object sets variables that are variable size in the block with a dimension of 1 to fixed size. When this property is <code>false</code> , variables in the block that have the Variable size property enabled are always variable size. Prior to R2023a, the object treats variables with at least one dimension of length 1 as fixed size. This property only affects output variables that have the Variable size property enabled. See "Variable size" (Simulink).
VectorOutputs1D	RW	Whether the MATLAB Function block outputs column vectors as one-dimensional data, specified as a numeric or logical <code>0</code> (<code>false</code>) or <code>1</code> (<code>true</code>). For more information, see "Interpret output column vectors as one-dimensional data" (Simulink).

Stateflow.EMFunction

Use `Stateflow.EMFunction` objects to create MATLAB functions for coding algorithms that are more easily expressed by using MATLAB code instead of the graphical Stateflow constructs. Typical applications include:

- Matrix-oriented calculations
- Data analysis and visualization

You can call a MATLAB function in the actions of states and transitions. For more information, see "Reuse MATLAB Code by Defining MATLAB Functions".

Property Name	Access	Description
BadIntersection	RO	Whether the MATLAB function graphically intersects a box, state, or function, specified as a numeric or logical <code>1</code> (<code>true</code>) or <code>0</code> (<code>false</code>).
Chart	RO	Chart that contains the MATLAB function, specified as a <code>Stateflow.Chart</code> object.
CommentText	RW	Comment text for the MATLAB function, specified as a string scalar or character vector. This property applies only when the <code>IsExplicitlyCommented</code> property is <code>true</code> . In the Stateflow Editor, when you point to the comment badge  on the MATLAB function, the text appears as a tooltip. When you set the <code>IsExplicitlyCommented</code> property to <code>false</code> , the value of <code>CommentText</code> reverts to "".

Property Name	Access	Description
Description	RW	Description for the MATLAB function, specified as a string scalar or character vector.
Document	RW	Document link for the MATLAB function, specified as a string scalar or character vector.
EmlDefaultFimath	RW	<p>Default fimath properties for the MATLAB function, specified as one of these values:</p> <ul style="list-style-type: none"> • "Same as MATLAB Default" — Use the same fimath properties as the current default fimath object. • "Other:UserSpecified" — Use the InputFimath property to specify the default fimath object. <p>This property applies only when the ActionLanguage of the chart that contains the function is "C". Otherwise, the behavior of data depends on the value of the EmlDefaultFimath property for the chart.</p>
FontSize	RW	Font size for the MATLAB function label, specified as a scalar. The StateFont.Size property of the chart that contains the graphical function sets the initial value of this property.
Id	RO	Unique identifier, specified as an integer scalar. Unlike SSIdNumber, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
InlineOption	RW	<p>Appearance of the MATLAB function in generated code, specified as one of these values:</p> <ul style="list-style-type: none"> • "Auto" — An internal calculation determines the appearance of the function in generated code. • "Function" — The function is implemented as a separate C function. • "Inline" — Calls to the function are replaced by code as long as the function is not part of a recursion. <p>For more information, see "Inline State Functions in Generated Code" (Simulink Coder).</p>
InputFimath	RW	<p>Default fimath object, specified as a string scalar or character vector. When the EmlDefaultFimath property for the MATLAB function is "Other:UserSpecified", you can use this property to:</p> <ul style="list-style-type: none"> • Enter an expression that constructs a fimath object. • Enter the variable name for a fimath object in the MATLAB or model workspace. <p>This property applies only when the ActionLanguage of the chart that contains the function is "C". Otherwise, the behavior of data depends on the value of the InputFimath property for the chart.</p>
IsCommented	RO	Whether the MATLAB function is commented out, specified as a numeric or logical 1 (true) or 0 (false). This property is true when either IsExplicitlyCommented or IsImplicitlyCommented is true.

Property Name	Access	Description
IsExplicitlyCommented	RW	Whether to comment out the MATLAB function, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). Setting this property to <code>true</code> is equivalent to right-clicking the MATLAB function and selecting Comment Out . For more information, see “Comment Out Objects in a Stateflow Chart”.
IsImplicitlyCommented	RO	Whether the MATLAB function is implicitly commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). The MATLAB function is implicitly commented out when you explicitly comment out an object that contains it. If the MATLAB function is contained in an atomic subchart or an atomic box, this property is <code>false</code> unless the explicitly commented object is also contained in the atomic subchart or atomic box.
LabelString	RW	Label for the MATLAB function, specified as a string scalar or character vector.
Machine	RO	Machine that contains the MATLAB function, specified as a <code>Stateflow.Machine</code> object.
Name	RW	Name of the MATLAB function, specified as a string scalar or character vector.
Path	RO	Location of the parent of the MATLAB function in the model hierarchy, specified as a character vector.
Position	RW	Position and size of the MATLAB function, specified as a four-element numeric vector of the form <code>[left top width height]</code> .
SSIdNumber	RO	Session-independent identifier, specified as an integer scalar. Use this property to distinguish the MATLAB function from other objects in the model.
SaturateOnIntegerOverflow	RW	Whether the data in the MATLAB function saturates on integer overflow, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). When this property is disabled, the data in the function wraps on integer overflow. For more information, see “Handle Integer Overflow for Chart Data”. This property applies only when the <code>ActionLanguage</code> of the chart that contains the function is “C”. Otherwise, the behavior of data depends on the value of the <code>SaturateOnIntegerOverflow</code> property for the chart.
Script	RW	Code for the MATLAB function, specified as a string scalar or character vector. To enter multiple lines of code, you can: <ul style="list-style-type: none"> Call the MATLAB function <code>sprintf</code> and use the escape sequence <code>\n</code> to insert newline characters: <pre>str = sprintf("function y=f(x)\ny=x+1;\nend"); function.Script = str;</pre> Enter a concatenated text expression that uses the function <code>newline</code> to create newline characters: <pre>str = "function y=f(x)" + newline + ... "y=x+1;" + newline + ... "end"; function.Script = str;</pre>

Property Name	Access	Description
Subviewer	RO	Subviewer for the MATLAB function, specified as a <code>Stateflow.Chart</code> , <code>Stateflow.State</code> , <code>Stateflow.Box</code> , or <code>Stateflow.Function</code> object. The subviewer is the chart or subchart where you can graphically view the MATLAB function.
Tag	RW	User-defined tag for the MATLAB function, specified as data of any type.

Stateflow.Editor

Use the `Stateflow.Editor` object to access the graphical aspects of a Stateflow chart or state transition table. You can use the `Stateflow.Editor` object to control the position, size, and magnification level of the Stateflow Editor window.

Property Name	Access	Description
WindowPosition	RW	Position and size of the Stateflow editor window, specified as a four-element numeric vector of the form <code>[left top width height]</code> .
ZoomFactor	RW	Magnification level of the chart or state transition table in the editor, specified as a scalar value between 0.5 and 10. A value of 1 corresponds to a magnification of 100%.

Stateflow.Event

Use `Stateflow.Event` objects to trigger actions in one of these objects:

- A parallel state in a Stateflow chart
- Another Stateflow chart
- A Simulink triggered or function-call subsystem


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

Property Name	Access	Description
Debug	RW	Debugger properties for the event, specified as a <code>Stateflow.EventDebug</code> object with these properties: <ul style="list-style-type: none"> • Breakpoints.StartBroadcast — Whether to set the <code>Start</code> of Broadcast breakpoint, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • Breakpoints.EndBroadcast — Whether to set the <code>End</code> of Broadcast breakpoint, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see “Set Breakpoints to Debug Charts”.
Description	RW	Description for the event, specified as a string scalar or character vector.
Document	RW	Document link for the event, specified as a string scalar or character vector.

Property Name	Access	Description
Id	RO	Unique identifier, specified as an integer scalar. Use this property to distinguish the event from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
Machine	RO	Machine that contains the event, specified as a <code>Stateflow.Machine</code> object.
Name	RW	Name of the event, specified as a string scalar or character vector.
Path	RO	Location of the parent of the event in the model hierarchy, specified as a character vector.
Port	RW	Port index for the event, specified as an integer scalar. This property applies only to input and output events. For more information, see "Port".
Scope	RW	Scope of the event, specified as "Local", "Input", or "Output". For more information, see "Scope".
Tag	RW	User-defined tag for the event, specified as data of any type.
Trigger	RW	Type of trigger associated with the event, specified as a string scalar or character vector that depends on the scope of the data: <ul style="list-style-type: none"> For input events, use "Function call", "Rising", "Falling", or "Either". For output events, use "Function call" or "Either". <p>This property does not apply to local events. For more information, see "Trigger".</p>

Stateflow.Function

Use `Stateflow.Function` objects to create graphical functions that contain control-flow logic and iterative loops. You create graphical functions with flow charts that use connective junctions and transitions. You can call a graphical function in the actions of states and transitions. For more information, see "Reuse Logic Patterns by Defining Graphical Functions".

Property Name	Access	Description
BadIntersection	RO	Whether the graphical function graphically intersects a box, state, or function, specified as a numeric or logical 1 (true) or 0 (false).
Chart	RO	Chart that contains the graphical function, specified as a <code>Stateflow.Chart</code> object.
CommentText	RW	Comment text for the graphical function, specified as a string scalar or character vector. This property applies only when the <code>IsExplicitlyCommented</code> property is true. In the Stateflow Editor, when you point to the comment badge  on the graphical function, the text appears as a tooltip. When you set the <code>IsExplicitlyCommented</code> property to false, the value of <code>CommentText</code> reverts to "".
ContentPreviewEnabled	RW	Whether to display a preview of the graphical function contents, specified as a numeric or logical 1 (true) or 0 (false). This property applies only when the <code>IsSubchart</code> property is true.

Property Name	Access	Description
Debug	RW	<p>Debugger properties for the graphical function, specified as a <code>Stateflow.FunctionDebug</code> object with this property:</p> <ul style="list-style-type: none"> • Breakpoints.OnDuring — Whether to set the <code>During</code> Function Call breakpoint, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). <p>For more information, see “Set Breakpoints to Debug Charts”.</p>
Description	RW	Description for the graphical function, specified as a string scalar or character vector.
Document	RW	Document link for the graphical function, specified as a string scalar or character vector.
FontSize	RW	Font size for the graphical function label, specified as a scalar. The <code>StateFont.Size</code> property of the chart that contains the graphical function sets the initial value of this property.
Id	RO	Unique identifier, specified as an integer scalar. Unlike <code>SSIdNumber</code> , the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
InlineOption	RW	<p>Appearance of the graphical function in generated code, specified as one of these values:</p> <ul style="list-style-type: none"> • "Auto" — An internal calculation determines the appearance of the function in generated code. • "Function" — The function is implemented as a separate C function. • "Inline" — Calls to the function are replaced by code as long as the function is not part of a recursion. <p>For more information, see “Inline State Functions in Generated Code” (Simulink Coder).</p>
IsCommented	RO	Whether the graphical function is commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property is <code>true</code> when either <code>IsExplicitlyCommented</code> or <code>IsImplicitlyCommented</code> is <code>true</code> .
IsExplicitlyCommented	RW	Whether to comment out the graphical function, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). Setting this property to <code>true</code> is equivalent to right-clicking the graphical function and selecting Comment Out . For more information, see “Comment Out Objects in a Stateflow Chart”.
IsGrouped	RW	Whether the function is a grouped function, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). When you copy and paste a grouped function, you copy not only the function but all of its contents. For more information, see “Copy and Paste by Grouping” on page 2-28.
IsImplicitlyCommented	RO	Whether the graphical function is implicitly commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). The graphical function is implicitly commented out when you explicitly comment out an object that contains it. If the graphical function is contained in an atomic subchart or an atomic box, this property is <code>false</code> unless the explicitly commented object is also contained in the atomic subchart or atomic box.


Property Name	Access	Description
IsSubchart	RW	Whether the function is a subchart, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>).
LabelString	RW	Label for the graphical function, specified as a string scalar or character vector.
Machine	RO	Machine that contains the graphical function, specified as a <code>Stateflow.Machine</code> object.
Name	RW	Name of the graphical function, specified as a string scalar or character vector.
Path	RO	Location of the parent of the graphical function in the model hierarchy, specified as a character vector.
Position	RW	Position and size of the graphical function, specified as a four-element numeric vector of the form <code>[left top width height]</code> .
SSIdNumber	RO	Session-independent identifier, specified as an integer scalar. Use this property to distinguish the graphical function from other objects in the model.
Subviewer	RO	Subviewer for the graphical function, specified as a <code>Stateflow.Chart</code> , <code>Stateflow.State</code> , <code>Stateflow.Box</code> , or <code>Stateflow.Function</code> object. The subviewer is the chart or subchart where you can graphically view the graphical function.
Tag	RW	User-defined tag for the graphical function, specified as data of any type.

Stateflow.Junction

Use `Stateflow.Junction` objects to create junctions that:

- Represent decision points in a transition path
- Record the activity of substates inside a superstate

For more information, see “Combine Transitions and Junctions to Create Branching Paths” and “Resume Prior Substate Activity by Using History Junctions”.

Property Name	Access	Description
ArrowSize	RW	Size of incoming transition arrows, specified as a scalar.
Chart	RO	Chart that contains the junction, specified as a <code>Stateflow.Chart</code> object.
CommentText	RW	Comment text added to the junction, specified as a string scalar or character vector. This property applies only when the <code>IsExplicitlyCommented</code> property is <code>true</code> . In the Stateflow Editor, when you point to the comment badge  on the junction, the text appears as a tooltip. When you set the <code>IsExplicitlyCommented</code> property to <code>false</code> , the value of <code>CommentText</code> reverts to <code>""</code> .
Description	RW	Description for the junction, specified as a string scalar or character vector.
Document	RW	Document link for the junction, specified as a string scalar or character vector.

Property Name	Access	Description
Id	RO	Unique identifier, specified as an integer scalar. Unlike <code>SSIdNumber</code> , the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
IsCommented	RO	Whether the junction is commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property is <code>true</code> when either <code>IsExplicitlyCommented</code> or <code>IsImplicitlyCommented</code> is <code>true</code> .
IsExplicitlyCommented	RW	Whether to comment out the junction, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). Setting this property to <code>true</code> is equivalent to right-clicking the junction and selecting Comment Out . For more information, see “Comment Out Objects in a Stateflow Chart”.
IsImplicitlyCommented	RO	Whether the junction is implicitly commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). The junction is implicitly commented out when you explicitly comment out an object that contains it. If the junction is contained in an atomic subchart or an atomic box, this property is <code>false</code> unless the explicitly commented object is also contained in the atomic subchart or atomic box.
Machine	RO	Machine that contains the junction, specified as a <code>Stateflow.Machine</code> object.
Path	RO	Location of the parent of the junction in the model hierarchy, specified as a character vector.
Position	RW	Position and size of the junction, specified as a <code>Stateflow.JunctionPosition</code> object with these properties: <ul style="list-style-type: none"> • Center — Position of the center of the junction, specified as a two-element numeric vector [<code>x</code> <code>y</code>] of coordinates relative to the upper left corner of the chart. • Radius — Radius of the junction, specified as a scalar.
SSIdNumber	RO	Session-independent identifier, specified as an integer scalar. Use this property to distinguish the junction from other objects in the model.
Subviewer	RO	Subviewer for the junction, specified as a <code>Stateflow.Chart</code> , <code>Stateflow.State</code> , <code>Stateflow.Box</code> , or <code>Stateflow.Function</code> object. The subviewer is the chart or subchart where you can graphically view the junction.
Tag	RW	User-defined tag for the junction, specified as data of any type.
Type	RW	Type of junction, specified as one of these values: <ul style="list-style-type: none"> • "CONNECTIVE" — Connective junction that represents a decision point in a transition path • "HISTORY" — History junction that records the activity of substates inside a superstate

Stateflow.Machine

From a Stateflow perspective, `Stateflow.Machine` objects are equivalent to Simulink models. A `Stateflow.Machine` object contains `Stateflow.Chart`, `Stateflow.StateTransitionTableChart`, `Stateflow.TruthTableChart`, and

Stateflow.EMChart objects that represent the Stateflow charts, State Transition Table blocks, Truth Table blocks, and MATLAB Function blocks in a Simulink model. For more information, see “Overview of the Stateflow API” on page 1-2.

Property Name	Access	Description										
Created	RO	Date of the creation of the machine, specified as a character vector.										
Creator	RW	Creator of the machine, specified as a string scalar or character vector.										
Debug	RW	<p>Debugger properties for charts in the machine, specified as a Stateflow.MachineDebug object with these properties:</p> <ul style="list-style-type: none"> • Animation.Enabled — Whether to animate the charts in the machine during simulation, specified as a numeric or logical 1 (true) or 0 (false). Disabling this property is equivalent to selecting None in the Animation Speed drop-down list in the Debug tab. • Animation.Delay — Delay that the chart animation uses for highlighting each transition segment in the machine, specified as a scalar. These values correspond to the settings of the Animation Speed drop-down list in the Debug tab: <table border="1" data-bbox="646 877 1474 1100"> <thead> <tr> <th>Delay Value</th> <th>Animation Speed</th> </tr> </thead> <tbody> <tr> <td>0.5</td> <td>Slow</td> </tr> <tr> <td>0.2</td> <td>Medium</td> </tr> <tr> <td>0</td> <td>Fast</td> </tr> <tr> <td>-1</td> <td>Lightning Fast</td> </tr> </tbody> </table> • Animation.MaintainHighlighting — Whether to maintain the highlighting of active states in the machine after the simulation ends, specified as a numeric or logical 1 (true) or 0 (false). 	Delay Value	Animation Speed	0.5	Slow	0.2	Medium	0	Fast	-1	Lightning Fast
Delay Value	Animation Speed											
0.5	Slow											
0.2	Medium											
0	Fast											
-1	Lightning Fast											
Description	RW	Description for the machine, specified as a string scalar or character vector.										
Dirty	RW	Whether the Simulink model for the machine has changed after being opened or saved, specified as a numeric or logical 1 (true) or 0 (false).										
Document	RW	Document link for the machine, specified as a string scalar or character vector.										
FullFileName	RO	Full file path of the Simulink model for the machine, specified as a character vector.										
Iced	RO	Whether the machine is locked, specified as a numeric or logical 1 (true) or 0 (false). This property is equivalent to the property Locked , but is used internally to prevent changes in the machine during simulation.										
Id	RO	Unique identifier, specified as an integer scalar. Use this property to distinguish the machine from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.										
IsLibrary	RO	Whether the Simulink model for the machine builds a library and not an application, specified as a numeric or logical 1 (true) or 0 (false).										

Property Name	Access	Description
Locked	RW	Whether the machine is locked, specified as a numeric or logical 1 (true) or 0 (false). Enable this property to prevent changes in the Stateflow charts, state transition tables, and truth table blocks in this machine.
Modified	RW	Record of modifications to the machine, specified as a string scalar or character vector.
Name	RO	Name of the Simulink model for the machine, specified as a character vector.
Path	RO	Location of the machine in the model hierarchy, specified as a character vector.
Tag	RW	User-defined tag for the machine, specified as data of any type.
Version	RW	Version of the machine, specified as a string scalar or character vector.

Stateflow.Message

Use `Stateflow.Message` objects to communicate data locally or between Stateflow charts in Simulink models. For more information, see “Communicate with Stateflow Charts by Sending Messages”.

Property Name	Access	Description
CompiledSize	RO	Message data size as determined by the compiler, specified as a character vector.
CompiledType	RO	Data type as determined by the compiler, specified as a character vector.

Property Name	Access	Description
DataType	RW	<p>Data type of the message, specified as a string scalar or character vector that depends on the <code>Props.Type.Method</code> property of the message:</p> <ul style="list-style-type: none"> If the <code>Props.Type.Method</code> property of the message is "Inherit", the value of this property is "Inherit: Same as Simulink". If the <code>Props.Type.Method</code> property of the message is "Built-in", you can specify this property with one of these options: <ul style="list-style-type: none"> "double" "single" "int8" "int16" "int32" "int64" "uint8" "uint16" "uint32" "uint64" "boolean" "string" "ml" (Supported only in charts that use C as the action language) Otherwise, the <code>Props.Type</code> properties of the message determine the value of this property. <p>For more information, see the section Add Data on page 1-0 in "Create Charts by Using the Stateflow API" on page 1-19.</p>
Description	RW	Description for the message, specified as a string scalar or character vector.
Document	RW	Document link for the message, specified as a string scalar or character vector.
Id	RO	Unique identifier, specified as an integer scalar. Unlike <code>SSIdNumber</code> , the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
InitializeMethod	RW	<p>Method for initializing the value of the message data, specified as a string scalar or character vector that depends on the scope of the message:</p> <ul style="list-style-type: none"> For local and output messages, use "Expression" or "Parameter". For input messages, use "Not Needed". <p>To specify the initial value of the message data, use the <code>Props.InitialValue</code> property.</p> <p>For more information, see "Initial Value".</p>

Property Name	Access	Description
Machine	RO	Machine that contains the message, specified as a <code>Stateflow.Machine</code> object.
MessagePriorityOrder	RW	Type of priority queue for the message, specified as one of these values: <ul style="list-style-type: none"> "Ascending" — Messages are received in ascending order of the message data value. "Descending" — Messages are received in descending order of the message data value. <p>This property applies only when the <code>QueueType</code> property of the message is "Priority". For more information, see "Queue Type".</p>
Name	RW	Name of the message, specified as a string scalar or character vector.
Path	RO	Location of the parent of the message in the model hierarchy, specified as a character vector.
Port	RW	Port index for the message, specified as an integer scalar. This property applies only to input and output messages. For more information, see "Port".
Priority	RW	Priority for the message, specified as a string scalar or character vector. If two distinct messages occur at the same time, this property determines which message is processed first. A smaller numeric value indicates a higher priority. This property applies only to local and output messages in discrete-event charts. For more information, see "Create Custom Queuing Systems Using Discrete-Event Stateflow Charts" (SimEvents).


Property Name	Access	Description
Props	RW	<p>Data specification properties, specified as a <code>Stateflow.DataProps</code> object with these properties:</p> <ul style="list-style-type: none"> • Type.Method — Method for setting the data type of the message, specified as "Inherited", "Built-in", "Bus Object", "Enumerated", "Expression", or "Fixed point". This property is equivalent to the Mode field of the Data Type Assistant in the Model Explorer and the Data properties dialog box. For more information, see "Specify Type of Stateflow Data". • Type.BusObject — Name of the <code>Simulink.Bus</code> object that defines the message data, specified as a string scalar or character vector. This property applies only when the <code>Type.Method</code> property of the data object is "Bus Object". For more information, see "Access Bus Signals Through Stateflow Structures". • Type.EnumType — Name of the enumerated type that defines the message data, specified as a string scalar or character vector. This property applies only when the <code>Type.Method</code> property of the data object is "Enumerated". For more information, see "Reference Values by Name by Using Enumerated Data". • Type.Expression — Expression that evaluates to the data type of the message data, specified as a string scalar or character vector. This property applies only when the <code>Type.Method</code> property of the data object is "Expression". For more information, see "Specify Data Properties by Using MATLAB Expressions". • Type.Signed — Signedness, specified as a numeric or logical 1 (true) or 0 (false). This property applies only when the <code>Type.Method</code> property of the data object is "Fixed point". For more information, see "Fixed-Point Data in Stateflow Charts". • Type.WordLength — Word length, in bits, specified as a string scalar or character vector. This property applies only when the <code>Type.Method</code> property of the data object is "Fixed point". For more information, see "Fixed-Point Data in Stateflow Charts". • Type.Fixpt.ScalingMode — Method for scaling the fixed-point message data, specified as "Binary point", "Slope and bias", or "None". This property applies only when the <code>Type.Method</code> property of the data object is "Fixed point". For more information, see "Fixed-Point Data in Stateflow Charts". • Type.Fixpt.FractionLength — Fraction length, in bits, specified as a string scalar or character vector. This property applies only when the <code>Type.Method</code> property is "Fixed point" and the <code>Type.Fixpt.ScalingMode</code> property is "Binary point". • Type.Fixpt.Slope — Slope, specified as a string scalar or character vector. This property applies only when the <code>Type.Method</code> property is "Fixed point" and the <code>Type.Fixpt.ScalingMode</code> property is "Slope and bias". • Type.Fixpt.Bias — Bias, specified as a string scalar or character vector. This property applies only to when the <code>Type.Method</code> property

Property Name	Access	Description
		<p>is "Fixed point" and the <code>Type.Fixpt.ScalingMode</code> property is "Slope and bias".</p> <ul style="list-style-type: none"> • Type.Fixpt.Lock — Whether to prevent replacement of the fixed-point type with an autoscaled type chosen by the Fixed-Point Tool (Fixed-Point Designer), specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property applies only when the <code>Type.Method</code> property of the data object is "Fixed point". • Array.Size — Size of the message data, specified as a string scalar or character vector. For more information, see "Specify Size of Stateflow Data". • Complexity — Whether the message accepts complex values, specified as "On" or "Off". For more information, see "Complex Data in Stateflow Charts". • InitialValue — Initial value, specified as a string scalar or character vector.
QueueCapacity	RW	Length of the internal queue for the message, specified as an integer scalar. This property applies only to local messages and to input messages that have <code>UseInternalQueue</code> set to <code>true</code> . For more information, see "Queue Capacity".
QueueOverflowDiagnostic	RW	Level of diagnostic action when the number of incoming messages exceeds the queue capacity for the message, specified as "Error", "Warning", or "None". This property applies only to local messages and to input messages that have <code>UseInternalQueue</code> set to <code>true</code> . For more information, see "Queue Overflow Diagnostic".
QueueType	RW	Order in which messages are removed from the receiving queue, specified as one of these values: <ul style="list-style-type: none"> • "FIFO" — First in, first out. • "LIFO" — Last in, first out. • "Priority" — Remove messages according to the value in the data field. To specify the order, use the <code>MessagePriorityOrder</code> property for the message. <p>This property applies only to local messages and to input messages that have <code>UseInternalQueue</code> set to <code>true</code>. For more information, see "Queue Type".</p>
SSIdNumber	RO	Session-independent identifier, specified as an integer scalar. Use this property to distinguish the message from other objects in the model.
Scope	RW	Scope of the message, specified as "Local", "Input", or "Output". For more information, see "Scope".
Tag	RW	User-defined tag for the message, specified as data of any type.
UseInternalQueue	RW	Whether the Stateflow chart maintains an internal receiving queue for the input message, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property applies only to input messages. For more information, see "Use Internal Queue".

Stateflow.Port

Use `Stateflow.Port` objects to create ports and junctions that provide entry and exit connections across boundaries in the Stateflow hierarchy. Entry and exit ports improve componentization by isolating the transition logic for entering and exiting states. Unlike supertransitions, they can be used in atomic subcharts. For more information, see “Create Entry and Exit Connections Across State Boundaries”.

Entry and exit ports are located on the boundary of a state or atomic subchart. Each port has a matching junction that marks the entry or exit point inside the state or atomic subchart. The port and junction are represented by separate `Stateflow.Port` objects.

Property Name	Access	Description
ArrowSize	RW	Size of incoming transition arrows, specified as a scalar.
Chart	RO	Chart that contains the port or junction, specified as a <code>Stateflow.Chart</code> object.
CommentText	RW	Comment text added to the entry or exit junction, specified as a string scalar or character vector. This property applies only to entry and exit junctions when the <code>IsExplicitlyCommented</code> property is <code>true</code> . In the Stateflow Editor, when you point to the comment badge  on the junction, the text appears as a tooltip. When you set the <code>IsExplicitlyCommented</code> property to <code>false</code> , the value of <code>CommentText</code> reverts to "".
Description	RW	Description for the port or junction, specified as a string scalar or character vector.
Document	RW	Document link for the port or junction, specified as a string scalar or character vector.
Home	RO	Home state or subchart, specified as a <code>Stateflow.State</code> or <code>Stateflow.AtomicSubchart</code> object. The home of an entry or exit port is the state or subchart whose boundary contains the port. This property applies only to entry and exit ports.
Id	RO	Unique identifier, specified as an integer scalar. Unlike <code>SSIdNumber</code> , the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
IsCommented	RO	Whether the port or junction is commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property is <code>true</code> when either <code>IsExplicitlyCommented</code> or <code>IsImplicitlyCommented</code> is <code>true</code> .
IsExplicitlyCommented	RW	Whether to comment out the junction and port pair, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). Setting this property to <code>true</code> is equivalent to right-clicking the entry or exit junction and selecting Comment Out . This property applies only to entry and exit junctions. Attempting to set this property on an entry or exit port results in an error. For more information, see “Comment Out Objects in a Stateflow Chart”.


Property Name	Access	Description
IsImplicitlyCommented	RO	Whether the port or junction is implicitly commented out, specified as a numeric or logical 1 (true) or 0 (false). The port or junction is implicitly commented out when you explicitly comment out an object that contains it. Additionally, an entry or exit port is implicitly commented out when you explicitly comment out the matching entry or exit junction. If the port or junction is contained in an atomic subchart or an atomic box, this property is false unless the explicitly commented object is also contained in the atomic subchart or atomic box.
LabelPosition	RW	Position and size of the port or junction label, specified as a four-element numeric vector of the form [left top width height].
LabelString	RW	Label for the port or junction, specified as a string scalar or character vector. Changing this property automatically sets the LabelString property for the matching Stateflow.Port object to the same value.
Linked	RO	Whether the port or junction has a matching junction or port, specified as a numeric or logical 1 (true) or 0 (false). This property is used to detect internal inconsistencies in the chart.
Machine	RO	Machine that contains the port or junction, specified as a Stateflow.Machine object.
Path	RO	Location of the parent of the port or junction in the model hierarchy, specified as a character vector.
PortType	RO	Type of port or junction, specified as one of these values: <ul style="list-style-type: none"> 'EntryJunction' — Entry junction inside a state or atomic subchart 'EntryPort' — Entry port on the boundary of a state or atomic subchart 'ExitJunction' — Exit junction inside a state or atomic subchart 'ExitPort' — Exit port on the boundary of a state or atomic subchart
Position	RW	Position and size of the port or junction, specified as a Stateflow.PortPosition object with these properties: <ul style="list-style-type: none"> Center — Position of the center of the port or junction, specified as a two-element numeric vector [x y] of coordinates relative to the upper left corner of the chart. Radius — Radius of the port or junction, specified as a scalar.
SSIdNumber	RO	Session-independent identifier, specified as an integer scalar. Use this property to distinguish the port or junction from other objects in the model.
Subviewer	RO	Subviewer for the port or junction, specified as a Stateflow.Chart, Stateflow.State, or Stateflow.Box object. The subviewer is the chart or subchart where you can graphically view the port.
Tag	RW	User-defined tag for the port or junction, specified as data of any type.

Stateflow.SLFunction

Use `Stateflow.SLFunction` objects to create Simulink functions that enable you to call Simulink subsystems in the actions of states and transitions. Typical applications include:

- Defining a function that requires Simulink blocks
- Scheduling execution of multiple controllers


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

Property Name	Access	Description
<code>BadIntersection</code>	RO	Whether the Simulink function graphically intersects a box, state, or function, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>).
<code>Chart</code>	RO	Chart that contains the Simulink function, specified as a <code>Stateflow.Chart</code> object.
<code>CommentText</code>	RW	Comment text added to the Simulink function, specified as a string scalar or character vector. This property applies only when the <code>IsExplicitlyCommented</code> property is <code>true</code> . In the Stateflow Editor, when you point to the comment badge  on the Simulink function, the text appears as a tooltip. When you set the <code>IsExplicitlyCommented</code> property to <code>false</code> , the value of <code>CommentText</code> reverts to <code>""</code> .
<code>ContentPreviewEnabled</code>	RW	Whether to display a preview of the Simulink function contents, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>).
<code>Description</code>	RW	Description for the Simulink function, specified as a string scalar or character vector.
<code>Document</code>	RW	Document link for the Simulink function, specified as a string scalar or character vector.
<code>FontSize</code>	RW	Font size for the Simulink function label, specified as a scalar. The <code>StateFont.Size</code> property of the chart that contains the Simulink function sets the initial value of this property.
<code>Id</code>	RO	Unique identifier, specified as an integer scalar. Unlike <code>SSIdNumber</code> , the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
<code>IsCommented</code>	RO	Whether the Simulink function is commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property is <code>true</code> when either <code>IsExplicitlyCommented</code> or <code>IsImplicitlyCommented</code> is <code>true</code> .
<code>IsExplicitlyCommented</code>	RW	Whether to comment out the Simulink function, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). Setting this property to <code>true</code> is equivalent to right-clicking the Simulink function and selecting Comment Out . For more information, see “Comment Out Objects in a Stateflow Chart”.
<code>IsImplicitlyCommented</code>	RO	Whether the Simulink function is implicitly commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). The Simulink function is implicitly commented out when you explicitly comment out an object that contains it. If the Simulink function is contained in an atomic subchart or an atomic box, this property is <code>false</code> unless the explicitly commented object is also contained in the atomic subchart or atomic box.

Property Name	Access	Description
LabelString	RW	Label for the Simulink function, specified as a string scalar or character vector.
Machine	RO	Machine that contains the Simulink function, specified as a <code>Stateflow.Machine</code> object.
Name	RW	Name of the Simulink function, specified as a string scalar or character vector.
Path	RO	Location of the parent of the Simulink function in the model hierarchy, specified as a character vector.
Position	RW	Position and size of the Simulink function, specified as a four-element numeric vector of the form [left top width height].
SSIdNumber	RO	Session-independent identifier, specified as an integer scalar. Use this property to distinguish the Simulink function from other objects in the model.
Subviewer	RO	Subviewer for the Simulink function, specified as a <code>Stateflow.Chart</code> , <code>Stateflow.State</code> , <code>Stateflow.Box</code> , or <code>Stateflow.Function</code> object. The subviewer is the chart or subchart where you can graphically view the Simulink function.
Tag	RW	User-defined tag for the Simulink function, specified as data of any type.

Stateflow.SimulinkBasedState

Use `Stateflow.SimulinkBasedState` objects to create Simulink subsystems within a Stateflow state. With Simulink based states, you can model hybrid dynamic systems or systems that switch between periodic and continuous time dynamics. For more information, see “Simulink Subsystems as States”.

Property Name	Access	Description
ArrowSize	RW	Size of incoming transition arrows, specified as a scalar.
BadIntersection	RO	Whether the Simulink based state graphically intersects a box, state, or function, specified as a numeric or logical 1 (true) or 0 (false).
Chart	RO	Chart that contains the Simulink based state, specified as a <code>Stateflow.Chart</code> object.
CommentText	RW	Comment text added to the Simulink based state, specified as a string scalar or character vector. This property applies only when the <code>IsExplicitlyCommented</code> property is true. In the Stateflow Editor, when you point to the comment badge  on the Simulink based state, the text appears as a tooltip. When you set the <code>IsExplicitlyCommented</code> property to false, the value of <code>CommentText</code> reverts to "".
ContentPreviewEnabled	RW	Whether to display a preview of the Simulink based state contents, specified as a numeric or logical 1 (true) or 0 (false).


Property Name	Access	Description
Debug	RW	<p>Debugger properties for the Simulink based state, specified as a <code>Stateflow.StateDebug</code> object with these properties:</p> <ul style="list-style-type: none"> • OnEntry — Whether to set the <code>On State Entry</code> breakpoint, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • OnDuring — Whether to set the <code>During State</code> breakpoint, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • OnExit — Whether to set the <code>On State Exit</code> breakpoint, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). <p>For more information, see “Set Breakpoints to Debug Charts”.</p>
Description	RW	Description for the Simulink based state, specified as a string scalar or character vector.
Document	RW	Document link for the Simulink based state, specified as a string scalar or character vector.
ExecutionOrder	RW	<p>Execution order for the Simulink based state in parallel (AND) decomposition, specified as an integer scalar. This property applies only when both of these conditions are satisfied:</p> <ul style="list-style-type: none"> • The <code>Type</code> property of the Simulink based state is “AND”. • The <code>UserSpecifiedStateTransitionExecutionOrder</code> property of the chart that contains the Simulink based state is <code>true</code>.
FontSize	RW	Font size for the Simulink based state label, specified as a scalar. The <code>StateFont.Size</code> property of the chart that contains the Simulink based state sets the initial value of this property.
HasOutputData	RW	Whether to create an active state data output port for the Simulink based state, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see “Monitor State Activity Through Active State Data”.
Id	RO	Unique identifier, specified as an integer scalar. Unlike <code>SSIdNumber</code> , the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
IsCommented	RO	Whether the Simulink based state is commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property is <code>true</code> when either <code>IsExplicitlyCommented</code> or <code>IsImplicitlyCommented</code> is <code>true</code> .
IsExplicitlyCommented	RW	Whether to comment out the Simulink based state, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). Setting this property to <code>true</code> is equivalent to right-clicking the Simulink based state and selecting Comment Out . For more information, see “Comment Out Objects in a Stateflow Chart”.
IsImplicitlyCommented	RO	Whether the Simulink based state is implicitly commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). The Simulink based state is implicitly commented out when you explicitly comment out an object that contains it. If the Simulink based state is contained in an atomic subchart, this property is <code>false</code> unless the explicitly commented object is also contained in the atomic subchart.

Property Name	Access	Description
LoggingInfo	RW	<p>Signal logging properties for the Simulink based state, specified as a <code>Stateflow.SigLoggingInfo</code> object with these properties:</p> <ul style="list-style-type: none"> • DataLogging — Whether to enable signal logging, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • DecimateData — Whether to limit the amount of logged data, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • Decimation — Decimation interval, specified as an integer scalar. This property applies only when the <code>DecimateData</code> property is <code>true</code>. • LimitDataPoints — Whether to limit the number of data points to log, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • MaxPoints — Maximum number of data points to log, specified as an integer scalar. This property applies only when the <code>LimitDataPoints</code> property is <code>true</code>. • NameMode — Source of the signal name, specified as "SignalName" or "Custom". • LoggingName — Custom signal name, specified as a string scalar or character vector. This property applies only when the <code>NameMode</code> property is "Custom". <p>Signal logging saves the self activity of the Simulink based state to the MATLAB workspace during simulation. For more information, see "Log Simulation Output for States and Data".</p>
Machine	RO	Machine that contains the Simulink based state, specified as a <code>Stateflow.Machine</code> object.
Name	RW	Name of the Simulink based state, specified as a string scalar or character vector.
OutputData	RO	Active state data object for the Simulink based state, specified as a <code>Stateflow.Data</code> object. This property applies only when the <code>HasOutputData</code> property for the Simulink based state is <code>true</code> .
OutputMonitoringMode	RW	Monitoring mode for the active state output data, specified as a string scalar or character vector. For Simulink based states, the only option is "SelfActivity".
OutputPortName	RW	Name of the active state data object for the Simulink based state, specified as a string scalar or character vector. This property applies only when the <code>HasOutputData</code> property for the Simulink based state is <code>true</code> .
Path	RO	Location of the parent of the Simulink based state in the model hierarchy, specified as a character vector.
Position	RW	Position and size of the Simulink based state, specified as a four-element numeric vector of the form [left top width height].
SSIdNumber	RO	Session-independent identifier, specified as an integer scalar. Use this property to distinguish the Simulink based state from other objects in the model.

Property Name	Access	Description
Subviewer	RO	Subviewer for the Simulink based state, specified as a <code>Stateflow.Chart</code> , <code>Stateflow.State</code> , or <code>Stateflow.Box</code> object. The subviewer is the chart or subchart where you can graphically view the Simulink based state.
Tag	RW	User-defined tag for the Simulink based state, specified as data of any type.
TestPoint	RW	Whether to set the Simulink based state as a test point, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). You can monitor testpoints with a floating scope during simulation. You can also log test point values to the MATLAB workspace. For more information, see “Monitor Test Points in Stateflow Charts”.
Type	RO	Decomposition of sibling states, specified as 'AND' or 'OR'. The Simulink based state inherits this property from the <code>Decomposition</code> property of its parent state or chart.

Stateflow.State

Use `Stateflow.State` objects to describe an operating mode of a reactive system. For more information, see “Represent Operating Modes by Using States”.

Property Name	Access	Description
ArrowSize	RW	Size of incoming transition arrows, specified as a scalar.
BadIntersection	RO	Whether the state graphically intersects a box, state, or function, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>).
Chart	RO	Chart that contains the state, specified as a <code>Stateflow.Chart</code> object.
CommentText	RW	Comment text added to the state, specified as a string scalar or character vector. This property applies only when the <code>IsExplicitlyCommented</code> property is <code>true</code> . In the Stateflow Editor, when you point to the comment badge  on the state, the text appears as a tooltip. When you set the <code>IsExplicitlyCommented</code> property to <code>false</code> , the value of <code>CommentText</code> reverts to "".
ContentPreviewEnabled	RW	Whether to display a preview of the state contents, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property applies only when the <code>IsSubchart</code> property is <code>true</code> .
Debug	RW	Debugger properties for the state, specified as a <code>Stateflow.StateDebug</code> object with these properties: <ul style="list-style-type: none"> • OnEntry — Whether to set the On State Entry breakpoint, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • OnDuring — Whether to set the During State breakpoint, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • OnExit — Whether to set the On State Exit breakpoint, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see “Set Breakpoints to Debug Charts”.

Property Name	Access	Description
Decomposition	RW	Decomposition of substates at the top level of containment in the state, specified as "EXCLUSIVE_OR" or "PARALLEL_AND". For more information, see "Specify Substate Decomposition".
Description	RW	Description for the state, specified as a string scalar or character vector.
DoNotAutogenerateEnum	RW	Whether to define the enumerated data type for the active state data output manually, specified as a numeric or logical 1 (true) or 0 (false). This property applies only when the <code>OutputMonitoringMode</code> property for the state is "ChildActivity" or "LeafStateActivity". For more information, see "Define State Activity Enumeration Type".
Document	RW	Document link for the state, specified as a string scalar or character vector.
DuringAction	RO	State during action, specified as a character vector. The value of this property depends on the <code>LabelString</code> property for the state. For more information, see "Specify Labels in States and Transitions Programmatically" on page 1-16. This property is not supported in Moore charts.
EntryAction	RO	State entry action, specified as a character vector. The value of this property depends on the <code>LabelString</code> property for the state. For more information, see "Specify Labels in States and Transitions Programmatically" on page 1-16. This property is not supported in Moore charts.
EnumTypeName	RW	Name of the enumerated data type for the active state data object for the state, specified as a string scalar or character vector. This property applies only when the <code>OutputMonitoringMode</code> property for the state is "ChildActivity" or "LeafStateActivity". For more information, see "Enum Name".
ExecutionOrder	RW	Execution order for the state in parallel (AND) decomposition, specified as an integer scalar. This property applies only when both of these conditions are satisfied: <ul style="list-style-type: none"> The <code>Type</code> property of the state is "AND". The <code>UserSpecifiedStateTransitionExecutionOrder</code> property of the chart that contains the state is true.
ExitAction	RO	State exit action, specified as a character vector. The value of this property depends on the <code>LabelString</code> property for the state. For more information, see "Specify Labels in States and Transitions Programmatically" on page 1-16. This property is not supported in Moore charts.
FontSize	RW	Font size for the state label, specified as a scalar. The <code>StateFont.Size</code> property of the chart that contains the state sets the initial value of this property.
HasOutputData	RW	Whether to create an active state data output port for the state, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Monitor State Activity Through Active State Data".

Property Name	Access	Description
Id	RO	Unique identifier, specified as an integer scalar. Unlike <code>SSIdNumber</code> , the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
InlineOption	RW	Appearance of the state functions in generated code, specified as one of these values: <ul style="list-style-type: none"> "Auto" — An internal calculation determines the appearance of state functions in generated code. "Function" — State functions are implemented as separate static functions. "Inline" — Calls to state functions are replaced by code as long as the function is not part of a recursion. For more information, see "Inline State Functions in Generated Code" (Simulink Coder).
IsCommented	RO	Whether the state is commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property is <code>true</code> when either <code>IsExplicitlyCommented</code> or <code>IsImplicitlyCommented</code> is <code>true</code> .
IsExplicitlyCommented	RW	Whether to comment out the state, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). Setting this property to <code>true</code> is equivalent to right-clicking the state and selecting Comment Out . For more information, see "Comment Out Objects in a Stateflow Chart".
IsGrouped	RW	Whether the state is a grouped state, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). When you copy and paste a grouped state, you copy not only the state but all of its contents. For more information, see "Copy and Paste by Grouping" on page 2-28.
IsImplicitlyCommented	RO	Whether the state is implicitly commented out, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). The state is implicitly commented out when you explicitly comment out an object that contains it. If the state is contained in an atomic subchart, this property is <code>false</code> unless the explicitly commented object is also contained in the atomic subchart.
IsSubchart	RW	Whether the state is a subchart, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>).
LabelString	RW	Label for the state, specified as a string scalar or character vector. For more information, see "Specify Labels in States and Transitions Programmatically" on page 1-16.

Property Name	Access	Description
LoggingInfo	RW	<p>Signal logging properties for the state, specified as a <code>Stateflow.SigLoggingInfo</code> object with these properties:</p> <ul style="list-style-type: none"> • DataLogging — Whether to enable signal logging, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • DecimateData — Whether to limit the amount of logged data, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • Decimation — Decimation interval, specified as an integer scalar. This property applies only when the <code>DecimateData</code> property is <code>true</code>. • LimitDataPoints — Whether to limit the number of data points to log, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). • MaxPoints — Maximum number of data points to log, specified as an integer scalar. This property applies only when the <code>LimitDataPoints</code> property is <code>true</code>. • NameMode — Source of the signal name, specified as "SignalName" or "Custom". • LoggingName — Custom signal name, specified as a string scalar or character vector. This property applies only when the <code>NameMode</code> property is "Custom". <p>Signal logging saves the self activity of the state to the MATLAB workspace during simulation. For more information, see “Log Simulation Output for States and Data”.</p>
Machine	RO	Machine that contains the state, specified as a <code>Stateflow.Machine</code> object.
MooreAction	RO	State action in a Moore chart, specified as a character vector. The value of this property depends on the <code>LabelString</code> property for the state. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16. This property is supported only in Moore charts. For more information, see “Design Guidelines for Moore Charts”.
Name	RW	Name of the state, specified as a string scalar or character vector.
OnAction	RO	<p>State on actions, specified as a cell array of character vectors in the form</p> <pre>{'trigger1','action1',...,'triggerN','actionN'}</pre> <p>The value of this property depends on the <code>LabelString</code> property for the state. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16. This property is not supported in Moore charts.</p>
OutputData	RO	Active state data object for the state, specified as a <code>Stateflow.Data</code> object. This property applies only when the <code>HasOutputData</code> property for the state is <code>true</code> .
OutputMonitoringMode	RW	Monitoring mode for the active state output data, specified as "SelfActivity", "ChildActivity", or "LeafStateActivity".

Property Name	Access	Description
OutputPortName	RW	Name of the active state data object for the state, specified as a string scalar or character vector. This property applies only when the <code>HasOutputData</code> property for the state is <code>true</code> .
Path	RO	Location of the parent of the state in the model hierarchy, specified as a character vector.
Position	RW	Position and size of the state, specified as a four-element numeric vector of the form <code>[left top width height]</code> .
SSIdNumber	RO	Session-independent identifier, specified as an integer scalar. Use this property to distinguish the state from other objects in the model.
Subviewer	RO	Subviewer for the state, specified as a <code>Stateflow.Chart</code> , <code>Stateflow.State</code> , or <code>Stateflow.Box</code> object. The subviewer is the chart or subchart where you can graphically view the state.
Tag	RW	User-defined tag for the state, specified as data of any type.
TestPoint	RW	Whether to set the state as a test point, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). You can monitor testpoints with a floating scope during simulation. You can also log test point values to the MATLAB workspace. For more information, see “Monitor Test Points in Stateflow Charts”.
Type	RO	Decomposition of sibling states, specified as <code>'AND'</code> or <code>'OR'</code> . The state inherits this property from the <code>Decomposition</code> property of its parent state or chart.

Stateflow.StateTransitionTableChart

Use a `Stateflow.StateTransitionTableChart` object to represent a finite state machine for sequential modal logic in tabular format. Instead of drawing states and transitions in a Stateflow chart, you can use a State Transition Table block to model a state machine in a concise, compact format that requires minimal maintenance of graphical objects. For more information, see “Use State Transition Tables to Express Sequential Logic in Tabular Form”.

Property Name	Access	Description
ActionLanguage	RW	Action language used to program the state transition table, specified as <code>"MATLAB"</code> or <code>"C"</code> . For more information, see “Differences Between MATLAB and C as Action Language Syntax”.
ChartColor	RW	Background color for the chart that is automatically generated for the state transition table, specified as a three-element numeric vector of the form <code>[red green blue]</code> that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.
ChartUpdate	RW	Activation method for the state transition table, specified as <code>"CONTINUOUS"</code> , <code>"DISCRETE"</code> , or <code>"INHERITED"</code> . For more information, see “Update Method”.

Property Name	Access	Description
Debug	RW	Debugger properties for the state transition table, specified as a <code>Stateflow.ChartDebug</code> object with this property: <ul style="list-style-type: none"> Breakpoints.OnEntry — Whether to set the On Chart Entry breakpoint, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see “Set Breakpoints to Debug Charts”.
Description	RW	Description for the state transition table, specified as a string scalar or character vector.
Dirty	RW	Whether the state transition table has changed after being opened or saved, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>).
DoNotAutogenerateEnum	RW	Whether to define the enumerated data type for the active state data output manually, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see “Define State Activity Enumeration Type”.
Document	RW	Document link for the state transition table, specified as a string scalar or character vector.
Editor	RO	Editor for the state transition table, specified as a <code>Stateflow.Editor</code> object. You can use this object to control the position, size, and magnification level of the Stateflow Editor window.
EmlDefaultFimath	RW	Default fimath properties for the state transition table, specified as one of these values: <ul style="list-style-type: none"> “Same as MATLAB Default” — Use the same fimath properties as the current default fimath object. “Other:UserSpecified” — Use the <code>InputFimath</code> property to specify the default fimath object. This property applies only when the <code>ActionLanguage</code> property of the state transition table is “MATLAB”.
EnableBitOps	RW	Whether to use bit operations in state and transition actions in the state transition table, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property applies only to state transition tables that use C as the action language. For more information, see “Enable C-bit operations”.
EnableNonTerminalStates	RW	Whether to enable super step semantics for the state transition table, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see “Super Step Semantics”.
EnableZeroCrossings	RW	Whether to enable zero-crossing detection on state transitions in the state transition table, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property applies only when the <code>ChartUpdate</code> property for the state transition table is set to “CONTINUOUS”. For more information, see “Disable Zero-Crossing Detection”.
EnumTypeName	RW	Name of the enumerated data type for the active state data object for the state transition table, specified as a string scalar or character vector. For more information, see “Enum Name”.

Property Name	Access	Description
ExecuteAtInitialization	RW	Whether to initialize the state configuration of the state transition table at time zero instead of at the first input event, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see “Execution of a Chart at Initialization”.
HasOutputData	RW	Whether to create an active state data output port for the state transition table, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see “Monitor State Activity Through Active State Data”.
Iced	RO	Whether the state transition table is locked, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). This property is equivalent to the property <code>Locked</code> , but is used internally to prevent changes in the state transition table during simulation.
Id	RO	Unique identifier, specified as an integer scalar. Use this property to distinguish the state transition table from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
InitializeOutput	RW	Whether to initialize the output data every time the state transition table wakes up, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). For more information, see “Initialize outputs every time chart wakes up”.
InputFimath	RW	Default fimath object, specified as a string scalar or character vector. When the <code>EmlDefaultFimath</code> property for the state transition table is “ <code>Other:UserSpecified</code> ”, you can use this property to: <ul style="list-style-type: none"> • 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 <code>ActionLanguage</code> property of the state transition table is “ <code>MATLAB</code> ”.
JunctionColor	RW	Color for junctions in the chart that is automatically generated for the state transition table, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.
Locked	RW	Whether the state transition table is locked, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). Enable this property to prevent changes in the state transition table.
Machine	RO	Machine that contains the state transition table, specified as a <code>Stateflow.Machine</code> object.
Name	RW	Name of the state transition table, specified as a string scalar or character vector.
NonTerminalMaxCounts	RW	Maximum number of transitions the state transition table can take in one super step, specified as an integer scalar. This property applies only when the <code>EnableNonTerminalStates</code> property for the state transition table is <code>true</code> .


Property Name	Access	Description
NonTerminalUnstableBehavior	RW	<p>Behavior if a super step for the state transition table exceeds the maximum number of transitions specified in the <code>NonTerminalMaxCounts</code> property before reaching a stable state, specified as one of these values:</p> <ul style="list-style-type: none"> "Proceed" — The state transition table goes to sleep with the last active state configuration. "Throw Error" — The state transition table generates an error. <p>This property applies only when the <code>EnableNonTerminalStates</code> property for the state transition table is <code>true</code>.</p>
OutputData	RO	Active state data object for the state transition table, specified as a <code>Stateflow.Data</code> object. This property applies only when the <code>HasOutputData</code> property for the state transition table is <code>true</code> .
OutputMonitoringMode	RW	Monitoring mode for the active state output data, specified as "ChildActivity" or "LeafStateActivity".
OutputPortName	RW	Name of the active state data object for the state transition table, specified as a string scalar or character vector. This property applies only when the <code>HasOutputData</code> property for the state transition table is <code>true</code> .
Path	RO	Location of the state transition table in the model hierarchy, specified as a character vector.
SampleTime	RW	Sample time for activating the state transition table, specified as a string scalar or character vector. This property applies only when the <code>ChartUpdate</code> property for the state transition table is "DISCRETE".
SaturateOnIntegerOverflow	RW	Whether the data in the state transition table saturates on integer overflow, specified as a numeric or logical 1 (<code>true</code>) or 0 (<code>false</code>). When this property is disabled, the data in the state transition table wraps on integer overflow. For more information, see "Handle Integer Overflow for Chart Data".
StateColor	RW	Color for the states in the chart that is automatically generated for the state transition table, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.
StateFont	RW	Font for the state labels in the chart that is automatically generated for the state transition table, specified as a <code>Stateflow.STTStateFont</code> object with these properties: <ul style="list-style-type: none"> Name — Font name, specified as a string scalar or character vector. Angle — Font angle, specified as "NORMAL" or "ITALIC". Weight — Font weight, specified as "NORMAL" or "BOLD". Size — Default font size for new states, specified as a scalar.
StateLabelColor	RW	Color for the state labels in the chart that is automatically generated for the state transition table, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.

Property Name	Access	Description
StateMachineType	RW	State machine semantics implemented by the state transition table, specified as "Classic", "Mealy", or "Moore". For more information, see "Overview of Mealy and Moore Machines".
StatesWhenEnabling	RW	Behavior of the states when a function-call input event reenables the state transition table, specified as one of these values: <ul style="list-style-type: none"> " " — The state transition table does not contain function-call input events. "held" — The state transition table maintains the most recent values of the states. "reset" — The state transition table reverts to the initial conditions of the states. For more information, see "Control States in Charts Enabled by Function-Call Input Events".
SupportVariableSizing	RW	Whether the state transition table supports variable-size data, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Declare Variable-Size Data in Stateflow Charts".
Tag	RW	User-defined tag for the state transition table, specified as data of any type.
TransitionColor	RW	Color for transitions in the chart that is automatically generated for the state transition table, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.
TransitionFont	RW	Font for the transition labels in the chart that is automatically generated for the state transition table, specified as a Stateflow.STTTransFont object with these properties: <ul style="list-style-type: none"> Name — Font name, specified as a string scalar or character vector. Angle — Font angle, specified as "NORMAL" or "ITALIC". Weight — Font weight, specified as "NORMAL" or "BOLD". Size — Default font size for new transitions, specified as a scalar.
TransitionLabelColor	RW	Color for the transition labels in the chart that is automatically generated for the state transition table, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.
TreatAsFi	RW	Inherited Simulink signals to treat as Fixed-Point Designer <code>fi</code> objects, specified as one of these values: <ul style="list-style-type: none"> "Fixed-point" — The state transition table treats all fixed-point inputs as <code>fi</code> objects. "Fixed-point & Integer" — The state transition table treats all fixed-point and integer inputs as <code>fi</code> objects. This property applies only when the ActionLanguage property of the state transition table is "MATLAB".

Property Name	Access	Description
Visible	RW	Whether the Stateflow Editor window is displaying the state transition table, specified as a numeric or logical 1 (true) or 0 (false).

Stateflow.Transition

Use `Stateflow.Transition` objects to create transitions from one operating mode to another. For more information, see “Transition Between Operating Modes”.


Property Name	Access	Description
ArrowSize	RW	Size of the transition arrow at the destination, specified as a scalar. When you change the destination of the transition, this property resets to the value of the <code>ArrowSize</code> property of the new destination.
Chart	RO	Chart that contains the transition, specified as a <code>Stateflow.Chart</code> object.
CommentText	RW	Comment text added to the transition, specified as a string scalar or character vector. This property applies only when the <code>IsExplicitlyCommented</code> property is true. In the Stateflow Editor, when you point to the comment badge  on the transition, the text appears as a tooltip. When you set the <code>IsExplicitlyCommented</code> property to false, the value of <code>CommentText</code> reverts to "".
Condition	RO	Transition condition, specified as a character vector. The value of this property depends on the <code>LabelString</code> property for the transition. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16.
ConditionAction	RO	Transition condition action, specified as a character vector. The value of this property depends on the <code>LabelString</code> property for the transition. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16.
Debug	RW	Debugger properties for the transition, specified as a <code>Stateflow.TransDebug</code> object with these properties: <ul style="list-style-type: none"> • Breakpoints.WhenTested — Whether to set the When Transition is Tested breakpoint, specified as a numeric or logical 1 (true) or 0 (false). • Breakpoints.WhenValid — Whether to set the When Transition is Valid breakpoint, specified as a numeric or logical 1 (true) or 0 (false). For more information, see “Set Breakpoints to Debug Charts”.
Description	RW	Description for the transition, specified as a string scalar or character vector.

Property Name	Access	Description
Destination	RW	Destination of the transition, specified as an empty array or a Stateflow API object of one of these types: <ul style="list-style-type: none"> • Stateflow.AtomicSubchart • Stateflow.Junction • Stateflow.SimulinkBasedState • Stateflow.State
DestinationEndPoint	RW	Position of the transition endpoint at its destination, specified as a two-element numeric vector [x y] of coordinates relative to the upper left corner of the chart.
DestinationOClock	RW	Location of the transition endpoint at its destination, specified as a scalar between 0 and 12 that describes a clock position.
Document	RW	Document link for the transition, specified as a string scalar or character vector.
ExecutionOrder	RW	Execution order for the transition when its source is active, specified as an integer scalar. This property applies only when the UserSpecifiedStateTransitionExecutionOrder property of the chart that contains the transition is true. For more information, see "Transition Evaluation Order".
FontSize	RW	Font size for the transition label, specified as a scalar. The TransitionFont.Size property of the chart that contains the transition sets the initial value of this property.
Id	RO	Unique identifier, specified as an integer scalar. Unlike SSIDNumber, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
IsCommented	RO	Whether the transition is commented out, specified as a numeric or logical 1 (true) or 0 (false). This property is true when either IsExplicitlyCommented or IsImplicitlyCommented is true.
IsExplicitlyCommented	RW	Whether to comment out the transition, specified as a numeric or logical 1 (true) or 0 (false). Setting this property to true is equivalent to right-clicking the transition and selecting Comment Out . For more information, see "Comment Out Objects in a Stateflow Chart".
IsImplicitlyCommented	RO	Whether the transition is implicitly commented out, specified as a numeric or logical 1 (true) or 0 (false). The transition is implicitly commented out when you explicitly comment out an object that contains it or when you comment out its source or destination. If the transition is contained in an atomic subchart or an atomic box, this property is false unless the explicitly commented object is also contained in the atomic subchart or atomic box.
IsVariant	RW	Whether the transition is a variant transition, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Control Indicator Lamp Dimmer Using Variant Conditions".
LabelPosition	RW	Position and size of the transition label, specified as a four-element numeric vector of the form [left top width height].

Property Name	Access	Description
LabelString	RW	Label for the transition, specified as a string scalar or character vector. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16.
Machine	RO	Machine that contains the transition, specified as a <code>Stateflow.Machine</code> object.
MidPoint	RW	Position of the midpoint of the transition, specified as a two-element numeric vector $[x \ y]$ of coordinates relative to the upper left corner of the chart.
Path	RO	Location of the parent of the transition in the model hierarchy, specified as a character vector.
SSIdNumber	RO	Session-independent identifier, specified as an integer scalar. Use this property to distinguish the transition from other objects in the model.
Source	RW	Source of the transition, specified as an empty array or a Stateflow API object of one of these types: <ul style="list-style-type: none"> • <code>Stateflow.AtomicSubchart</code> • <code>Stateflow.Junction</code> • <code>Stateflow.SimulinkBasedState</code> • <code>Stateflow.State</code>
SourceEndPoint	RW	Position of the transition endpoint at its source, specified as a two-element numeric vector $[x \ y]$ of coordinates relative to the upper left corner of the chart.
SourceOClock	RW	Location of the transition endpoint at its source, specified as a scalar between 0 and 12 that describes a clock position.
Subviewer	RO	Subviewer for the transition, specified as a <code>Stateflow.Chart</code> , <code>Stateflow.State</code> , <code>Stateflow.Box</code> , or <code>Stateflow.Function</code> object. The subviewer is the chart or subchart where you can graphically view the transition.
Tag	RW	User-defined tag for the transition, specified as data of any type.
TransitionAction	RO	Transition action, specified as a character vector. The value of this property depends on the <code>LabelString</code> property for the transition. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16.
Trigger	RO	Transition trigger, specified as a character vector. The value of this property depends on the <code>LabelString</code> property for the transition. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16.

Stateflow.TruthTable

Use `Stateflow.TruthTable` objects to create truth table functions that implement combinatorial logic design in a tabular format. You can use truth table functions to model decision making for fault detection and management and mode switching. For more information, see “Use Truth Tables to Model Combinatorial Logic”.

Property Name	Access	Description
ActionTable	RW	Action table for the truth table, specified as a cell array of character vectors.
BadIntersection	RO	Whether the truth table graphically intersects a box, state, or function, specified as a numeric or logical 1 (true) or 0 (false).
Chart	RO	Chart that contains the truth table, specified as a Stateflow.Chart object.
CommentText	RW	Comment text added to the truth table, specified as a string scalar or character vector. This property applies only when the IsExplicitlyCommented property is true. In the Stateflow Editor, when you point to the comment badge  on the truth table, the text appears as a tooltip. When you set the IsExplicitlyCommented property to false, the value of CommentText reverts to "".
ConditionTable	RW	Condition table for the truth table, specified as a cell array of character vectors.
Debug	RW	Debugger properties for the truth table, specified as a Stateflow.FunctionDebug object with this property: <ul style="list-style-type: none"> Breakpoints.OnDuring — Whether to set the During Function Call breakpoint, specified as a numeric or logical 1 (true) or 0 (false). This property applies only when both the Language property of the truth table and the ActionLanguage of the chart that contains the truth table are "C". For more information, see "Set Breakpoints to Debug Charts".
Description	RW	Description for the truth table, specified as a string scalar or character vector.
Document	RW	Document link for the truth table, specified as a string scalar or character vector.
EmlDefaultFimath	RW	Default fimath properties for the truth table, specified as one of these values: <ul style="list-style-type: none"> "Same as MATLAB Default" — Use the same fimath properties as the current default fimath object. "Other:UserSpecified" — Use the InputFimath property to specify the default fimath object. This property applies only when the Language property of the truth table is "MATLAB" and the ActionLanguage of the chart that contains the truth table is "C".
FontSize	RW	Font size for the truth table label, specified as a scalar. The StateFont.Size property of the chart that contains the truth table sets the initial value of this property.
Id	RO	Unique identifier, specified as an integer scalar. Unlike SSIdNumber, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Property Name	Access	Description
InlineOption	RW	<p>Appearance of the truth table in generated code, specified as one of these values:</p> <ul style="list-style-type: none"> "Auto" — An internal calculation determines the appearance of the truth table in generated code. "Function" — The truth table is implemented as a separate C function. "Inline" — Calls to the truth table are replaced by code as long as the truth table is not part of a recursion. <p>For more information, see "Inline State Functions in Generated Code" (Simulink Coder).</p>
InputFimath	RW	<p>Default fimath object, specified as a string scalar or character vector. When the EmlDefaultFimath property for the truth table is "Other:UserSpecified", you can use this property to:</p> <ul style="list-style-type: none"> Enter an expression that constructs a fimath object. Enter the variable name for a fimath object in the MATLAB or model workspace. <p>This property applies only when the Language property of the truth table is "MATLAB" and the ActionLanguage of the chart that contains the truth table is "C".</p>
IsCommented	RO	Whether the truth table is commented out, specified as a numeric or logical 1 (true) or 0 (false). This property is true when either IsExplicitlyCommented or IsImplicitlyCommented is true.
IsExplicitlyCommented	RW	Whether to comment out the truth table, specified as a numeric or logical 1 (true) or 0 (false). Setting this property to true is equivalent to right-clicking the truth table and selecting Comment Out . For more information, see "Comment Out Objects in a Stateflow Chart".
IsImplicitlyCommented	RO	Whether the truth table is implicitly commented out, specified as a numeric or logical 1 (true) or 0 (false). The truth table is implicitly commented out when you explicitly comment out an object that contains it. If the truth table is contained in an atomic subchart or an atomic box, this property is false unless the explicitly commented object is also contained in the atomic subchart or atomic box.
LabelString	RW	Label for the truth table, specified as a string scalar or character vector.
Language	RW	Action language used to program the truth table, specified as "MATLAB" or "C". The option "C" is supported only in truth tables in charts that use C as the action language. For more information, see "Differences Between MATLAB and C as Action Language Syntax".
Machine	RO	Machine that contains the truth table, specified as a Stateflow.Machine object.
Name	RW	Name of the truth table, specified as a string scalar or character vector.
OverSpecDiagnostic	RW	Level of diagnostic action when the truth table is overspecified, specified as "Error", "Warning", or "None". For more information, see "Correct Overspecified and Underspecified Truth Tables".

Property Name	Access	Description
Path	RO	Location of the parent of the truth table in the model hierarchy, specified as a character vector.
Position	RW	Position and size of the truth table, specified as a four-element numeric vector of the form [left top width height].
SSIdNumber	RO	Session-independent identifier, specified as an integer scalar. Use this property to distinguish the truth table from other objects in the model.
SaturateOnIntegerOverflow	RW	Whether the data in the truth table saturates on integer overflow, specified as a numeric or logical 1 (true) or 0 (false). When this property is disabled, the data in the truth table wraps on integer overflow. This property applies only when the Language property of the truth table is "MATLAB" and the ActionLanguage of the chart that contains the truth table is "C". For more information, see "Handle Integer Overflow for Chart Data".
Subviewer	RO	Subviewer for the truth table, specified as a Stateflow.Chart, Stateflow.State, Stateflow.Box, or Stateflow.Function object. The subviewer is the chart or subchart where you can graphically view the truth table.
Tag	RW	User-defined tag for the truth table, specified as data of any type.
UnderSpecDiagnostic	RW	Level of diagnostic action when the truth table is underspecified, specified as "Error", "Warning", or "None". For more information, see "Correct Overspecified and Underspecified Truth Tables".

Stateflow.TruthTableChart

Use Stateflow.TruthTableChart objects to implement combinatorial logic design in a tabular format. You can use Truth Table blocks to model decision making for fault detection and management and mode switching. For more information, see "Use Truth Tables to Model Combinatorial Logic".

Property Name	Access	Description
ActionTable	RW	Action table for the truth table, specified as a cell array of character vectors.
ChartUpdate	RW	Activation method for the truth table, specified as "CONTINUOUS", "DISCRETE", or "INHERITED". For more information, see "Update Method".
ConditionTable	RW	Condition table for the truth table, specified as a cell array of character vectors.
Description	RW	Description for the truth table, specified as a string scalar or character vector.
Dirty	RW	Whether the truth table has changed after being opened or saved, specified as a numeric or logical 1 (true) or 0 (false).
Document	RW	Document link for the truth table, specified as a string scalar or character vector.

Property Name	Access	Description
EmlDefaultFimath	RW	Default fimath properties for the truth table, specified as one of these values: <ul style="list-style-type: none"> "Same as MATLAB Default" — Use the same fimath properties as the current default fimath object. "Other:UserSpecified" — Use the InputFimath property to specify the default fimath object.
Iced	RO	Whether the truth table is locked, specified as a numeric or logical 1 (true) or 0 (false). This property is equivalent to the property Locked, but is used internally to prevent changes in the truth table during simulation.
Id	RO	Unique identifier, specified as an integer scalar. Use this property to distinguish the truth table from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.
InputFimath	RW	Default fimath object, specified as a string scalar or character vector. When the EmlDefaultFimath property for the truth table is "Other:UserSpecified", you can use this property to: <ul style="list-style-type: none"> Enter an expression that constructs a fimath object. Enter the variable name for a fimath object in the MATLAB or model workspace.
Locked	RW	Whether the truth table is locked, specified as a numeric or logical 1 (true) or 0 (false). Enable this property to prevent changes in the truth table.
Machine	RO	Machine that contains the truth table, specified as a Stateflow.Machine object.
Name	RW	Name of the truth table, specified as a string scalar or character vector.
OverSpecDiagnostic	RW	Level of diagnostic action when the truth table is overspecified, specified as "Error", "Warning", or "None". For more information, see "Correct Overspecified and Underspecified Truth Tables".
Path	RO	Location of the truth table in the model hierarchy, specified as a character vector.
SampleTime	RW	Sample time for activating the truth table, specified as a string scalar or character vector. This property applies only when the ChartUpdate property for the truth table is "DISCRETE".
SaturateOnIntegerOverflow	RW	Whether the data in the truth table saturates on integer overflow, specified as a numeric or logical 1 (true) or 0 (false). When this property is disabled, the data in the truth table wraps on integer overflow. For more information, see "Handle Integer Overflow for Chart Data".
SupportVariableSizing	RW	Whether the truth table supports variable-size data, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Declare Variable-Size Data in Stateflow Charts".
Tag	RW	User-defined tag for the truth table, specified as data of any type.

Property Name	Access	Description
TreatAsFi	RW	Inherited Simulink signals to treat as Fixed-Point Designer <code>fi</code> objects, specified as one of these values: <ul style="list-style-type: none">"Fixed-point" — The truth table treats all fixed-point inputs as <code>fi</code> objects."Fixed-point & Integer" — The truth table treats all fixed-point and integer inputs as <code>fi</code> objects.
UnderSpecDiagnostic	RW	Level of diagnostic action when the truth table is underspecified, specified as "Error", "Warning", or "None". For more information, see "Correct Overspecified and Underspecified Truth Tables".

See Also

`sfclipboard` | `sfnew` | `sfroot`

More About

- “Create Charts by Using the Stateflow API” on page 1-19
- “Create and Delete Stateflow Objects” on page 1-13
- “Modify Properties and Call Functions of Stateflow Objects” on page 1-11

API Object Reference

Stateflow.Annotation

Annotation in chart, state, box, or function

Description

Use `Stateflow.Annotation` objects to include descriptive comments in your chart. Annotations can contain any combination of:

- Text
- Images
- Equations using TeX commands
- Hyperlinks that open a website or perform MATLAB functions

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

Creation

Syntax

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

Description

`annotation = Stateflow.Annotation(parent)` creates a `Stateflow.Annotation` object in a parent chart, state, box, or graphical function.

Input Arguments

parent — Parent for new annotation

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object

Parent for the new annotation, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.Function`
- `Stateflow.State`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content**Text — Text for annotation**

"?" (default) | string scalar | character vector

Text for the annotation, specified as a string scalar or character vector.

Alignment — Alignment of text

"LEFT" (default) | "CENTER" | "RIGHT"

Alignment of the annotation text, specified as "LEFT", "CENTER", or "RIGHT".

Interpretation — Format of text

"OFF" (default) | "RICH" | "TEX"

Format of the annotation text, specified as "OFF", "RICH", or "TEX".

PlainText — Text without formatting

character vector

This property is read-only.

Annotation text without formatting, specified as a character vector.

IsImage — Whether annotation contains image

false or 0 (default) | true or 1

This property is read-only.

Whether the annotation contains an image, specified as a numeric or logical 1 (true) or 0 (false).

Graphical Appearance**Position — Position and size of annotation box**

[0 0 8 16] (default) | [left top width height]

Position and size of annotation box, specified as a four-element numeric vector of the form [left top width height].

InternalMargins — Space between text and border of annotation box

[0 0 0 0] (default) | [left top right bottom]

Space between the text and the border of the annotation box, specified as a four-element numeric vector of the form [left top right bottom].

DropShadow — Whether to display a drop shadow around annotation box

false or 0 (default) | true or 1

Whether to display a drop shadow around the annotation box, specified as a numeric or logical 1 (true) or 0 (false).

FixedHeight — Whether to fix height of annotation box

false or 0 (default) | true or 1

Whether to fix the height of the annotation box, specified as a numeric or logical 1 (true) or 0 (false).

- `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.

FixedWidth — Whether to fix width of annotation box`false` or 0 (default) | `true` or 1

Whether to fix the width of the annotation box, specified as a numeric or logical 1 (`true`) or 0 (`false`).

- `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.

BackgroundColor — Background color`[1 1 1]` (default) | `[red green blue]`

Background color for the annotation, specified as a three-element numeric vector of the form `[red green blue]` that specifies the red, green, and blue values. Each element must be in the range between 0 and 1. This property applies only when the `AutoBackgroundColor` property is `false`.

ForegroundColor — Foreground color`[0 0 0]` (default) | `[red green blue]`

Foreground color for the annotation, specified as a three-element numeric vector of the form `[red green blue]` that specifies the red, green, and blue values. Each element must be in the range between 0 and 1. This property applies only when the `AutoForegroundColor` property is `false`.

AutoBackgroundColor — Whether to use default background color`true` or 1 (default) | `false` or 0

Whether to use the default background color, specified as a numeric or logical 1 (`true`) or 0 (`false`).

- `true` — Use the default color specified by the `ChartColor` property of the chart that contains the annotation.
- `false` — Use the color specified by the `BackgroundColor` property of the annotation.

AutoForegroundColor — Whether to use default foreground color`true` or 1 (default) | `false` or 0

Whether to use the default foreground color, specified as a numeric or logical 1 (`true`) or 0 (`false`).

- `true` — Use the default color specified by the `StateLabelColor` property of the chart that contains the annotation.
- `false` — Use the color specified by the `ForegroundColor` property of the annotation.

Font — Font for annotation text`Stateflow.NoteFont` object

Font for the annotation text, specified as a `Stateflow.NoteFont` object with these properties:

- **Name** — Font name, specified as a character vector. This property is read-only. The `StateFont.Name` property of the chart that contains the annotation sets the value of this property.
- **Angle** — Font angle, specified as "NORMAL" or "ITALIC".

- **Weight** — Font weight, specified as "NORMAL" or "BOLD".
- **Size** — Font size, specified as a scalar.

Example: `annotation.Font.Angle = "ITALIC";`

Example: `annotation.Font.Weight = "BOLD";`

Example: `annotation.Font.Size = 8;`

Callbacks

ClickFcn — Callback on click

"" (default) | string scalar | character vector

Callback on click, specified as a string scalar or character vector. This callback contains MATLAB code to execute when you click the annotation.

LoadFcn — Callback at model load

"" (default) | string scalar | character vector

Callback at model load, specified as a string scalar or character vector. This callback contains MATLAB code to execute when you load the model that contains the annotation.

DeleteFcn — Callback at delete

"" (default) | string scalar | character vector

Callback at delete, specified as a string scalar or character vector. This callback contains MATLAB code to execute before you delete the annotation.

UseDisplayTextAsClickCallback — Whether to use annotation text as callback

false or 0 (default) | true or 1

Whether to use the annotation text as a callback, specified as a numeric or logical 1 (true) or 0 (false). When this property is enabled, the contents of the Text property is used as the callback when you click the annotation.

Hierarchy

Chart — Chart that contains annotation

Stateflow.Chart object

This property is read-only.

Chart that contains the annotation, specified as a Stateflow.Chart object.

Subviewer — Subviewer for annotation

Stateflow.Chart object | Stateflow.State object | Stateflow.Box object | Stateflow.Function object

This property is read-only.

Subviewer for the annotation, specified as a Stateflow.Chart, Stateflow.State, Stateflow.Box, or Stateflow.Function object. The subviewer is the chart or subchart where you can graphically view the annotation.

Machine — Machine that contains annotation`Stateflow.Machine` object

This property is read-only.

Machine that contains the annotation, specified as a `Stateflow.Machine` object.

Path — Location of parent in model hierarchy`character vector`

This property is read-only.

Location of the parent of the annotation in the model hierarchy, specified as a character vector.

Identification**Description — Description**`""` (default) | string scalar | character vector

Description for the annotation, specified as a string scalar or character vector.

Document — Document link`""` (default) | string scalar | character vector

Document link for the annotation, specified as a string scalar or character vector.

Tag — User-defined tag`[]` (default) | any data type | ...

User-defined tag for the annotation, specified as data of any type.

Id — Unique identifier`scalar`

This property is read-only.

Unique identifier, specified as an integer scalar. Use this property to distinguish the annotation from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>getParent</code>	Identify parent of object
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>fitToView</code>	Zoom in on graphical object
<code>setImage</code>	Insert image into annotation

Examples**Add Text Annotation to Chart**

Add an annotation in the chart `ch`. Set its content to `"This is an annotation."`

```
annotation = Stateflow.Annotation(ch);  
annotation.Text = "This is an annotation";
```

Add Image Annotation to Chart

Add an annotation in the chart `ch`. Use the file `myImageFile.png`, which is located in the folder `myfolder/annotation_images`, as the image for the annotation.

```
annotation = Stateflow.Annotation(ch);  
setImage(annotation, ...  
    fullfile("myfolder", "annotation_images", "myImageFile.png"));
```

Version History

Introduced in R2017b

See Also

[Stateflow.Box](#) | [Stateflow.Chart](#) | [Stateflow.Function](#) | [Stateflow.State](#)

Topics

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

“Add Descriptive Comments in a Chart”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.AtomicBox

Atomic box in chart, state, box, or function

Description

Use `Stateflow.AtomicBox` objects to encapsulate graphical, truth table, MATLAB, and Simulink functions in a separate namespace. Atomic boxes allow for:

- Faster simulation after making small changes to a function in a chart with many states or levels of hierarchy
- Reuse of the same functions across multiple charts and models
- Ease of team development for people working on different parts of the same chart
- Manual inspection of generated code for a specific function in a chart

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

Creation

Syntax

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

Description

`atomicBox = Stateflow.AtomicBox(parent)` creates a `Stateflow.AtomicBox` object in a parent chart, state, box, or graphical function.

Input Arguments

parent — Parent for new atomic box

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object

Parent for the new atomic box, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.Function`
- `Stateflow.State`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Name — Name of atomic box

"" (default) | string scalar | character vector

Name of the atomic box, specified as a string scalar or character vector.

LabelString — Label for atomic box

"?" (default) | string scalar | character vector

Label for the atomic box, specified as a string scalar or character vector.

IsLink — Whether atomic box is a library link

true or 1 | false or 0

This property is read-only.

Whether the atomic box is a library link, specified as a numeric or logical 1 (true) or 0 (false).

IsExplicitlyCommented — Whether to comment out atomic box

false or 0 (default) | true or 1

Whether to comment out the atomic box, specified as a numeric or logical 1 (true) or 0 (false). Setting this property to true is equivalent to right-clicking the atomic box and selecting **Comment Out**. For more information, see “Comment Out Objects in a Stateflow Chart”.

IsImplicitlyCommented — Whether atomic box is implicitly commented out

true or 1 | false or 0

This property is read-only.

Whether the atomic box is implicitly commented out, specified as a numeric or logical 1 (true) or 0 (false). The atomic box is implicitly commented out when you explicitly comment out an object that contains it. If the atomic box is contained in an atomic subchart or another atomic box, this property is false unless the explicitly commented object is also contained in the atomic subchart or atomic box.

IsCommented — Whether atomic box is commented out


true or 1 | false or 0

This property is read-only.

Whether the atomic box is commented out, specified as a numeric or logical 1 (true) or 0 (false). This property is true when either `IsExplicitlyCommented` or `IsImplicitlyCommented` is true.

CommentText — Comment text

"" (default) | string scalar | character vector

Comment text for the atomic box, specified as a string scalar or character vector. This property applies only when the `IsExplicitlyCommented` property is true. In the Stateflow Editor, when you point to the comment badge  on the atomic box, the text appears as a tooltip. When you set the `IsExplicitlyCommented` property to false, the value of `CommentText` reverts to "".

Graphical Appearance

Position — Position and size of atomic box

[0 0 90 60] (default) | [left top width height]

Position and size of the atomic box, specified as a four-element numeric vector of the form [left top width height].

BadIntersection — Whether atomic box intersects a box, state, or function

true or 1 | false or 0

This property is read-only.

Whether the atomic box graphically intersects a box, state, or function, specified as a numeric or logical 1 (true) or 0 (false).

ContentPreviewEnabled — Whether to display preview of atomic box contents

false or 0 (default) | true or 1

Whether to display a preview of the atomic box contents, specified as a numeric or logical 1 (true) or 0 (false).

FontSize — Font size for atomic box label

scalar

Font size for the atomic box label, specified as a scalar. The `StateFont.Size` property of the chart that contains the atomic box sets the initial value of this property.

Hierarchy

Chart — Chart that contains atomic box

`Stateflow.Chart` object

This property is read-only.

Chart that contains the atomic box, specified as a `Stateflow.Chart` object.

Subchart — Contents of atomic box

`Stateflow.Chart` object

This property is read-only.

Contents of the atomic box, specified as a `Stateflow.Chart` object. Use this object to add children, such as states and transitions, to the atomic box.

Subviewer — Subviewer for atomic box

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object

This property is read-only.

Subviewer for the atomic box, specified as a `Stateflow.Chart`, `Stateflow.State`, `Stateflow.Box`, or `Stateflow.Function` object. The subviewer is the chart or subchart where you can graphically view the atomic box.

Machine — Machine that contains atomic box

Stateflow.Machine object

This property is read-only.

Machine that contains the atomic box, specified as a Stateflow.Machine object.

Path — Location of parent in model hierarchy

character vector

This property is read-only.

Location of the parent of the atomic box in the model hierarchy, specified as a character vector.

Identification**Description — Description**

"" (default) | string scalar | character vector

Description for the atomic box, specified as a string scalar or character vector.

Document — Document link

"" (default) | string scalar | character vector

Document link for the atomic box, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the atomic box, specified as data of any type.

SSIdNumber — Session-independent identifier

scalar

This property is read-only.

Session-independent identifier, specified as an integer scalar. Use this property to distinguish the atomic box from other objects in the model.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Unlike SSIdNumber, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

getParent	Identify parent of object
getReferences	Identify references to symbol name
renameReferences	Rename symbol and update references to symbol name
commentedBy	Identify objects that implicitly comment out a graphical object

<code>getMappingForSymbol</code>	Get mapping for symbol in atomic subchart, atomic box, or Simulink based state
<code>setMappingForSymbol</code>	Set mapping for symbol in atomic subchart, atomic box, or Simulink based state
<code>clearMappingForSymbol</code>	Clear mapping for symbol in atomic subchart, atomic box, or Simulink based state
<code>disableMappingForSymbol</code>	Disable input event in atomic subchart or box
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>highlight</code>	Highlight graphical object
<code>fitToView</code>	Zoom in on graphical object

Examples

Add Atomic Box to Chart

Add an atomic box in the chart `ch`. Set its name to `A`.

```
atomicBox = Stateflow.AtomicBox(ch);  
atomicBox.Name = "A";
```

Version History

Introduced in R2012b

R2023a: New object functions and properties

Errors starting in R2023a

`Stateflow.AtomicBox` objects have new object functions and properties:

- The object function `getReferences` returns the locations where a chart refers to the name of an atomic box.
- The object function `renameReferences` renames an atomic box and updates all references to the atomic box name in the chart.
- The object function `commentedBy` identifies the explicitly commented objects that cause an atomic box to be commented out.
- The property `IsCommented` indicates whether an atomic box is commented out. This property replaces the object function `isCommented`.

R2022b: Map variables for atomic boxes

Edit the mapping of atomic box symbols by calling the object functions `getMappingForSymbol`, `setMappingForSymbol`, `clearMappingForSymbol`, and `disableMappingForSymbol`.

See Also

`Stateflow.Box` | `Stateflow.Chart` | `Stateflow.Function` | `Stateflow.State`

Topics

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

“Reuse Functions by Using Atomic Boxes”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.AtomicSubchart

Atomic subchart in chart, state, or box

Description

Use `Stateflow.AtomicSubchart` objects to create independent subcomponents in a Stateflow chart. Atomic subcharts allow for:

- Reuse of the same state or subchart across multiple charts and models
- Faster simulation after making small changes to a chart with many states or levels of hierarchy
- Ease of team development when multiple people are working on different parts of the same chart
- Manual inspection of generated code for a specific state or subchart in a chart

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

Creation

Syntax

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

Description

`atomicSubchart = Stateflow.AtomicSubchart(parent)` creates a `Stateflow.AtomicSubchart` object in a parent chart, state, or box.

Input Arguments

parent — Parent for new atomic subchart

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object

Parent for the new atomic subchart, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.State`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Name — Name of atomic subchart

"" (default) | string scalar | character vector

Name of the atomic subchart, specified as a string scalar or character vector.

LabelString — Label for atomic subchart

"?" (default) | string scalar | character vector

Label for the atomic subchart, specified as a string scalar or character vector.

IsLink — Whether atomic subchart is a library link

true or 1 | false or 0

This property is read-only.

Whether the atomic subchart is a library link, specified as a numeric or logical 1 (true) or 0 (false).

IsExplicitlyCommented — Whether to comment out atomic subchart

false or 0 (default) | true or 1

Whether to comment out the atomic subchart, specified as a numeric or logical 1 (true) or 0 (false). Setting this property to true is equivalent to right-clicking the atomic subchart and selecting **Comment Out**. For more information, see “Comment Out Objects in a Stateflow Chart”.

IsImplicitlyCommented — Whether atomic subchart is implicitly commented out

true or 1 | false or 0

This property is read-only.

Whether the atomic subchart is implicitly commented out, specified as a numeric or logical 1 (true) or 0 (false). The atomic subchart is implicitly commented out when you explicitly comment out an object that contains it. If the atomic subchart is contained in another atomic subchart, this property is false unless the explicitly commented object is also contained in the atomic subchart.

IsCommented — Whether atomic subchart is commented out


true or 1 | false or 0

This property is read-only.

Whether the atomic subchart is commented out, specified as a numeric or logical 1 (true) or 0 (false). This property is true when either IsExplicitlyCommented or IsImplicitlyCommented is true.

CommentText — Comment text

"" (default) | string scalar | character vector

Comment text for the atomic subchart, specified as a string scalar or character vector. This property applies only when the IsExplicitlyCommented property is true. In the Stateflow Editor, when you point to the comment badge  on the atomic subchart, the text appears as a tooltip. When you set the IsExplicitlyCommented property to false, the value of CommentText reverts to "".

Graphical Appearance

Position — Position and size of atomic subchart

[0 0 90 60] (default) | [left top width height]

Position and size of the atomic subchart, specified as a four-element numeric vector of the form [left top width height].

BadIntersection — Whether atomic subchart intersects a box, state, or function

true or 1 | false or 0

This property is read-only.

Whether the atomic subchart graphically intersects a box, state, or function, specified as a numeric or logical 1 (true) or 0 (false).

ContentPreviewEnabled — Whether to display preview of atomic subchart contents

false or 0 (default) | true or 1

Whether to display a preview of the atomic subchart contents, specified as a numeric or logical 1 (true) or 0 (false).

ArrowSize — Size of incoming transition arrows

8 (default) | scalar

Size of incoming transition arrows, specified as a scalar.

FontSize — Font size for atomic subchart label

scalar

Font size for the atomic subchart label, specified as a scalar. The `StateFont.Size` property of the chart that contains the atomic subchart sets the initial value of this property.

State Decomposition

Type — Decomposition of sibling states

'AND' | 'OR'

This property is read-only.

Decomposition of sibling states, specified as 'AND' or 'OR'. The atomic subchart inherits this property from the `Decomposition` property of its parent state or chart.

ExecutionOrder — Execution order in parallel (AND) decomposition

scalar

Execution order for the atomic subchart in parallel (AND) decomposition, specified as an integer scalar. This property applies only when both of these conditions are satisfied:

- The `Type` property of the atomic subchart is "AND".
- The `UserSpecifiedStateTransitionExecutionOrder` property of the chart that contains the atomic subchart is true.

Active State Output

HasOutputData — Whether to create active state data output

false or 0 (default) | true or 1

Whether to create an active state data output port for the atomic subchart, specified as a numeric or logical 1 (true) or 0 (false). For more information, see “Monitor State Activity Through Active State Data”.

OutputData — Active state data object

Stateflow.Data object

This property is read-only.

Active state data object for the atomic subchart, specified as a Stateflow.Data object. This property applies only when the HasOutputData property for the atomic subchart is true.

OutputPortName — Name of active state data object

string scalar | character vector

Name of the active state data object for the atomic subchart, specified as a string scalar or character vector. This property applies only when the HasOutputData property for the atomic subchart is true.

OutputMonitoringMode — Monitoring mode for active state output

"SelfActivity"

Monitoring mode for the active state output data, specified as a string scalar or character vector. For atomic subcharts, the only option is "SelfActivity".

Signal Logging and Test Point Monitoring

LoggingInfo — Signal logging properties

Stateflow.SigLoggingInfo object

Signal logging properties for the atomic subchart, specified as a Stateflow.SigLoggingInfo object with these properties:

- **DataLogging** — Whether to enable signal logging, specified as a numeric or logical 1 (true) or 0 (false).
- **DecimateData** — Whether to limit the amount of logged data, specified as a numeric or logical 1 (true) or 0 (false).
- **Decimation** — Decimation interval, specified as an integer scalar. This property applies only when the DecimateData property is true.
- **LimitDataPoints** — Whether to limit the number of data points to log, specified as a numeric or logical 1 (true) or 0 (false).
- **MaxPoints** — Maximum number of data points to log, specified as an integer scalar. This property applies only when the LimitDataPoints property is true.
- **NameMode** — Source of the signal name, specified as "SignalName" or "Custom".
- **LoggingName** — Custom signal name, specified as a string scalar or character vector. This property applies only when the NameMode property is "Custom".

Signal logging saves the self activity of the atomic subchart to the MATLAB workspace during simulation. For more information, see “Log Simulation Output for States and Data”.

Example: `state.LoggingInfo.DataLogging = true;`

TestPoint — Whether to set atomic subchart as test point

false or 0 (default) | true or 1

Whether to set the atomic subchart as a test point, specified as a numeric or logical 1 (`true`) or 0 (`false`). You can monitor testpoints with a floating scope during simulation. You can also log test point values to the MATLAB workspace. For more information, see “Monitor Test Points in Stateflow Charts”.

Debugging

Debug — Debugger properties

`Stateflow.StateDebug` object

Debugger properties for the state, atomic subchart, or Simulink based state, specified as a `Stateflow.StateDebug` object with these properties:

- **OnEntry** — Whether to set the `On State Entry` breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).
- **OnDuring** — Whether to set the `During State` breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).
- **OnExit** — Whether to set the `On State Exit` breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).

For more information, see “Set Breakpoints to Debug Charts”.

Example: `atomicSubchart.Debug.Breakpoints.OnEntry = true;`

Example: `atomicSubchart.Debug.Breakpoints.OnDuring = true;`

Example: `atomicSubchart.Debug.Breakpoints.OnExit = true;`

Hierarchy

Chart — Chart that contains atomic subchart

`Stateflow.Chart` object

This property is read-only.

Chart that contains the atomic subchart, specified as a `Stateflow.Chart` object.

Subchart — Contents of atomic subchart

`Stateflow.Chart` object

This property is read-only.

Contents of the atomic subchart, specified as a `Stateflow.Chart` object. Use this object to add children, such as states and transitions, to the atomic subchart. For more information, see “Add Exit Port and Junction to Atomic Subchart” on page 2-20.

Subviewer — Subviewer for atomic subchart

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object

This property is read-only.

Subviewer for the atomic subchart, specified as a `Stateflow.Chart`, `Stateflow.State`, or `Stateflow.Box` object. The subviewer is the chart or subchart where you can graphically view the atomic subchart.

Machine — Machine that contains atomic subchart

`Stateflow.Machine` object

This property is read-only.

Machine that contains the atomic subchart, specified as a `Stateflow.Machine` object.

Path — Location of parent in model hierarchy

character vector

This property is read-only.

Location of the parent of the atomic subchart in the model hierarchy, specified as a character vector.

Identification

Description — Description

"" (default) | string scalar | character vector

Description for the atomic subchart, specified as a string scalar or character vector.

Document — Document link

"" (default) | string scalar | character vector

Document link for the atomic subchart, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the atomic subchart, specified as data of any type.

SSIdNumber — Session-independent identifier

scalar

This property is read-only.

Session-independent identifier, specified as an integer scalar. Use this property to distinguish the atomic subchart from other objects in the model.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Unlike `SSIdNumber`, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>getParent</code>	Identify parent of object
<code>getReferences</code>	Identify references to symbol name
<code>renameReferences</code>	Rename symbol and update references to symbol name
<code>commentedBy</code>	Identify objects that implicitly comment out a graphical object
<code>getMappingForSymbol</code>	Get mapping for symbol in atomic subchart, atomic box, or Simulink based state
<code>setMappingForSymbol</code>	Set mapping for symbol in atomic subchart, atomic box, or Simulink based state
<code>clearMappingForSymbol</code>	Clear mapping for symbol in atomic subchart, atomic box, or Simulink based state
<code>disableMappingForSymbol</code>	Disable input event in atomic subchart or box
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>highlight</code>	Highlight graphical object
<code>fitToView</code>	Zoom in on graphical object

Examples

Add Atomic Subchart to Chart

Add an atomic subchart in the chart `ch`. Set its name to `A`.

```
atomicSubchart = Stateflow.AtomicSubchart(ch);  
atomicSubchart.Name = "A";
```

Add Exit Port and Junction to Atomic Subchart

In an atomic subchart called `A`, add an exit port and an exit junction with the label `"exit"`.

Find the `Stateflow.AtomicSubchart` object that corresponds to the atomic subchart `A` in the chart `ch`.

```
atomicSubchart = find(ch, "-isa", "Stateflow.AtomicSubchart", Name="A");
```

Add an exit junction to the atomic subchart. Use the `Subchart` property of the atomic subchart as the parent of the exit junction. Display the value of the `PortType` property of the exit junction.

```
exitJunction = Stateflow.Port(atomicSubchart.Subchart, "ExitJunction");  
exitJunction.PortType
```

```
ans =
```

```
    'ExitJunction'
```

Set the label of the exit junction to `"exit"`.

```
exitJunction.labelString = "exit";
```

Find the `Stateflow.Port` object for the matching exit port. Display the value of the `PortType` property of the exit port.

```
exitPort = Stateflow.findMatchingPort(exitJunction);
exitPort.PortType
```

```
ans =
    'ExitPort'
```

Display the label of the exit port.

```
exitPort.labelString
```

```
ans =
    'exit'
```

Map Variables in Atomic Subchart

In an atomic subchart called A, modify the mapping for the subchart input u1.

Open the model `sf_atomic_iodata_fixed.slx`.

```
open_system("sf_atomic_iodata_fixed")
```

Access the `Stateflow.AtomicSubchart` object for the atomic subchart A.

```
subsystem = find(sfroot, "-isa", "Stateflow.AtomicSubchart", ...
    Name="A");
```

Use the `Subchart` property to access the `Stateflow.Data` object for subchart input u1.

```
subsystemSymbol = find(subsystem.Subchart, ...
    "-isa", "Stateflow.Data", Name="u1");
```

Use the `Chart` property to access the `Stateflow.Data` object for chart input u2.

```
chartSymbol = find(subsystem.Chart, ...
    "-isa", "Stateflow.Data", Name="u2");
```

Check the mapping for subchart input u1.

```
getMappingForSymbol(subsystem, subsystemSymbol).Name

ans =
    'u1'
```

Map subchart input u1 to chart input u2.

```
setMappingForSymbol(subsystem, subsystemSymbol, chartSymbol)
getMappingForSymbol(subsystem, subsystemSymbol).Name

ans =
    'u2'
```

Clear the mapping for subchart input u1.

```
clearMappingForSymbol(subsystem, subsystemSymbol)
getMappingForSymbol(subsystem, subsystemSymbol).Name
```

```
ans =  
'u1'
```

Version History

Introduced in R2010b

R2023a: New object functions and properties

Errors starting in R2023a

Stateflow.AtomicSubchart objects have new object functions and properties:

- The object function `getReferences` returns the locations where a chart refers to the name of an atomic subchart.
- The object function `renameReferences` renames an atomic subchart and updates all references to the atomic subchart name in the chart.
- The object function `commentedBy` identifies the explicitly commented objects that cause an atomic subchart to be commented out.
- The property `IsCommented` indicates whether an atomic subchart is commented out. This property replaces the object function `isCommented`.

R2022b: Map variables for atomic subcharts

Edit the mapping of atomic subchart symbols by calling the object functions `getMappingForSymbol`, `setMappingForSymbol`, `clearMappingForSymbol`, and `disableMappingForSymbol`.

See Also

Functions

`find` | `Stateflow.findMatchingPort`

Objects

`Stateflow.Box` | `Stateflow.Chart` | `Stateflow.Port` | `Stateflow.State`

Topics

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

“Create Reusable Subcomponents by Using Atomic Subcharts”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.Box

Box in chart, state, box, or function

Description

Use `Stateflow.Box` objects to organize objects such as functions and states in your chart. You can also use a box to encapsulate states and functions in a separate namespace. For more information, see “Group Chart Objects by Using Boxes”.

Creation

Syntax

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

Description

`box = Stateflow.Box(parent)` creates a `Stateflow.Box` object in a parent chart, state, box, or graphical function.

Input Arguments

parent — Parent for new box

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object

Parent for the new box, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.Function`
- `Stateflow.State`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Name — Name of box

"" (default) | string scalar | character vector

Name of the box, specified as a string scalar or character vector.

LabelString — Label for box`"?"` (default) | string scalar | character vector

Label for the box, specified as a string scalar or character vector.

IsExplicitlyCommented — Whether to comment out box`false` or 0 (default) | `true` or 1

Whether to comment out the box, specified as a numeric or logical 1 (`true`) or 0 (`false`). Setting this property to `true` is equivalent to right-clicking the box and selecting **Comment Out**. For more information, see “Comment Out Objects in a Stateflow Chart”.

IsImplicitlyCommented — Whether box is implicitly commented out`true` or 1 | `false` or 0

This property is read-only.


Whether the box is implicitly commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). The box is implicitly commented out when you explicitly comment out an object that contains it. If the box is contained in an atomic subchart or an atomic box, this property is `false` unless the explicitly commented object is also contained in the atomic subchart or atomic box.

IsCommented — Whether box is commented out`true` or 1 | `false` or 0

This property is read-only.

Whether the box is commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property is `true` when either `IsExplicitlyCommented` or `IsImplicitlyCommented` is `true`.

CommentText — Comment text`""` (default) | string scalar | character vector

Comment text for the box, specified as a string scalar or character vector. This property applies only when the `IsExplicitlyCommented` property is `true`. In the Stateflow Editor, when you point to the comment badge  on the box, the text appears as a tooltip. When you set the `IsExplicitlyCommented` property to `false`, the value of `CommentText` reverts to `""`.

Graphical Appearance**Position — Position and size of box**`[0 0 90 60]` (default) | [`left top width height`]

Position and size of the box, specified as a four-element numeric vector of the form [`left top width height`].

BadIntersection — Whether box intersects a box, state, or function`true` or 1 | `false` or 0

This property is read-only.

Whether the box graphically intersects a box, state, or function, specified as a numeric or logical 1 (`true`) or 0 (`false`).

IsGrouped — Whether box is a grouped box

false or 0 (default) | true or 1

Whether the box is a grouped box, specified as a numeric or logical 1 (true) or 0 (false). When you copy and paste a grouped box, you copy not only the box but all of its contents. For more information, see “Copy and Paste by Grouping” on page 2-28.

IsSubchart — Whether box is a subchart

false or 0 (default) | true or 1

Whether the box is a subchart, specified as a numeric or logical 1 (true) or 0 (false).

ContentPreviewEnabled — Whether to display preview of box contents

false or 0 (default) | true or 1

Whether to display a preview of the box contents, specified as a numeric or logical 1 (true) or 0 (false). This property applies only when the IsSubchart property is true.

FontSize — Font size for box label

scalar

Font size for the box label, specified as a scalar. The StateFont.Size property of the chart that contains the box sets the initial value of this property.

State Decomposition**ExecutionOrder — Execution order in parallel (AND) decomposition**

scalar

Execution order for the substates of the box in parallel (AND) decomposition, specified as an integer scalar. This property applies only when both of these conditions are satisfied:

- The Decomposition property of the parent state or chart is "PARALLEL_AND".
- The UserSpecifiedStateTransitionExecutionOrder property of the chart that contains the box is true.

Hierarchy**Chart — Chart that contains box**

Stateflow.Chart object

This property is read-only.

Chart that contains the box, specified as a Stateflow.Chart object.

Subviewer — Subviewer for box

Stateflow.Chart object | Stateflow.State object | Stateflow.Box object | Stateflow.Function object

This property is read-only.

Subviewer for the box, specified as a Stateflow.Chart, Stateflow.State, Stateflow.Box, or Stateflow.Function object. The subviewer is the chart or subchart where you can graphically view the box.

Machine — Machine that contains box`Stateflow.Machine` object

This property is read-only.

Machine that contains the box, specified as a `Stateflow.Machine` object.

Path — Location of parent in model hierarchy

character vector

This property is read-only.

Location of the parent of the box in the model hierarchy, specified as a character vector.

Identification**Description — Description**

"" (default) | string scalar | character vector

Description for the box, specified as a string scalar or character vector.

Document — Document link

"" (default) | string scalar | character vector

Document link for the box, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the box, specified as data of any type.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Use this property to distinguish the box from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>find</code>	Identify specified objects in hierarchy
<code>getChildren</code>	Identify children of object
<code>getParent</code>	Identify parent of object
<code>getReferences</code>	Identify references to symbol name
<code>renameReferences</code>	Rename symbol and update references to symbol name
<code>commentedBy</code>	Identify objects that implicitly comment out a graphical object
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>highlight</code>	Highlight graphical object
<code>fitToView</code>	Zoom in on graphical object

Examples

Add Box to Chart

Add a box in the chart `ch`. Set its name to `A`.

```
box = Stateflow.Box(ch);  
box.Name = "A";
```

Version History

Introduced before R2006a

R2023a: New object functions and properties

Errors starting in R2023a

`Stateflow.Box` objects have new object functions and properties:

- The object function `getReferences` returns the locations where a chart refers to the name of a box.
- The object function `renameReferences` renames a box and updates all references to the box name in the chart.
- The object function `commentedBy` identifies the explicitly commented objects that cause a box to be commented out.
- The property `IsCommented` indicates whether a box is commented out. This property replaces the object function `isCommented`.

R2022b: ExecutionOrder property

The `ExecutionOrder` property specifies the order of execution for the substates of the box in parallel (AND) decomposition.

See Also

`Stateflow.Chart` | `Stateflow.Function` | `Stateflow.State`

Topics

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

“Group Chart Objects by Using Boxes”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.Clipboard

Clipboard to copy and paste Stateflow objects

Description

Use the `Stateflow.Clipboard` object to copy and paste graphical and nongraphical objects within the same chart, between charts in the same Simulink model, or between charts in different models.

Creation

There is only one `Stateflow.Clipboard` object, which is created automatically when you start Stateflow. To access this object, call the `sfclipboard` function:

```
clipboard = sfclipboard;
```

Object Functions

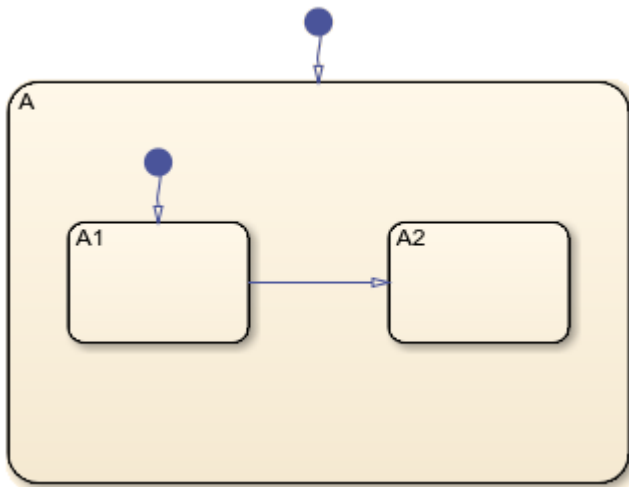
`copy` Copy array of objects to clipboard

`pasteTo` Paste objects in clipboard to specified container object

Examples

Copy and Paste by Grouping

Group a state and copy its contents to the chart. When you group a state, box, or graphical function, you can copy and paste all the objects contained in the grouped object, as well as all the relationships among these objects. This method is the simplest way of copying and pasting objects programmatically. If a state is not grouped, copying the state does not copy any of its contents.



Open the model and access the `Stateflow.Chart` object for the chart.

```
open_system("sfHierarchyAPIExample")
ch = find(sfroot, "-isa", "Stateflow.Chart");
```

Find the Stateflow.State object named A.

```
sA = find(ch, "-isa", "Stateflow.State", Name="A");
```

Group state A and its contents by setting the IsGrouped property for sA to true. Save the previous setting of this property so you can revert to it later.

```
prevGrouping = sA.IsGrouped;
sA.IsGrouped = true;
```

Change the name of the state to Copy_of_A. Save the previous name so you can revert to it later.

```
prevName = sA.Name;
newName = "Copy_of_" + prevName;
sA.Name = newName;
```

Access the clipboard object.

```
cb = sfclipboard;
```

Copy the grouped state to the clipboard.

```
copy(cb, sA);
```

Restore the state properties to their original settings.

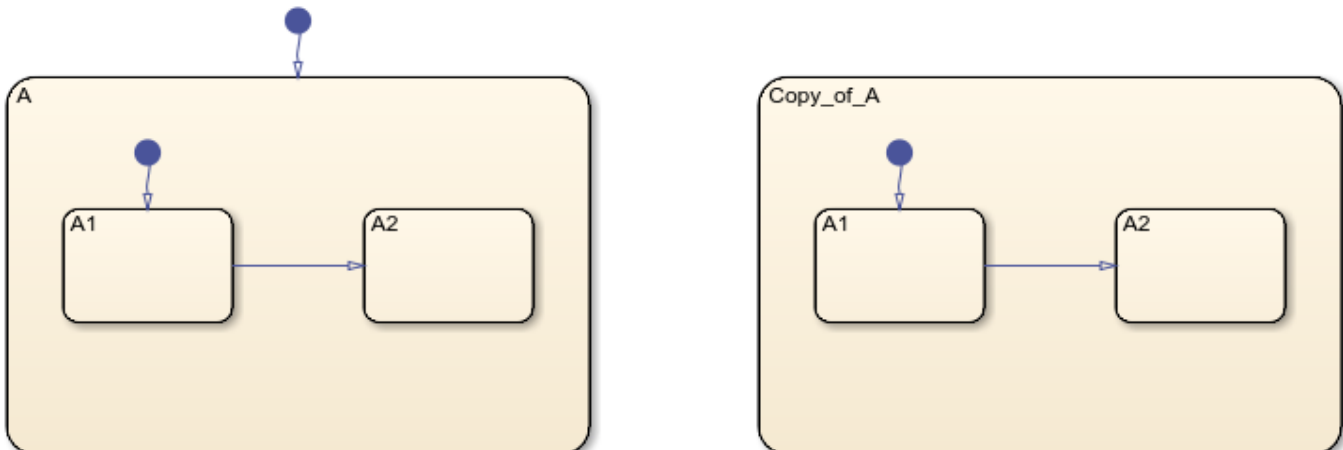
```
sA.IsGrouped = prevGrouping;
sA.Name = prevName;
```

Paste a copy of the objects from the clipboard to the chart.

```
pasteTo(cb, ch);
```

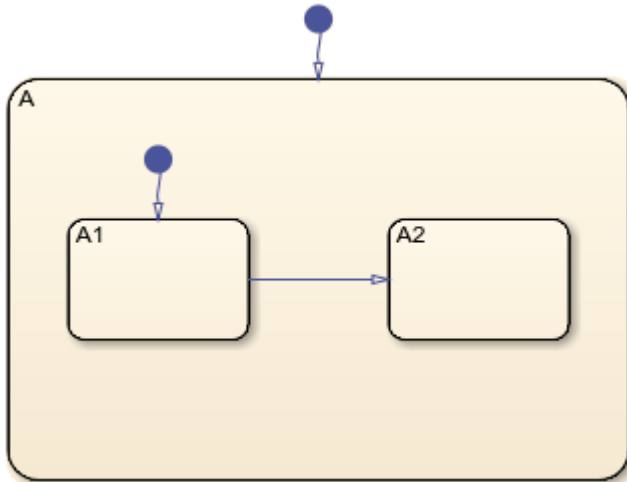
Adjust the state properties of the new state.

```
sNew = find(ch, "-isa", "Stateflow.State", Name=newName);
sNew.Position = sA.Position + [400 0 0 0];
sNew.IsGrouped = prevGrouping;
```



Copy and Paste Array of Objects

Copy states A1 and A2, along with the transition between them, to a new state in the chart. To preserve transition connections and containment relationships between objects, copy all the connected objects at once.



Open the model and access the `Stateflow.Chart` object for the chart.

```
open_system("sfHierarchyAPIExample")
ch = find(sfroot, "-isa", "Stateflow.Chart");
```

Find the `Stateflow.State` object named A.

```
sA = find(ch, "-isa", "Stateflow.State", Name="A");
```

Add a new state called B. To enable pasting of other objects inside B, convert the new state to a subchart.

```
sB = Stateflow.State(ch);
sB.Name = "B";
sB.Position = sA.Position + [400 0 0 0];
sB.IsSubchart = true;
```

Create an array called `objArray` that contains the states and transitions in state A. Use the function `setdiff` to remove state A from the array of objects to copy.

```
objArrayS = find(sA, "-isa", "Stateflow.State");
objArrayS = setdiff(objArrayS, sA);
objArrayT = find(sA, "-isa", "Stateflow.Transition");
objArray = [objArrayS objArrayT];
```

Access the clipboard object.

```
cb = sfclipboard;
```

Copy the objects in `objArray` and paste them in subchart B.

```
copy(cb,objArray);
pasteTo(cb,sB);
```

Revert B to a state.

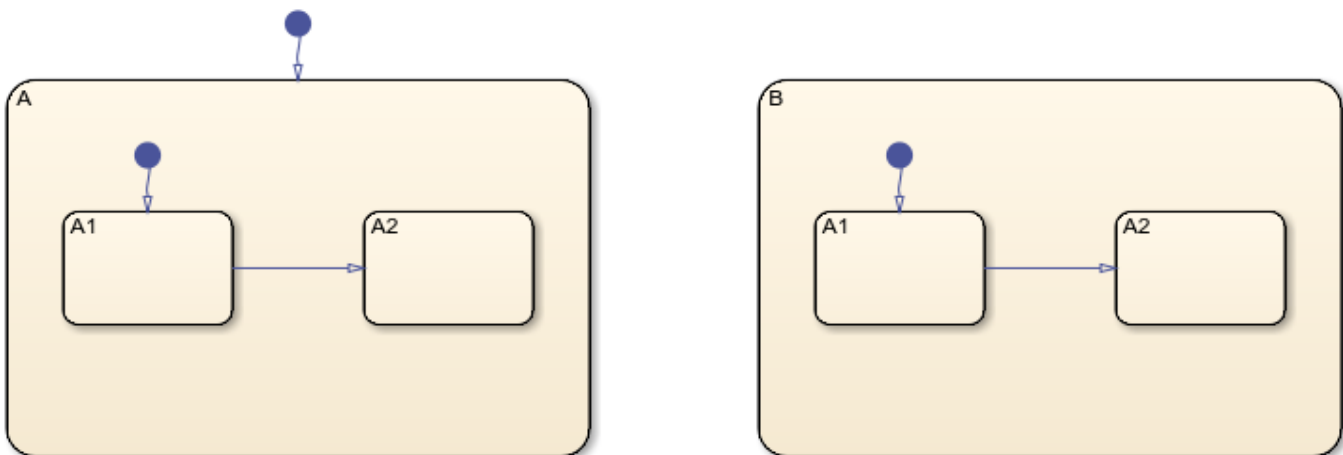
```
sB.IsSubchart = false;
sB.IsGrouped = false;
```

Reposition the states and transitions in B.

```
newStates = find(sB,"-isa","Stateflow.State");
newStates = setdiff(newStates,sB);

newTransitions = find(sB,"-isa","Stateflow.Transition");
newOClocks = get(newTransitions,{"SourceOClock","DestinationOClock"});

for i = 1:numel(newStates)
    newStates(i).Position = newStates(i).Position + [25 35 0 0];
end
set(newTransitions,{"SourceOClock","DestinationOClock"},newOClocks);
```



Version History

Introduced before R2006a

See Also

Functions

find | setdiff | sfclipboard

Objects

Stateflow.State

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.Chart

Graphical representation of a finite state machine

Description

Use a `Stateflow.Chart` object to create a graphical representation of a finite state machine based on a state transition diagram. In a Chart block, states and transitions form the basic building blocks of a sequential logic system. States correspond to operating modes and transitions represent pathways between states. For more information, see “Model Finite State Machines by Using Stateflow Charts” and “Create Charts by Using the Stateflow API” on page 1-19.

Creation

To create a `Stateflow.Chart` object, call the function `sfnew`. For example, to create an empty chart in a new Simulink model called `myModel`, enter:

```
sfnew myModel
```

Alternatively, you can add a new chart to an existing model by using the function `add_block`:

```
add_block("sflib/Chart", "myModel/Chart")
```

Then, to access the `Stateflow.Chart` object, call the `find` function for the `Simulink.Root` object:

```
chart = find(sfroot, "-isa", "Stateflow.Chart", ...  
            Path="myModel/Chart");
```

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Name — Name of chart

"Chart" (default) | string scalar | character vector

Name of the chart, specified as a string scalar or character vector.

ActionLanguage — Action language

"MATLAB" (default) | "C"

Action language used to program the chart, specified as "MATLAB" or "C". For more information, see “Differences Between MATLAB and C as Action Language Syntax”.

StateMachineType — State machine semantics

"Classic" (default) | "Mealy" | "Moore"

State machine semantics implemented by the chart, specified as "Classic", "Mealy", or "Moore". For more information, see "Overview of Mealy and Moore Machines".

SupportVariableSizing — Whether chart supports variable-size data

true or 1 (default) | false or 0

Whether the chart supports variable-size data, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Declare Variable-Size Data in Stateflow Charts".

TreatDimensionOfLengthOneAsFixedSize — Whether chart output data with at least one dimension of length 1 are fixed size

true or 1 (default) | false or 0

Whether chart output data with at least one dimension of length 1 are fixed size, specified as a numeric or logical 0 (false) or 1 (true). When this property is true, the object sets data that are variable size in the chart with a dimension of 1 to fixed size. When this property is false, data in the chart that has the **Variable size** property enabled are always variable size. Prior to R2023a, the object treats data with at least one dimension of length 1 as fixed size.

This property only affects output data that have the **Variable size** property enabled. See "Variable size" (Simulink).

State Decomposition

Decomposition — Decomposition of substates

"EXCLUSIVE_OR" (default) | "PARALLEL_AND"

Decomposition of substates at the top level of containment in the chart, specified as "EXCLUSIVE_OR" or "PARALLEL_AND". For more information, see "Define Exclusive and Parallel Modes by Using State Decomposition".

Chart Initialization

ExecuteAtInitialization — Whether to initialize state configuration

false or 0 (default) | true or 1

Whether to initialize the state configuration of the chart at time zero instead of at the first input event, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Execution of a Chart at Initialization".

StatesWhenEnabling — Behavior of states when event reenables chart

"" (default) | "held" | "reset"

Behavior of the states when a function-call input event reenables the chart, specified as one of these values:

- "" — The chart does not contain function-call input events.
- "held" — The chart maintains the most recent values of the states.
- "reset" — The chart reverts to the initial conditions of the states.

For more information, see "Control States in Charts Enabled by Function-Call Input Events".

InitializeOutput — Whether to initialize output data

false or 0 (default) | true or 1

Whether to initialize the output data every time the chart wakes up, specified as a numeric or logical 1 (`true`) or 0 (`false`). For more information, see “Initialize outputs every time chart wakes up”.

Active State Output

HasOutputData — Whether to create active state data output

`false` or 0 (default) | `true` or 1

Whether to create an active state data output port for the chart, specified as a numeric or logical 1 (`true`) or 0 (`false`). For more information, see “Monitor State Activity Through Active State Data”.

OutputData — Active state data object

`Stateflow.Data` object

This property is read-only.

Active state data object for the chart, specified as a `Stateflow.Data` object. This property applies only when the `HasOutputData` property for the chart is `true`.

OutputPortName — Name of active state data object

string scalar | character vector

Name of the active state data object for the chart, specified as a string scalar or character vector. This property applies only when the `HasOutputData` property for the chart is `true`.

OutputMonitoringMode — Monitoring mode for active state output

"ChildActivity" (default) | "LeafStateActivity"

Monitoring mode for the active state output data, specified as "ChildActivity" or "LeafStateActivity".

EnumTypeName — Name of enumerated data type for active state data object

string scalar | character vector

Name of the enumerated data type for the active state data object for the chart, specified as a string scalar or character vector. For more information, see “Enum Name”.

DoNotAutogenerateEnum — Whether to define enumerated data type manually

`false` or 0 (default) | `true` or 1

Whether to define the enumerated data type for the active state data output manually, specified as a numeric or logical 1 (`true`) or 0 (`false`). For more information, see “Define State Activity Enumeration Type”.

Discrete and Continuous-Time Semantics

ChartUpdate — Activation method for chart

"INHERITED" (default) | "CONTINUOUS" | "DISCRETE"

Activation method for the chart, specified as "CONTINUOUS", "DISCRETE", or "INHERITED". For more information, see “Update Method”.

SampleTime — Sample time for activating chart

"-1" (default) | string scalar | character vector

Sample time for activating the chart, specified as a string scalar or character vector. This property applies only when the `ChartUpdate` property for the chart is "DISCRETE".

EnableZeroCrossings — Whether to enable zero-crossing detection

`true` or 1 (default) | `false` or 0

Whether to enable zero-crossing detection on state transitions in the chart, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property applies only when the `ChartUpdate` property for the chart is set to "CONTINUOUS". For more information, see "Disable Zero-Crossing Detection".

Super Step Semantics

EnableNonTerminalStates — Whether to enable super step semantics

`false` or 0 (default) | `true` or 1

Whether to enable super step semantics for the chart, specified as a numeric or logical 1 (`true`) or 0 (`false`). For more information, see "Super Step Semantics".

NonTerminalMaxCounts — Maximum number of transitions in one super step

1000 (default) | scalar

Maximum number of transitions the chart can take in one super step, specified as an integer scalar. This property applies only when the `EnableNonTerminalStates` property for the chart is `true`.

NonTerminalUnstableBehavior — Behavior if super step exceeds maximum number of transitions

"Proceed" (default) | "Throw Error"

Behavior if a super step for the chart exceeds the maximum number of transitions specified in the `NonTerminalMaxCounts` property before reaching a stable state, specified as one of these values:

- "Proceed" — The chart goes to sleep with the last active state configuration.
- "Throw Error" — The chart generates an error.

This property applies only when the `EnableNonTerminalStates` property for the chart is `true`.

Exported Functions

ExportChartFunctions — Whether to export chart-level functions

`false` or 0 (default) | `true` or 1

Whether to export chart-level functions to other blocks in the Simulink model, specified as a numeric or logical 1 (`true`) or 0 (`false`). For more information, see "Export Stateflow Functions for Reuse".

AllowGlobalAccessToExportedFunctions — Whether exported functions are globally visible

`false` or 0 (default) | `true` or 1

Whether exported functions from the chart are globally visible in the Simulink model, specified as a numeric or logical 1 (`true`) or 0 (`false`). When this property is enabled, blocks throughout the model can call functions exported from the chart without using qualified notation. This property applies only when the `ExportChartFunctions` property for the chart is `true`.

Integer and Fixed-Point Data

SaturateOnIntegerOverflow — Whether data saturates on integer overflow

true or 1 (default) | false or 0

Whether the data in the chart saturates on integer overflow, specified as a numeric or logical 1 (true) or 0 (false). When this property is disabled, the data in the chart wraps on integer overflow. For more information, see “Handle Integer Overflow for Chart Data”.

TreatAsFi — Inherited Simulink signals to treat as fi objects

"Fixed-point" (default) | "Fixed-point & Integer"

Inherited Simulink signals to treat as Fixed-Point Designer fi objects, specified as one of these values:

- "Fixed-point" — The chart treats all fixed-point inputs as fi objects.
- "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.

EmlDefaultFimath — Default fimath properties

"Same as MATLAB Default" (default) | "Other:UserSpecified"

Default fimath properties for the chart, specified as one of these values:

- "Same as MATLAB Default" — Use the same fimath properties as the current default fimath object.
- "Other:UserSpecified" — Use the InputFimath property to specify the default fimath object.

This property applies only when the ActionLanguage property of the chart is "MATLAB".

InputFimath — Default fimath object

string scalar | character vector

Default fimath object, specified as a string scalar or character vector. When the EmlDefaultFimath property for the chart is "Other:UserSpecified", you can use this property to:

- 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 to charts that use MATLAB as the action language.

Code Generation

GeneratePreprocessorConditionals — Whether generated code includes a preprocessor conditional

false or 0 (default) | true or 1

Whether the generated code includes a preprocessor conditional statement for the variant conditions in the chart, specified as a numeric or logical 1 (true) or 0 (false). This property applies only when generating code with Embedded Coder. For more information, see “Control Indicator Lamp Dimmer Using Variant Conditions”.

C Action Language

EnableBitOps — Whether to use bit operations

false or 0 (default) | true or 1

Whether to use bit operations in state and transition actions in the chart, specified as a numeric or logical 1 (true) or 0 (false). This property applies only to charts that use C as the action language. For more information, see “Enable C-bit operations”.

UserSpecifiedStateTransitionExecutionOrder — Whether to use explicit ordering of parallel states and transitions

true or 1 (default) | false or 0

Whether to use explicit ordering of parallel states and transitions, specified as a numeric or logical 1 (true) or 0 (false). This property applies only to charts that use C as the action language. For more information, see “User-specified state/transition execution order”.

Debugging

Debug — Debugger properties

Stateflow.ChartDebug object

Debugger properties for the chart, specified as a Stateflow.ChartDebug object with this property:

- **Breakpoints.OnEntry** — Whether to set the On Chart Entry breakpoint, specified as a numeric or logical 1 (true) or 0 (false).

For more information, see “Set Breakpoints to Debug Charts”.

Example: `chart.Debug.Breakpoints.OnEntry = true;`

Graphical Appearance

Editor — Editor

Stateflow.Editor object

This property is read-only.

Editor for the chart, specified as a Stateflow.Editor object. You can use this object to control the position, size, and magnification level of the Stateflow Editor window.

Visible — Whether editor is displaying chart

true or 1 | false or 0

Whether the Stateflow Editor window is displaying the chart, specified as a numeric or logical 1 (true) or 0 (false).

ChartColor — Background color

[1 0.9608 0.8824] (default) | [red green blue]

Background color for the chart, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.

StateColor — Color for states

[0 0 0] (default) | [red green blue]

Color for the boxes, functions, and states in the chart, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.

TransitionColor — Color for transitions

[0.2902 0.3294 0.6039] (default) | [red green blue]

Color for transitions in the chart, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.

JunctionColor — Color for junctions

[0.6824 0.3294 0] (default) | [red green blue]

Color for junctions in the chart, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.

StateFont — Font for state labels

Stateflow.StateFont object

Font for the box, function, and state labels in the chart, specified as a Stateflow.StateFont object with these properties:

- **Name** — Font name, specified as a string scalar or character vector. This property also determines the font for annotations in the chart.
- **Angle** — Font angle, specified as "NORMAL" or "ITALIC".
- **Weight** — Font weight, specified as "NORMAL" or "BOLD".
- **Size** — Default font size for new boxes, functions, and states, specified as a scalar. This property also determines the default font size for new annotations in the chart.

Example: chart.StateFont.Name = "Arial";

Example: chart.StateFont.Angle = "ITALIC";

Example: chart.StateFont.Weight = "BOLD";

Example: chart.StateFont.Size = 8;

StateLabelColor — Color for state labels

[0 0 0] (default) | [red green blue]

Color for the box, function, and state labels in the chart, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.

TransitionFont — Font for transition labels

Stateflow.TransFont object

Font for the transition labels in the chart, specified as a Stateflow.TransFont object with these properties:

- **Name** — Font name, specified as a string scalar or character vector.
- **Angle** — Font angle, specified as "NORMAL" or "ITALIC".

- **Weight** — Font weight, specified as "NORMAL" or "BOLD".
- **Size** — Default font size for new transitions, specified as a scalar.

Example: `chart.TransitionFont.Name = "Arial";`

Example: `chart.TransitionFont.Angle = "ITALIC";`

Example: `chart.TransitionFont.Weight = "BOLD";`

Example: `chart.TransitionFont.Size = 8;`

TransitionLabelColor — Color for transition labels

`[0.2902 0.3294 0.6039]` (default) | `[red green blue]`

Color for the transition labels in the chart, specified as a three-element numeric vector of the form `[red green blue]` that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.

Hierarchy

Machine — Machine that contains chart

`Stateflow.Machine` object

This property is read-only.

Machine that contains the chart, specified as a `Stateflow.Machine` object.

Path — Location of chart in model hierarchy

character vector

This property is read-only.

Location of the chart in the model hierarchy, specified as a character vector.

Dirty — Whether chart has changed

`true` or `1` | `false` or `0`

Whether the chart has changed after being opened or saved, specified as a numeric or logical `1` (`true`) or `0` (`false`).

Locked — Whether chart is locked

`false` or `0` (default) | `true` or `1`

Whether the chart is locked, specified as a numeric or logical `1` (`true`) or `0` (`false`). Enable this property to prevent changes in the chart.

Iced — Whether chart is locked

`false` or `0` (default) | `true` or `1`

This property is read-only.

Whether the chart is locked, specified as a numeric or logical `1` (`true`) or `0` (`false`). This property is equivalent to the property `Locked`, but is used internally to prevent changes in the chart during simulation.

Identification

Description — Description

"" (default) | string scalar | character vector

Description for the chart, specified as a string scalar or character vector.

Document — Document link

"" (default) | string scalar | character vector

Document link for the chart, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the chart, specified as data of any type.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Use this property to distinguish the chart from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>find</code>	Identify specified objects in hierarchy
<code>getChildren</code>	Identify children of object
<code>defaultTransitions</code>	Identify default transitions in specified object
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>fitToView</code>	Zoom in on graphical object

Examples

Create Empty Stateflow Chart

Call the function `sfnew` to open a new Simulink model that contains an empty Stateflow chart.

```
sfnew
```

Access the `Simulink.Root` object by calling the `sfroot` function.

```
rt = sfroot;
```

Access the `Stateflow.Chart` object by calling the `find` function for the `Simulink.Root` object.

```
chart = find(rt, "-isa", "Stateflow.Chart");
```

Version History

Introduced before R2006a

R2023a: Set chart output data of any dimension as variable size

You can now set output data of any dimension to be variable size by setting the `TreatDimensionOfLengthOneAsFixedSize` property to `false`. Prior to R2023a, the object treats data with at least one dimension of length 1 as fixed size.

See Also**Blocks**

Chart

Functions

`sfnew` | `sfroot` | `add_block`

Topics

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

“Model Finite State Machines by Using Stateflow Charts”

“Specify Properties for Stateflow Charts”

“Create Charts by Using the Stateflow API” on page 1-19

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.Data

Data in chart, state, box, or function

Description

Use `Stateflow.Data` objects to store values that are visible at a specific level of the Stateflow hierarchy. For more information, see “Add Stateflow Data” and “Set Data Properties”.

Creation

Syntax

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

Description

`data = Stateflow.Data(parent)` creates a `Stateflow.Data` object in a parent chart, state, box, or function.

Input Arguments

parent — Parent for new data object

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object | ...

Parent for the new data object, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.EMFunction`
- `Stateflow.Function`
- `Stateflow.SLFunction`
- `Stateflow.State`
- `Stateflow.TruthTable`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Interface

Name — Name of data object

"data" (default) | string scalar | character vector

Name of the data object, specified as a string scalar or character vector.

Scope — Scope of data object

"Local" (default) | "Input" | "Output" | "Constant" | "Parameter" | "Data Store Memory" | "Temporary" | "Imported" | "Exported"

Scope of the data object, specified as one of these values:

- "Local"
- "Input"
- "Output"
- "Constant"
- "Parameter"
- "Data Store Memory"
- "Temporary"
- "Imported"
- "Exported"

For more information, see "Scope".

Port — Port index for data object

scalar

Port index for the data object, specified as an integer scalar. This property applies only to input and output data. For more information, see "Port".

UpdateMethod — Method for updating data object

"Discrete" (default) | "Continuous"

Method for updating data object, specified as "Discrete" or "Continuous". This property applies only when the `ChartUpdate` property of the chart that contains the data is "CONTINUOUS". For more information, see "Continuous-Time Modeling in Stateflow".

InitializeMethod — Method for initializing data object

"Expression" (default) | "Parameter" | "Not Needed"

Method for initializing the value of the data object, specified as a string scalar or character vector that depends on the scope of the data:

- For local and output data, use "Expression" or "Parameter".
- For constant data, use "Expression".
- For input data, parameters, and data store memory, use "Not Needed".

To specify the initial value of the data object, use the `Props.InitialValue` property.

This property is equivalent to the **Initial Value** drop-down list in the Model Explorer and the Data properties dialog box. For more information, see "Initial value".

SaveToWorkspace — Whether to save data object to workspace variable

false or 0 (default) | true or 1

Whether to save the value of the data object to a variable of the same name in the MATLAB base workspace at the end of the simulation, specified as a numeric or logical 1 (true) or 0 (false). This

property applies only to data in charts that use C as the action language. For more information, see “Save final value to base workspace”.

Tunable — Whether data object is tunable parameter

true or 1 (default) | false or 0

Whether the data object is a tunable parameter, specified as a numeric or logical 1 (true) or 0 (false). Only tunable parameters can be modified during simulation. This property applies only to parameter data.

Data Specification

DataType — Type of data object

"Inherit: From definition in chart" (default) | "double" | "single" | "int32" | "uint32" | "boolean" | ...

Type of the data object, specified as a string scalar or character vector that depends on the `Props.Type.Method` property of the data object:

- If the `Props.Type.Method` property of the data object is "Inherit", the value of this property is "Inherit: From definition in chart" for local data and "Inherit: Same as Simulink" for input, output, and parameter data.
- If the `Props.Type.Method` property of the data object is "Built-in", you can specify this property with one of these options:
 - "double"
 - "single"
 - "int8"
 - "int16"
 - "int32"
 - "int64"
 - "uint8"
 - "uint16"
 - "uint32"
 - "uint64"
 - "boolean"
 - "string"
 - "ml" (Supported only in charts that use C as the action language)
- Otherwise, the `Props.Type` properties of the data object determine the value of this property.

For more information, see the section Add Data on page 1-0 in “Create Charts by Using the Stateflow API” on page 1-19.

Props — Data specification properties

`Stateflow.DataProps` object

Data specification properties, specified as a `Stateflow.DataProps` object with these properties:

- **Type.Method** — Method for setting the type of the data object, specified as a string scalar or character vector.

- For local, input, output, or parameter data, use "Inherited", "Built-in", "Bus Object", "Enumerated", "Expression", or "Fixed point".
- For constant data, use "Built-in", "Expression", or "Fixed point".
- For data store memory data, use "Inherited".

This property is equivalent to the **Mode** field of the Data Type Assistant in the Model Explorer and the Data properties dialog box. For more information, see “Specify Type of Stateflow Data”.

- **Type.BusObject** — Name of the Simulink.Bus object that defines the data object, specified as a string scalar or character vector. This property applies only when the Type.Method property of the data object is "Bus Object". For more information, see “Access Bus Signals Through Stateflow Structures”.
- **Type.EnumType** — Name of the enumerated type that defines the data object, specified as a string scalar or character vector. This property applies only when the Type.Method property of the data object is "Enumerated". For more information, see “Reference Values by Name by Using Enumerated Data”.
- **Type.Expression** — Expression that evaluates to the data type of the data object, specified as a string scalar or character vector. This property applies only when the Type.Method property of the data object is "Expression". For more information, see “Specify Data Properties by Using MATLAB Expressions”.
- **Type.Signed** — Signedness, specified as a numeric or logical 1 (true) or 0 (false). This property applies only when the Type.Method property of the data object is "Fixed point". For more information, see “Fixed-Point Data in Stateflow Charts”.
- **Type.WordLength** — Word length, in bits, specified as a string scalar or character vector. This property applies only when the Type.Method property of the data object is "Fixed point". For more information, see “Fixed-Point Data in Stateflow Charts”.
- **Type.Fixpt.ScalingMode** — Method for scaling the fixed-point data object, specified as "Binary point", "Slope and bias", or "None". This property applies only when the Type.Method property of the data object is "Fixed point". For more information, see “Fixed-Point Data in Stateflow Charts”.
- **Type.Fixpt.FractionLength** — Fraction length, in bits, specified as a string scalar or character vector. This property applies only when the Type.Method property is "Fixed point" and the Type.Fixpt.ScalingMode property is "Binary point".
- **Type.Fixpt.Slope** — Slope, specified as a string scalar or character vector. This property applies only when the Type.Method property is "Fixed point" and the Type.Fixpt.ScalingMode property is "Slope and bias".
- **Type.Fixpt.Bias** — Bias, specified as a string scalar or character vector. This property applies only when the Type.Method property is "Fixed point" and the Type.Fixpt.ScalingMode property is "Slope and bias".
- **Type.Fixpt.Lock** — Whether to prevent replacement of the fixed-point type with an autoscaled type chosen by the Fixed-Point Tool (Fixed-Point Designer), specified as a numeric or logical 1 (true) or 0 (false). This property applies only when the Type.Method property of the data object is "Fixed point".
- **Array.Size** — Size of the data object, specified as a string scalar or character vector. For more information, see “Specify Size of Stateflow Data”.
- **Array.IsDynamic** — Whether the data object has variable size, specified as a numeric or logical 1 (true) or 0 (false). This property is equivalent to the **Variable Size** check box in the **Property Inspector**, the Model Explorer, or the Data properties dialog box. For more information, see “Declare Variable-Size Data in Stateflow Charts”.

- **Array.FirstIndex** — Index for the first element of the array data object, specified as a string scalar or character vector. This property applies only to array data in charts that use C as the action language. For more information, see “Save final value to base workspace”.
- **Complexity** — Whether the data object accepts complex values, specified as "On" or "Off". For more information, see “Complex Data in Stateflow Charts”.
- **InitialValue** — Initial value, specified as a string scalar or character vector. For more information, see “Initial value”.
- **Range.Minimum** — Minimum value, specified as a string scalar or character vector. For more information, see “Limit range”.
- **Range.Maximum** — Maximum value, specified as a string scalar or character vector. For more information, see “Limit range”.
- **ResolveToSignalObject** — Whether the data object resolves to a `Simulink.Signal` object that you define in the model or base workspace, specified as a numeric or logical 1 (true) or 0 (false). For more information, see “Resolve Data Properties from Simulink Signal Objects”.
- **Unit.Name** — Unit of measurement, specified as a string scalar or character vector. This property applies only to input and output data. For more information, see “Specify Units for Stateflow Data”.

CompiledSize — Data size as determined by compiler`""` (default) | character vector

This property is read-only.

Data size as determined by the compiler, specified as a character vector.

CompiledType — Data type as determined by compiler`"unknown"` (default) | character vector

This property is read-only.

Data type as determined by the compiler, specified as a character vector.

Active State Output**OutputState — State or chart monitored by data object**`[]` (default) | `Stateflow.AtomicSubchart` object | `Stateflow.Chart` | `Stateflow.SimulinkBasedState` object | `Stateflow.State` object | `Stateflow.StateTransitionTableChart` object

This property is read-only.

State or chart monitored by the data object, specified as an empty array or a `Stateflow.AtomicSubchart`, `Stateflow.Chart`, `Stateflow.SimulinkBasedState`, `Stateflow.State`, or `Stateflow.StateTransitionTableChart` object. For more information, see “Monitor State Activity Through Active State Data”.

Signal Logging and Test Point Monitoring**LoggingInfo — Signal logging properties**`Stateflow.SigLoggingInfo` object

Signal logging properties for the data object, specified as a `Stateflow.SigLoggingInfo` object with these properties:

- **DataLogging** — Whether to enable signal logging, specified as a numeric or logical 1 (`true`) or 0 (`false`).
- **DecimateData** — Whether to limit the amount of logged data, specified as a numeric or logical 1 (`true`) or 0 (`false`).
- **Decimation** — Decimation interval, specified as an integer scalar. This property applies only when the `DecimateData` property is `true`.
- **LimitDataPoints** — Whether to limit the number of data points to log, specified as a numeric or logical 1 (`true`) or 0 (`false`).
- **MaxPoints** — Maximum number of data points to log, specified as an integer scalar. This property applies only when the `LimitDataPoints` property is `true`.
- **NameMode** — Source of the signal name, specified as "SignalName" or "Custom".
- **LoggingName** — Custom signal name, specified as a string scalar or character vector. This property applies only when the `NameMode` property is "Custom".

Signal logging saves the values of the data object to the MATLAB workspace during simulation. For more information, see “Log Simulation Output for States and Data”.

Example: `data.LoggingInfo.DataLogging = true;`

TestPoint — Whether to set data object as test point

`false` or 0 (default) | `true` or 1

Whether to set the data object as a test point, specified as a numeric or logical 1 (`true`) or 0 (`false`). You can monitor testpoints with a floating scope during simulation. You can also log test point values to the MATLAB workspace. For more information, see “Monitor Test Points in Stateflow Charts”.

Debugging

Debug — Debugger properties

`Stateflow.DataDebug` object

Debugger properties for the data object, specified as a `Stateflow.DataDebug` object with this property:

- **Watch** — Whether to track the value of the data object in the Breakpoints and Watch window, specified as a numeric or logical 1 (`true`) or 0 (`false`). For more information, see “View Data in the Breakpoints and Watch Window”.

Example: `data.Debug.Watch = true;`

Hierarchy

Machine — Machine that contains data object

`Stateflow.Machine` object

This property is read-only.

Machine that contains the data object, specified as a `Stateflow.Machine` object.

Path — Location of parent in model hierarchy

character vector

This property is read-only.

Location of the parent of the data object in the model hierarchy, specified as a character vector.

Identification

Description — Description

"" (default) | string scalar | character vector

Description for the data object, specified as a string scalar or character vector.

Document — Document link

"" (default) | string scalar | character vector

Document link for the data object, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the data object, specified as data of any type.

SSIdNumber — Session-independent identifier

scalar

This property is read-only.

Session-independent identifier, specified as an integer scalar. Use this property to distinguish the data object from other objects in the model.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Unlike `SSIdNumber`, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>getParent</code>	Identify parent of object
<code>getReferences</code>	Identify references to symbol name
<code>renameReferences</code>	Rename symbol and update references to symbol name
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment

Examples

Add Data to Chart

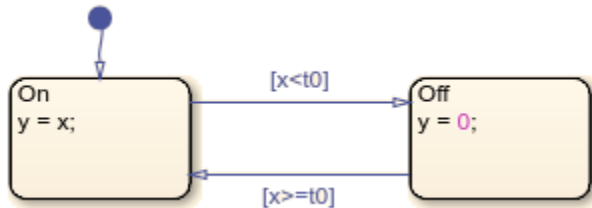
Add a data object to the chart `ch`. Specify its name, scope, and data type.

```
data = Stateflow.Data(ch);  
data.Name = "x";  
data.Scope = "Input";
```

```
data.Props.Type.Method = "Built-in";
data.DataType = "single";
```

Rename Data and Update References in Chart

Rename and update the references to the chart output `y`.



Open the model and access the `Stateflow.Data` object for the chart output `y`.

```
open_system("sfRectifyAPIExample")
data = find(sfroot, "-isa", "Stateflow.Data", Scope="Output");
data.Name
```

```
ans =
'y'
```

Find the locations where the chart refers to the chart output.

```
references = getReferences(data)

references=2x1 object
2x1 SymbolReference array with properties:

    Position
    WhereUsed
```

Display the names and entry actions of the states that refer to the chart output.

```
whereUsed = [references.WhereUsed];
classes = arrayfun(@class,whereUsed,UniformOutput=false);
idx = (classes=="Stateflow.State");
states = whereUsed(idx);
get(states,{"Name" "EntryAction"})

ans = 2x2 cell
    {'On' }    {'y = x;'}
    {'Off'}    {'y = 0;'}
  
```

Change the Name property of the chart output to `"z"` and update the references to the output in the chart.

```
renameReferences(data, "z")
data.Name
```

```
ans =  
'z'
```

Display the names and modified entry actions of the states that refer to the chart output.

```
get(states, {"Name" "EntryAction"})
```

```
ans = 2x2 cell  
    {'On' }    {'z = x;'}  
    {'Off'}   {'z = 0;'}  
    
```

Version History

Introduced before R2006a

R2023a: New object functions

The object functions `getReferences` and `renameReferences` find and update the locations, such as state and transition actions, where the chart refers to the name of a data object:

- The object function `getReferences` returns the locations where a chart refers to a data name.
- The object function `renameReferences` renames a data object and updates all references to the data name in the chart.

R2021b: Stateflow no longer supports creating machine-parented data

Errors starting in R2021b

The `Stateflow.Data` function does not support arguments of type `Stateflow.Machine`. The presence of machine-parented data in a model prevents the reuse of generated code and other code optimizations. This type of data is also incompatible with many Simulink and Stateflow features. To make Stateflow data accessible to other charts and blocks in a model, use data store memory. For more information, see “Access Data Store Memory from a Chart”.

See Also

`Stateflow.Box` | `Stateflow.Chart` | `Stateflow.EMFunction` | `Stateflow.Function` |
`Stateflow.SimulinkBasedState` | `Stateflow.SLFunction` | `Stateflow.State` |
`Stateflow.TruthTable`

Topics

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

“Add Stateflow Data”

“Set Data Properties”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.DataArray

Array properties for data and messages

Description

Use a `Stateflow.DataArray` object to specify the array properties for a data object or message.

Creation

Each data object and message has its own `Stateflow.DataArray` object. To access the `Stateflow.DataArray` object, use the `Props.Array` property for the `Stateflow.Data` or `Stateflow.Message` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Size – Size

" -1" (default) | string scalar | character vector

Size of the data object or message data, specified as a string scalar or character vector. For more information, see “Specify Size of Stateflow Data”.

IsDynamic – Whether data object has variable size

false or 0 (default) | true or 1

Whether the data object has variable size, specified as a numeric or logical 1 (true) or 0 (false). This property is equivalent to the **Variable Size** check box in the **Property Inspector**, the Model Explorer, or the Data properties dialog box. For more information, see “Declare Variable-Size Data in Stateflow Charts”.

FirstIndex – Index for first element of array

string scalar | character vector

Index for the first element of the array data object, specified as a string scalar or character vector. This property applies only to array data in charts that use C as the action language. For more information, see “Save final value to base workspace”.

Examples

Specify Size of Data

Access the `Stateflow.Props` and `Stateflow.DataArray` objects for the `Stateflow.Data` object `x`.

```
properties = x.Props;  
array = properties.Array;
```

Specify the size of the data object.

```
array.size = "[2 3]";
```

Version History

Introduced before R2006a

See Also

[Stateflow.Data](#) | [Stateflow.Message](#)

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Specify Size of Stateflow Data”

“Declare Variable-Size Data in Stateflow Charts”

“Save final value to base workspace”

Stateflow.DataDebug

Debugger properties for data

Description

Use a `Stateflow.DataDebug` object to specify the debugger properties for a data object.

Creation

Each data object has its own `Stateflow.DataDebug` object. To access the `Stateflow.DataDebug` object, use the `Debug` property for the `Stateflow.Data` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Watch — Whether to track data object

`false` or `0` (default) | `true` or `1`

Whether to track the value of the data object in the Breakpoints and Watch window, specified as a numeric or logical `1` (`true`) or `0` (`false`). For more information, see “View Data in the Breakpoints and Watch Window”.

Examples

Add Data to Breakpoints and Watch Window

Access the `Stateflow.DataDebug` object for the `Stateflow.Data` object `x`.

```
debug = x.Debug;
```

Add the data object to the Breakpoints and Watch window.

```
debug.Watch = true;
```

Version History

Introduced before R2006a

See Also

`Stateflow.Data`

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“View Data in the Breakpoints and Watch Window”

Stateflow.DataProps

Data specification properties for data and messages

Description

Use a `Stateflow.DataProps` object to specify the data properties for a data object or message.

Creation

Each data object and message has its own `Stateflow.DataProps` object. To access the `Stateflow.DataProps` object, use the `Props` property for the `Stateflow.Data` or `Stateflow.Message` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Type — Data type properties

`Stateflow.DataType` object

Data type properties, specified as a `Stateflow.DataType` object with these properties:

- **Method** — Method for setting the type of the data object or message, specified as a string scalar or character vector.
 - For local, input, output, or parameter data, use "Inherited", "Built-in", "Bus Object", "Enumerated", "Expression", or "Fixed point".
 - For constant data, use "Built-in", "Expression", or "Fixed point".
 - For data store memory data, use "Inherited".
 - For messages, use "Inherited", "Built-in", "Bus Object", "Enumerated", "Expression", or "Fixed point".

This property is equivalent to the **Mode** field of the Data Type Assistant in the Model Explorer and the Data properties dialog box. For more information, see “Specify Type of Stateflow Data”.

- **BusObject** — Name of the Simulink.Bus object that defines the data object or message data, specified as a string scalar or character vector. This property applies only when the `Method` property of the data object is "Bus Object". For more information, see “Access Bus Signals Through Stateflow Structures”.
- **EnumType** — Name of the enumerated type that defines the data object or message data, specified as a string scalar or character vector. This property applies only when the `Method` property of the data object is "Enumerated". For more information, see “Reference Values by Name by Using Enumerated Data”.

- **Expression** — Expression that evaluates to the data type of the data object or message data, specified as a string scalar or character vector. This property applies only when the **Method** property of the data object is "Expression". For more information, see "Specify Data Properties by Using MATLAB Expressions".
- **Signed** — Signedness, specified as a numeric or logical 1 (**true**) or 0 (**false**). This property applies only when the **Method** property of the data object is "Fixed point". For more information, see "Fixed-Point Data in Stateflow Charts".
- **WordLength** — Word length, in bits, specified as a string scalar or character vector. This property applies only when the **Method** property of the data object is "Fixed point". For more information, see "Fixed-Point Data in Stateflow Charts".
- **Fixpt.ScalingMode** — Method for scaling the fixed-point data object or message data, specified as "Binary point", "Slope and bias", or "None". This property applies only when the **Method** property of the data object is "Fixed point". For more information, see "Fixed-Point Data in Stateflow Charts".
- **Fixpt.FractionLength** — Fraction length, in bits, specified as a string scalar or character vector. This property applies only when the **Method** property is "Fixed point" and the **Fixpt.ScalingMode** property is "Binary point".
- **Fixpt.Slope** — Slope, specified as a string scalar or character vector. This property applies only when the **Method** property is "Fixed point" and the **Fixpt.ScalingMode** property is "Slope and bias".
- **Fixpt.Bias** — Bias, specified as a string scalar or character vector. This property applies only when the **Method** property is "Fixed point" and the **Fixpt.ScalingMode** property is "Slope and bias".
- **Fixpt.Lock** — Whether to prevent replacement of the fixed-point type with an autoscaled type chosen by the Fixed-Point Tool (Fixed-Point Designer), specified as a numeric or logical 1 (**true**) or 0 (**false**). This property applies only when the **Method** property of the data object is "Fixed point".

Array — Array properties

Stateflow.DataArray object

Array properties, specified as a Stateflow.DataArray object with these properties:

- **Size** — Size of the data object or message data, specified as a string scalar or character vector. For more information, see "Specify Size of Stateflow Data".
- **IsDynamic** — Whether the data object has variable size, specified as a numeric or logical 1 (**true**) or 0 (**false**). This property is equivalent to the **Variable Size** check box in the **Property Inspector**, the Model Explorer, or the Data properties dialog box. For more information, see "Declare Variable-Size Data in Stateflow Charts".
- **FirstIndex** — Index for the first element of the array data object, specified as a string scalar or character vector. This property applies only to array data in charts that use C as the action language. For more information, see "Save final value to base workspace".

Complexity — Whether data object or message accepts complex values

"Off" (default) | "On"

Whether the data object or message accepts complex values, specified as "On" or "Off". For more information, see "Complex Data in Stateflow Charts".

InitialValue — Initial value

"" (default) | string scalar | character vector

Initial value, specified as a string scalar or character vector.

Range — Range of acceptable values

Stateflow.DataRange object

Range of acceptable values for the data object, specified as a Stateflow.DataRange object with these properties:

- **Minimum** — Minimum value, specified as a string scalar or character vector.
- **Maximum** — Maximum value, specified as a string scalar or character vector.

This property does not apply to message data. For more information, see “Limit range”.

ResolveToSignalObject — Whether data object resolves to Simulink.Signal object

false or 0 (default) | true or 1

Whether the data object resolves to a Simulink.Signal object that you define in the model or base workspace, specified as a numeric or logical 1 (true) or 0 (false). This property does not apply to message data. For more information, see “Resolve Data Properties from Simulink Signal Objects”.

Unit — Unit of measurement for input and output data

Stateflow.Unit object

Unit of measurement for input and output data objects, specified as a Stateflow.Unit object with this property:

- **Name** — Name of the unit of measurement, specified as a string scalar or character vector.

This property applies only to input and output data. For more information, see “Specify Units for Stateflow Data”.

Examples

Specify Fixed-Point Data Type

Access the Stateflow.Props, Stateflow.DataType, and Stateflow.FixptType objects for the Stateflow.Data object x.

```
properties = x.Props;
type = properties.Type;
fixpt = type.Fixpt;
```

Specify the fixed-point properties.

```
type.Method = "Fixed point";
type.Signed = true;
type.WordLength = "5";
fixpt.ScalingMode = "Binary point";
fixpt.FractionLength = "2";
```

Verify the data type.

```
x.DataType
```

```
ans =  
    'fixdt(1,5,2)'
```

Specify Size of Data

Access the `Stateflow.Props` and `Stateflow.DataArray` objects for the `Stateflow.Data` object `x`.

```
properties = x.Props;  
array = properties.Array;
```

Specify the size of the data object.

```
array.size = "[2 3]";
```

Specify Range of Values for Data

Access the `Stateflow.Props` and `Stateflow.DataRanges` objects for the `Stateflow.Data` object `x`.

```
properties = x.Props;  
range = properties.Range;
```

Specify the minimum and maximum acceptable values.

```
range.Minimum = "0";  
range.Maximum = "1024";
```

Specify Units for Data

Access the `Stateflow.Props` and `Stateflow.Unit` objects for the `Stateflow.Data` object `x`.

```
properties = x.Props;  
unit = properties.Unit;
```

Specify the units as meters.

```
unit.Name = "m";
```

Version History

Introduced before R2006a

See Also

[Stateflow.Data](#) | [Stateflow.Message](#)

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.DataRange

Range of acceptable values for data

Description

Use a `Stateflow.DataRange` object to specify the range of acceptable values for a data object. For more information, see “Limit range”.

Creation

Each data object and message has its own `Stateflow.DataRange` object. However, the object only applies for `Stateflow.Data` objects. To access the `Stateflow.DataRange` object, use the `Props.Range` property for the `Stateflow.Data` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Minimum — Minimum value

"" (default) | string scalar | character vector

Minimum value, specified as a string scalar or character vector.

Maximum — Maximum value

"" (default) | string scalar | character vector

Maximum value, specified as a string scalar or character vector.

Examples

Specify Range of Values for Data

Access the `Stateflow.Props` and `Stateflow.DataRanges` objects for the `Stateflow.Data` object `x`.

```
properties = x.Props;  
range = properties.Range;
```

Specify the minimum and maximum acceptable values.

```
range.Minimum = "0";  
range.Maximum = "1024";
```

Version History

Introduced before R2006a

See Also

Stateflow.Data

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.DataType

Data type properties for data and messages

Description

Use a `Stateflow.DataType` object to specify the data type properties for a data object or message.

Creation

Each data object and message has its own `Stateflow.DataType` object. To access the `Stateflow.DataType` object, use the `Props.Type` property for the `Stateflow.Data` or `Stateflow.Message` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Method — Method for setting data type

"Inherited" (default) | "Built-in" | "Bus Object" | "Enumerated" | "Expression" | "Fixed point"

Method for setting the type of the data object or message, specified as a string scalar or character vector.

- For local, input, output, or parameter data, use "Inherited", "Built-in", "Bus Object", "Enumerated", "Expression", or "Fixed point".
- For constant data, use "Built-in", "Expression", or "Fixed point".
- For data store memory data, use "Inherited".
- For messages, use "Inherited", "Built-in", "Bus Object", "Enumerated", "Expression", or "Fixed point".

This property is equivalent to the **Mode** field of the Data Type Assistant in the Model Explorer and the Data properties dialog box. For more information, see “Specify Type of Stateflow Data”.

BusObject — Name of Simulink.Bus object

"" (default) | string scalar | character vector

Name of the `Simulink.Bus` object that defines the data object or message data, specified as a string scalar or character vector. This property applies only when the `Method` property of the data object is "Bus Object". For more information, see “Access Bus Signals Through Stateflow Structures”.

EnumType — Name of enumerated type

"" (default) | string scalar | character vector

Name of the enumerated type that defines the data object or message data, specified as a string scalar or character vector. This property applies only when the `Method` property of the data object is "Enumerated". For more information, see "Reference Values by Name by Using Enumerated Data".

Expression — Expression that evaluates to data type

" " (default) | string scalar | character vector

Expression that evaluates to the data type of the data object or message data, specified as a string scalar or character vector. This property applies only when the `Method` property of the data object is "Expression". For more information, see "Specify Data Properties by Using MATLAB Expressions".

Signed — Signedness

true or 1 (default) | false or 0

Signedness, specified as a numeric or logical 1 (true) or 0 (false). This property applies only when the `Method` property of the data object is "Fixed point". For more information, see "Fixed-Point Data in Stateflow Charts".

WordLength — Word length

"16" (default) | string scalar | character vector

Word length, in bits, specified as a string scalar or character vector. This property applies only when the `Method` property of the data object is "Fixed point". For more information, see "Fixed-Point Data in Stateflow Charts".

Fixpt — Fixed-point properties

`Stateflow.FixptType` object

Fixed-point properties, specified as a `Stateflow.FixptType` object with these properties:

- **ScalingMode** — Method for scaling the fixed-point data object or message data, specified as "Binary point", "Slope and bias", or "None".
- **FractionLength** — Fraction length, in bits, specified as a string scalar or character vector. This property applies only when the `ScalingMode` property is "Binary point".
- **Slope** — Slope, specified as a string scalar or character vector. This property applies only when the `ScalingMode` property is "Slope and bias".
- **Bias** — Bias, specified as a string scalar or character vector. This property applies only when the `ScalingMode` property is "Slope and bias".
- **Lock** — Whether to prevent replacement of the fixed-point type with an autoscaled type chosen by the Fixed-Point Tool (Fixed-Point Designer), specified as a numeric or logical 1 (true) or 0 (false).

This property applies only when the `Method` property of the data object is "Fixed point". For more information, see "Fixed-Point Data in Stateflow Charts".

Examples

Specify Fixed-Point Data Type

Access the `Stateflow.Props`, `Stateflow.DataType`, and `Stateflow.FixptType` objects for the `Stateflow.Data` object `x`.

```
properties = x.Props;  
type = properties.Type;  
fixpt = type.Fixpt;
```

Specify the fixed-point properties.

```
type.Method = "Fixed point";  
type.Signed = true;  
type.WordLength = "5";  
fixpt.ScalingMode = "Binary point";  
fixpt.FractionLength = "2";
```

Verify the data type.

```
x.DataType
```

```
ans =  
    'fixdt(1,5,2)'
```

Version History

Introduced before R2006a

See Also

Stateflow.Data | Stateflow.Message

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.Editor

Graphical aspects of a chart or state transition table

Description

Use the `Stateflow.Editor` object to access the graphical aspects of a Stateflow chart or state transition table. You can use the `Stateflow.Editor` object to control the position, size, and magnification level of the Stateflow Editor window.

Creation

Each chart has its own `Stateflow.Editor` object. When you create a chart, an `Stateflow.Editor` object is automatically created for it. To access the `Stateflow.Editor` object, use the `Editor` property for the chart. For example, if `ch` is a `Stateflow.Chart` or `Stateflow.StateTransitionTableChart` object, enter:

```
editor = ch.Editor;
```

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

WindowPosition — Position and size of window

[left top width height]

Position and size of the Stateflow editor window, specified as a four-element numeric vector of the form [left top width height].

ZoomFactor — Magnification level

scalar

Magnification level of the chart or state transition table in the editor, specified as a scalar value between 0.5 and 10. A value of 1 corresponds to a magnification of 100%.

Object Functions

`zoomIn` Zoom in on Stateflow chart
`zoomOut` Zoom out on Stateflow chart

Examples

Zoom in on Stateflow Chart

Increase the magnification level of a nonempty chart `ch`.

```
editor = ch.Editor;  
zoomIn(editor)
```

If the magnification level for the chart was initially 100%, this command increases it to 130%.

Zoom out on Stateflow Chart

Decrease the magnification level of a nonempty chart `ch`.

```
editor = ch.Editor;  
zoomOut(editor)
```

If the magnification level for the chart was initially 100%, this command decreases it to 76.9%.

Set Zoom Factor

Set the `ZoomFactor` property for a nonempty chart `ch` to an absolute magnification level of 150%.

```
editor = ch.Editor;  
editor.ZoomFactor = 1.5;
```

Version History

Introduced before R2006a

See Also

`Stateflow.Chart` | `Stateflow.StateTransitionTableChart`

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.EMChart

Stateflow interface to MATLAB Function block

Description

Use `Stateflow.EMChart` objects to configure MATLAB Function blocks through the Stateflow programmatic interface.

MATLAB Function blocks define custom functionality in Simulink models. Use these blocks when:

- You have an existing MATLAB function that models custom functionality, or it is easy for you to create such a function.
- Your model requires custom functionality that is not or cannot be captured in the Simulink graphical language.
- You find it easier to model custom functionality by using a MATLAB function than by using a Simulink block diagram.
- The custom functionality that you want to model does not include continuous or discrete dynamic states. To model dynamic states, use S-functions. See “Create and Configure MATLAB S-Functions” (Simulink).

For more information, see “Implement MATLAB Functions in Simulink with MATLAB Function Blocks” (Simulink).

Tip You can also configure the properties of a MATLAB Function block programmatically by using a `MATLABFunctionConfiguration` object. This object provides a direct interface to the properties of a MATLAB Function block. For more information, see “Configure MATLAB Function Blocks Programmatically” (Simulink).

Creation

Each MATLAB Function block has its own `Stateflow.EMChart` object. When you add a MATLAB Function block to a Simulink model, a `Stateflow.EMChart` object is automatically created for it. For example, you can use the function `add_block` to add a MATLAB Function with the name `MATLAB Function` to a model called `myModel`:

```
add_block("simulink/User-Defined Functions/MATLAB Function", ...
         "myModel/MATLAB Function")
```

Then, to access the `Stateflow.EMChart` object, call the `find` function for the `Simulink.Root` object:

```
block = find(sfroot, "-isa", "Stateflow.EMChart", ...
            "Path", "myModel/MATLAB Function");
```

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API

objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Name — Name of MATLAB Function block

"MATLAB Function" (default) | string scalar | character vector

Name of the MATLAB Function block, specified as a string scalar or character vector.

Script — Code for MATLAB Function block

string scalar | character vector

Code for the MATLAB Function block, specified as a string scalar or character vector. To enter multiple lines of code, you can:

- Call the MATLAB function `sprintf` and use `\n` to insert newline characters:

```
str = sprintf("function y=f(x)\ny=x+1;\nend");
block.Script = str;
```

- Enter a concatenated text expression that uses the function `newline` to create newline characters:

```
str = "function y=f(x)" + newline + ...
      "y=x+1;" + newline + ...
      "end";
block.Script = str;
```

SupportVariableSizing — Whether MATLAB Function block supports variable-size data

true or 1 (default) | false or 0

Whether the MATLAB Function block supports variable-size data, specified as a numeric or logical 1 (true) or 0 (false). For more information, see “Declare Variable-Size MATLAB Function Block Variables” (Simulink).

AllowDirectFeedthrough — Whether MATLAB Function block supports direct feedthrough semantics

true or 1 (default) | false or 0

Whether the MATLAB Function block supports direct feedthrough semantics, specified as a numeric or logical 1 (true) or 0 (false). For more information, see “Allow direct feedthrough” (Simulink).

VectorOutputs1D — Whether MATLAB Function block outputs column vectors as one-dimensional data

false or 0 (default) | true or 1

Whether the MATLAB Function block outputs column vectors as one-dimensional data, specified as a numeric or logical 0 (false) or 1 (true). For more information, see “Interpret output column vectors as one-dimensional data” (Simulink).

TreatDimensionOfLengthOneAsFixedSize — Whether MATLAB Function block output variables with at least one dimension of length 1 are fixed size

true or 1 (default) | false or 0

Whether MATLAB Function block output variables with at least one dimension of length 1 are fixed size, specified as a numeric or logical 0 (false) or 1 (true). When this property is true, the object

sets variables that are variable size in the block with a dimension of 1 to fixed size. When this property is `false`, variables in the block that have the **Variable size** property enabled are always variable size. Prior to R2023a, the object treats variables with at least one dimension of length 1 as fixed size.

This property only affects output variables that have the **Variable size** property enabled. See “Variable size” (Simulink).

Interface

Inputs — Input arguments

array of `Stateflow.Data` objects

This property is read-only.

Input arguments of the MATLAB Function block, specified as an array of `Stateflow.Data` objects. The value of this property depends on the inputs defined in the `Script` property for the block.

Outputs — Output arguments

array of `Stateflow.Data` objects

This property is read-only.

Output arguments of the MATLAB Function block, specified as an array of `Stateflow.Data` objects. The value of this property depends on the outputs defined in the `Script` property for the block.

Discrete and Continuous-Time Semantics

ChartUpdate — Activation method for MATLAB Function block

"INHERITED" (default) | "CONTINUOUS" | "DISCRETE"

Activation method for the MATLAB Function block, specified as "CONTINUOUS", "DISCRETE", or "INHERITED". For more information, see “Update method” (Simulink).

SampleTime — Sample time for activating MATLAB Function block

"-1" (default) | string scalar | character vector

Sample time for activating the MATLAB Function block, specified as a string scalar or character vector. This property applies only when the `ChartUpdate` property for the MATLAB function is "DISCRETE".

Integer and Fixed-Point Data

SaturateOnIntegerOverflow — Whether data saturates on integer overflow

true or 1 (default) | false or 0

Whether the data in the MATLAB Function block saturates on integer overflow, specified as a numeric or logical 1 (true) or 0 (false). When this property is disabled, the data in the function wraps on integer overflow. For more information, see “Saturate on integer overflow” (Simulink).

TreatAsFi — Inherited Simulink signals to treat as fi objects

"Fixed-point" (default) | "Fixed-point & Integer"

Inherited Simulink signals to treat as Fixed-Point Designer `fi` objects, specified as one of these values:

- "Fixed-point" — The MATLAB Function block treats all fixed-point inputs as `fi` objects.
- "Fixed-point & Integer" — The MATLAB Function block treats all fixed-point and integer inputs as `fi` objects.

EmlDefaultFimath — Default fimath properties

"Same as MATLAB Default" (default) | "Other:UserSpecified"

Default `fimath` properties for the MATLAB Function block, specified as one of these values:

- "Same as MATLAB Default" — Use the same `fimath` properties as the current default `fimath` object.
- "Other:UserSpecified" — Use the `InputFimath` property to specify the default `fimath` object.

InputFimath — Default fimath object

string scalar | character vector

Default `fimath` object, specified as a string scalar or character vector. When the `EmlDefaultFimath` property for the MATLAB Function block is "Other:UserSpecified", you can use this property to:

- Enter an expression that constructs a `fimath` object.
- Enter the variable name for a `fimath` object in the MATLAB or model workspace.

Hierarchy

Machine — Machine that contains MATLAB Function block

Stateflow.Machine object

This property is read-only.

Machine that contains the MATLAB Function block, specified as a `Stateflow.Machine` object.

Path — Location of MATLAB Function block in model hierarchy

string scalar | character vector

This property is read-only.

Location of the MATLAB Function block in the model hierarchy, specified as a character vector.

Dirty — Whether MATLAB Function block has changed

true or 1 | false or 0

Whether the MATLAB Function block has changed after being opened or saved, specified as a numeric or logical 1 (true) or 0 (false).

Locked — Whether MATLAB Function block is locked

false or 0 (default) | true or 1

Whether the MATLAB Function block is locked, specified as a numeric or logical 1 (true) or 0 (false). Enable this property to prevent changes in the MATLAB Function block.

Iced — Whether MATLAB Function block is locked

false or 0 (default) | true or 1

This property is read-only.

Whether the MATLAB Function block is locked, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property is equivalent to the property `Locked`, but is used internally to prevent changes in the MATLAB Function block during simulation.

Identification

Description — Description

"" (default) | string scalar | character vector

Description for the MATLAB Function block, specified as a string scalar or character vector.

Document — Document link

"" (default) | string scalar | character vector

Document link for the MATLAB Function block, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the MATLAB Function block, specified as data of any type.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Use this property to distinguish the MATLAB Function block from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>find</code>	Identify specified objects in hierarchy
<code>getChildren</code>	Identify children of object
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment

Examples

Program MATLAB Function Block

Access the `Stateflow.EMChart` object for a MATLAB Function block named `My Function` in a model called `myModel`.

```
block = find(sfroot,"-isa","Stateflow.EMChart", ...  
    "Path","myModel/My Function");
```

Store the MATLAB code to calculate the mean and standard deviation for a vector of values as a string scalar.

```
str = "function [mean,stdev] = stats(vals)" + newline + ...  
    "% Calculates a statistical mean and a standard" + newline + ...
```

```

"% deviation for the values in vals." + newline + newline + ...
"len = length(vals);" + newline + ...
"mean = avg(vals,len);" + newline + ...
"stdev = sqrt(sum(((vals-avg(vals,len)).^2))/len);" + newline + ...
"plot(vals,"-+");" + newline + newline + ...
"function mean = avg(array,size)" + newline + ...
"mean = sum(array)/size;";

```

Populate the block with code by modifying the Script property of the corresponding Stateflow.EMChart object.

```
block.Script = str;
```

Open the function in the **MATLAB Function Block Editor**.

```
view(block)
```

The editor shows this code.

```

function [mean,stdev] = stats(vals)
% Calculates a statistical mean and a standard
% deviation for the values in vals.

len = length(vals);
mean = avg(vals,len);
stdev = sqrt(sum(((vals-avg(vals,len)).^2))/len);
plot(vals,"-+");

function mean = avg(array,size)
mean = sum(array)/size;

```

Import Code from MATLAB Function

Open a Simulink model called myModel.

```
open_system("myModel")
```

Add a MATLAB Function block to myModel named My Function.

```
blockPath = "myModel/My Function";
add_block("simulink/User-Defined Functions/MATLAB Function",blockPath)
```

Populate the block with code from the MATLAB function myFunction.m.

```

block = find(sfroot,"-isa","Stateflow.EMChart", ...
    "Path",blockPath);
block.Script = fileread("myFunction.m");

```

Find Number of MATLAB Function Blocks in Model

Open a Simulink model called myModel.

```
open_system("myModel")
```

Find the MATLAB Function blocks in the model, including the block in the library.

```
blocks = find(sfroot, "-isa", "Stateflow.EMChart");
```

Note This command finds objects for the MATLAB Function blocks in all open models and libraries. To find only the MATLAB Function blocks in `myModel`, close every file except `myModel` or replace `sfroot` with `get_param("myModel", "Object")`.

Count the number of blocks.

```
numel(blocks)
```

Version History

Introduced in R2011a

R2023a: Set output variables of any dimension as variable size

You can now set output variables of any dimension to be variable size by setting the `TreatDimensionOfLengthOneAsFixedSize` property to `false`. Prior to R2023a, the object treats variables with at least one dimension of length 1 as fixed size.

R2021b: Change to output column vectors

You can output column vectors in MATLAB Function blocks as two-dimensional or one-dimensional data with the `VectorOutputs1D` property.

Before R2021b, MATLAB Function blocks always output column vectors as one-dimensional data. After R2021b, MATLAB Function blocks output column vectors as two-dimensional data by default. To maintain the original behavior of the block, set the `VectorOutputs1D` property to `true`.

See Also

Blocks

MATLAB Function

Functions

`sfroot` | `add_block` | `fileread` | `numel`

Objects

`MATLABFunctionConfiguration`

Topics

"Overview of the Stateflow API" on page 1-2

"Specify MATLAB Function Block Properties" (Simulink)

"Summary of Stateflow API Objects and Properties" on page 1-36

Stateflow.EMFunction

MATLAB function in chart, state, box, or function

Description

Use `Stateflow.EMFunction` objects to create MATLAB functions for coding algorithms that are more easily expressed by using MATLAB code instead of the graphical Stateflow constructs. Typical applications include:

- Matrix-oriented calculations
- Data analysis and visualization

You can call a MATLAB function in the actions of states and transitions. For more information, see “Reuse MATLAB Code by Defining MATLAB Functions”.

Creation

Syntax

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

Description

`function = Stateflow.EMFunction(parent)` creates a `Stateflow.EMFunction` object in a parent chart, state, box, or function.

Input Arguments

parent — Parent for new MATLAB function

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object

Parent for the new MATLAB function, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.Function`
- `Stateflow.State`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Name — Name of MATLAB function

"" (default) | string scalar | character vector

Name of the MATLAB function, specified as a string scalar or character vector.

LabelString — Label for MATLAB function

"?" (default) | string scalar | character vector

Label for the MATLAB function, specified as a string scalar or character vector.

Script — Code for MATLAB function

string scalar | character vector

Code for the MATLAB function, specified as a string scalar or character vector. To enter multiple lines of code, you can:

- Call the MATLAB function `sprintf` and use the escape sequence `\n` to insert newline characters:

```
str = sprintf("function y=f(x)\ny=x+1;\nend");  
function.Script = str;
```

- Enter a concatenated text expression that uses the function `newline` to create newline characters:

```
str = "function y=f(x)" + newline + ...  
      "y=x+1;" + newline + ...  
      "end";  
function.Script = str;
```

IsExplicitlyCommented — Whether to comment out MATLAB function

false or 0 (default) | true or 1

Whether to comment out the MATLAB function, specified as a numeric or logical 1 (`true`) or 0 (`false`). Setting this property to `true` is equivalent to right-clicking the MATLAB function and selecting **Comment Out**. For more information, see “Comment Out Objects in a Stateflow Chart”.

IsImplicitlyCommented — Whether MATLAB function is implicitly commented out

true or 1 | false or 0

This property is read-only.

Whether the MATLAB function is implicitly commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). The MATLAB function is implicitly commented out when you explicitly comment out an object that contains it. If the MATLAB function is contained in an atomic subchart or an atomic box, this property is `false` unless the explicitly commented object is also contained in the atomic subchart or atomic box.

IsCommented — Whether MATLAB function is commented out


true or 1 | false or 0

This property is read-only.

Whether the MATLAB function is commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property is `true` when either `IsExplicitlyCommented` or `IsImplicitlyCommented` is `true`.

CommentText — Comment text

"" (default) | string scalar | character vector

Comment text for the MATLAB function, specified as a string scalar or character vector. This property applies only when the `IsExplicitlyCommented` property is `true`. In the Stateflow Editor, when you point to the comment badge  on the MATLAB function, the text appears as a tooltip. When you set the `IsExplicitlyCommented` property to `false`, the value of `CommentText` reverts to "".

Graphical Appearance**Position — Position and size of MATLAB function**

[0 0 90 60] (default) | [left top width height]

Position and size of the MATLAB function, specified as a four-element numeric vector of the form [left top width height].

BadIntersection — Whether MATLAB function intersects a box, state, or function

true or 1 | false or 0

This property is read-only.

Whether the MATLAB function graphically intersects a box, state, or function, specified as a numeric or logical 1 (`true`) or 0 (`false`).

FontSize — Font size for MATLAB function label

scalar

Font size for the MATLAB function label, specified as a scalar. The `StateFont.Size` property of the chart that contains the graphical function sets the initial value of this property.

Integer and Fixed-Point Data**SaturateOnIntegerOverflow — Whether data saturates on integer overflow**

true or 1 (default) | false or 0

Whether the data in the MATLAB function saturates on integer overflow, specified as a numeric or logical 1 (`true`) or 0 (`false`). When this property is disabled, the data in the function wraps on integer overflow. For more information, see "Handle Integer Overflow for Chart Data".

This property applies only when the `ActionLanguage` of the chart that contains the function is "C". Otherwise, the behavior of data depends on the value of the `SaturateOnIntegerOverflow` property for the chart.

EmlDefaultFimath — Default fimath properties

"Same as MATLAB Default" (default) | "Other:UserSpecified"

Default `fimath` properties for the MATLAB function, specified as one of these values:

- "Same as MATLAB Default" — Use the same `fimath` properties as the current default `fimath` object.
- "Other:UserSpecified" — Use the `InputFimath` property to specify the default `fimath` object.

This property applies only when the `ActionLanguage` of the chart that contains the function is "C". Otherwise, the behavior of data depends on the value of the `EmlDefaultFimath` property for the chart.

InputFimath — Default fimath object

string scalar | character vector

Default `fimath` object, specified as a string scalar or character vector. When the `EmlDefaultFimath` property for the MATLAB function is "Other:UserSpecified", you can use this property to:

- 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 `ActionLanguage` of the chart that contains the function is "C". Otherwise, the behavior of data depends on the value of the `InputFimath` property for the chart.

Code Generation**InlineOption — Appearance in generated code**

"Auto" (default) | "Function" | "Inline"

Appearance of the MATLAB function in generated code, specified as one of these values:

- "Auto" — An internal calculation determines the appearance of the function in generated code.
- "Function" — The function is implemented as a separate C function.
- "Inline" — Calls to the function are replaced by code as long as the function is not part of a recursion.

For more information, see "Inline State Functions in Generated Code" (Simulink Coder).

Hierarchy**Chart — Chart that contains MATLAB function**

`Stateflow.Chart` object

This property is read-only.

Chart that contains the MATLAB function, specified as a `Stateflow.Chart` object.

Subviewer — Subviewer for MATLAB function

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object

This property is read-only.

Subviewer for the MATLAB function, specified as a `Stateflow.Chart`, `Stateflow.State`, `Stateflow.Box`, or `Stateflow.Function` object. The subviewer is the chart or subchart where you can graphically view the MATLAB function.

Machine — Machine that contains MATLAB function

`Stateflow.Machine` object

This property is read-only.

Machine that contains the MATLAB function, specified as a `Stateflow.Machine` object.

Path — Location of parent in model hierarchy

character vector

This property is read-only.

Location of the parent of the MATLAB function in the model hierarchy, specified as a character vector.

Identification

Description — Description

" " (default) | string scalar | character vector

Description for the MATLAB function, specified as a string scalar or character vector.

Document — Document link

" " (default) | string scalar | character vector

Document link for the MATLAB function, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the MATLAB function, specified as data of any type.

SSIdNumber — Session-independent identifier

scalar

This property is read-only.

Session-independent identifier, specified as an integer scalar. Use this property to distinguish the MATLAB function from other objects in the model.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Unlike `SSIdNumber`, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>find</code>	Identify specified objects in hierarchy
<code>getChildren</code>	Identify children of object
<code>getParent</code>	Identify parent of object
<code>getReferences</code>	Identify references to symbol name
<code>renameReferences</code>	Rename symbol and update references to symbol name
<code>commentedBy</code>	Identify objects that implicitly comment out a graphical object
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>highlight</code>	Highlight graphical object
<code>fitToView</code>	Zoom in on graphical object

Examples

Add MATLAB Function to Chart

Add a MATLAB function in the chart `ch`. Set its label to "[y1,y2] = f(x1,x2,x3)".

```
function = Stateflow.EMFunction(ch);  
function.LabelString = "[y1,y2] = f(x1,x2,x3)";
```

Version History

Introduced before R2006a

R2023a: New object functions and properties

Errors starting in R2023a

`Stateflow.EMFunction` objects have new object functions and properties:

- The object function `getReferences` returns the locations where a chart refers to the name of a MATLAB function.
- The object function `renameReferences` renames a MATLAB function and updates all references to the function name in the chart.
- The object function `commentedBy` identifies the explicitly commented objects that cause a MATLAB function to be commented out.
- The property `IsCommented` indicates whether a MATLAB function is commented out. This property replaces the object function `isCommented`.

See Also

`Stateflow.Box` | `Stateflow.Chart` | `Stateflow.Function` | `Stateflow.State`

Topics

"Overview of the Stateflow API" on page 1-2

"Reuse MATLAB Code by Defining MATLAB Functions"

"Summary of Stateflow API Objects and Properties" on page 1-36

Stateflow.ChartBreakpoints

Breakpoint properties for chart or state transition table

Description

Use a `Stateflow.ChartBreakpoints` object to specify the breakpoint properties for a chart or state transition table. For more information, see “Set Breakpoints to Debug Charts”.

Creation

Each chart and state transition table has its own `Stateflow.ChartBreakpoints` object. To access the `Stateflow.ChartBreakpoints` object, use the `Debug.Breakpoints` property of the `Stateflow.Chart` or `Stateflow.StateTransitionTableChart` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

OnEntry — Whether to set On Chart Entry breakpoint

`false` or 0 (default) | `true` or 1

Whether to set the On Chart Entry breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).

Examples

Set Breakpoint for Chart

Access the `Stateflow.ChartDebug` and `Stateflow.ChartBreakpoints` objects for the `Stateflow.Chart` object `ch`.

```
debug = ch.Debug;  
breakpoints = debug.Breakpoints;
```

Set the On Chart Entry breakpoint.

```
breakpoints.OnEntry = true;
```

Version History

Introduced before R2006a

See Also

Stateflow.Chart | Stateflow.StateTransitionTableChart

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Set Breakpoints to Debug Charts”

Stateflow.ChartDebug

Debugger properties for chart or state transition table

Description

Use a `Stateflow.ChartDebug` object to specify the debugger properties for a chart or state transition table.

Creation

Each chart and state transition table has its own `Stateflow.ChartDebug` object. To access the `Stateflow.ChartDebug` object, use the `Debug` property for the `Stateflow.Chart` or `Stateflow.StateTransitionTableChart` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Breakpoints — Breakpoint properties

`Stateflow.ChartBreakpoints` object

Breakpoint properties for the chart or state transition table, specified as a `Stateflow.ChartBreakpoints` object with this property:

- **OnEntry** — Whether to set the On Chart Entry breakpoint, specified as a numeric or logical 1 (true) or 0 (false).

For more information, see “Set Breakpoints to Debug Charts”.

Examples

Set Breakpoint for Chart

Access the `Stateflow.ChartDebug` and `Stateflow.ChartBreakpoints` objects for the `Stateflow.Chart` object `ch`.

```
debug = ch.Debug;  
breakpoints = debug.Breakpoints;
```

Set the On Chart Entry breakpoint.

```
breakpoints.OnEntry = true;
```

Version History

Introduced before R2006a

See Also

Stateflow.Chart | Stateflow.StateTransitionTableChart

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Set Breakpoints to Debug Charts”

Stateflow.Event

Event in chart, state, or box

Description

Use `Stateflow.Event` objects to trigger actions in one of these objects:

- A parallel state in a Stateflow chart
- Another Stateflow chart
- A Simulink triggered or function-call subsystem

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

Creation

Syntax

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

Description

`event = Stateflow.Event(parent)` creates a `Stateflow.Event` object in a parent chart, state, or box.

Input Arguments

parent — Parent for new event

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object

Parent for the new event, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.State`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Interface

Name — Name of event

"event" (default) | string scalar | character vector

Name of the event, specified as a string scalar or character vector.

Scope — Scope of event

"Local" (default) | "Input" | "Output"

Scope of the event, specified as "Local", "Input", or "Output". For more information, see "Scope".

Trigger — Type of trigger

"Function call" (default) | "Rising" | "Falling" | "Either"

Type of trigger associated with the event, specified as a string scalar or character vector that depends on the scope of the data:

- For input events, use "Function call", "Rising", "Falling", or "Either".
- For output events, use "Function call" or "Either".

This property does not apply to local events. For more information, see "Trigger".

Port — Port index for event

scalar

Port index for the event, specified as an integer scalar. This property applies only to input and output events. For more information, see "Port".

Debugging**Debug — Debugger properties**

Stateflow.EventDebug object

Debugger properties for the event, specified as a Stateflow.EventDebug object with these properties:

- **Breakpoints.StartBroadcast** — Whether to set the Start of Broadcast breakpoint, specified as a numeric or logical 1 (true) or 0 (false).
- **Breakpoints.EndBroadcast** — Whether to set the End of Broadcast breakpoint, specified as a numeric or logical 1 (true) or 0 (false).

For more information, see "Set Breakpoints to Debug Charts".

```
Example: event.Debug.Breakpoints.StartBroadcast = true;
```

```
Example: event.Debug.Breakpoints.EndBroadcast = true;
```

Hierarchy**Machine — Machine that contains event**

Stateflow.Machine object

This property is read-only.

Machine that contains the event, specified as a Stateflow.Machine object.

Path — Location of parent in model hierarchy

character vector

This property is read-only.

Location of the parent of the event in the model hierarchy, specified as a character vector.

Identification

Description — Description

"" (default) | string scalar | character vector

Description for the event, specified as a string scalar or character vector.

Document — Document link

"" (default) | string scalar | character vector

Document link for the event, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the event, specified as data of any type.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Use this property to distinguish the event from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

getParent	Identify parent of object
getReferences	Identify references to symbol name
renameReferences	Rename symbol and update references to symbol name
dialog	Open properties dialog box
view	Display object in editing environment

Examples

Add Event to Chart

Add a event to the chart ch. Specify its name and scope.

```
event = Stateflow.Event(ch);
event.Name = "E";
event.Scope = "Input";
```

Version History

Introduced before R2006a

R2023a: New object functions

The object functions `getReferences` and `renameReferences` find and update the locations, such as state and transition actions, where the chart refers to the name of an event:

- The object function `getReferences` returns the locations where a chart refers to an event name.
- The object function `renameReferences` renames an event and updates all references to the event name in the chart.

See Also

`Stateflow.Box` | `Stateflow.Chart` | `Stateflow.State`

Topics

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

“Synchronize Model Components by Broadcasting Events”

“Set Properties for an Event”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.EventBreakpoints

Breakpoint properties for event

Description

Use a `Stateflow.EventBreakpoints` object to specify the breakpoint properties for an event. For more information, see “Set Breakpoints to Debug Charts”.

Creation

Each event has its own `Stateflow.EventBreakpoints` object. To access the `Stateflow.EventBreakpoints` object, use the `Debug.Breakpoints` property of the `Stateflow.Event` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

StartBroadcast — Whether to set Start of Broadcast breakpoint

`false` or 0 (default) | `true` or 1

Whether to set the `Start` of Broadcast breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).

EndBroadcast — Whether to set End of Broadcast breakpoint

`false` or 0 (default) | `true` or 1

Whether to set the `End` of Broadcast breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).

Examples

Set Breakpoints for Event

Access the `Stateflow.EventDebug` and `Stateflow.EventBreakpoints` objects for the `Stateflow.Event` object event.

```
debug = event.Debug;
breakpoints = debug.Breakpoints;
```

Set the `Start` of Broadcast and `End` of Broadcast breakpoints.

```
breakpoints.StartBroadcast = true;  
breakpoints.EndBroadcast = true;
```

Version History

Introduced before R2006a

See Also

Stateflow.Event

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Set Breakpoints to Debug Charts”

Stateflow.EventDebug

Debugger properties for event

Description

Use a `Stateflow.EventDebug` object to specify the debugger properties for an event.

Creation

Each event has its own `Stateflow.EventDebug` object. To access the `Stateflow.EventDebug` object, use the `Debug` property for the `Stateflow.Event` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Breakpoints — Breakpoint properties

`Stateflow.EventBreakpoints` object

Breakpoint properties for the event, specified as a `Stateflow.EventBreakpoints` object with these properties:

- **StartBroadcast** — Whether to set the Start of Broadcast breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).
- **EndBroadcast** — Whether to set the End of Broadcast breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).

For more information, see “Set Breakpoints to Debug Charts”.

Examples

Set Breakpoints for Event

Access the `Stateflow.EventDebug` and `Stateflow.EventBreakpoints` objects for the `Stateflow.Event` object event.

```
debug = event.Debug;  
breakpoints = debug.Breakpoints;
```

Set the Start of Broadcast and End of Broadcast breakpoints.

```
breakpoints.StartBroadcast = true;  
breakpoints.EndBroadcast = true;
```

Version History

Introduced before R2006a

See Also

Stateflow.Event

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Set Breakpoints to Debug Charts”

Stateflow.FixptType

Fixed-point properties for data and messages

Description

Use a `Stateflow.FixptType` object to specify the fixed-point properties for a data object or message. For more information, see “Fixed-Point Data in Stateflow Charts”.

Creation

Each data object and message has its own `Stateflow.FixptType` object. To access the `Stateflow.FixptType` object, use the `Props.Type.Fixpt` property for the `Stateflow.Data` or `Stateflow.Message` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

ScalingMode — Method for scaling fixed-point data

"None" (default) | "Binary point" | "Slope and bias"

Method for scaling the fixed-point data object or message data, specified as "Binary point", "Slope and bias", or "None".

FractionLength — Fraction length

"" (default) | string scalar | character vector

Fraction length, in bits, specified as a string scalar or character vector. This property applies only to fixed-point data when the `ScalingMode` property is "Binary point".

Slope — Slope

"" (default) | string scalar | character vector

Slope, specified as a string scalar or character vector. This property applies only to fixed-point data when the `ScalingMode` property is "Slope and bias".

Bias — Bias

"" (default) | string scalar | character vector

Bias, specified as a string scalar or character vector. This property applies only to fixed-point data when the `ScalingMode` property is "Slope and bias".

Lock — Whether to prevent replacement of fixed-point type

false or 0 (default) | true or 1

Whether to prevent replacement of the fixed-point type with an autoscaled type chosen by the Fixed-Point Tool (Fixed-Point Designer), specified as a numeric or logical 1 (`true`) or 0 (`false`). For more information, see “Iterative Fixed-Point Conversion Using the Fixed-Point Tool” (Fixed-Point Designer).

Examples

Specify Fixed-Point Data Type

Access the `Stateflow.Props`, `Stateflow.DataType`, and `Stateflow.FixptType` objects for the `Stateflow.Data` object `x`.

```
properties = x.Props;  
type = properties.Type;  
fixpt = type.Fixpt;
```

Specify the fixed-point properties.

```
type.Method = "Fixed point";  
type.Signed = true;  
type.WordLength = "5";  
fixpt.ScalingMode = "Binary point";  
fixpt.FractionLength = "2";
```

Verify the data type.

```
x.DataType  
  
ans =  
    'fixdt(1,5,2)'
```

Version History

Introduced before R2006a

See Also

`Stateflow.Data` | `Stateflow.Message`

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.Function

Graphical function in chart, state, box, or function

Description

Use `Stateflow.Function` objects to create graphical functions that contain control-flow logic and iterative loops. You create graphical functions with flow charts that use connective junctions and transitions. You can call a graphical function in the actions of states and transitions. For more information, see “Reuse Logic Patterns by Defining Graphical Functions”.

Creation

Syntax

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

Description

`function = Stateflow.Function(parent)` creates a `Stateflow.Function` object in a parent chart, state, box, or function.

Input Arguments

parent — Parent for new graphical function

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object

Parent for the new graphical function, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.Function`
- `Stateflow.State`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Name — Name of graphical function

" " (default) | string scalar | character vector

Name of the graphical function, specified as a string scalar or character vector.

LabelString — Label for graphical function`"?" (default) | string scalar | character vector`

Label for the graphical function, specified as a string scalar or character vector.

IsExplicitlyCommented — Whether to comment out graphical function`false or 0 (default) | true or 1`

Whether to comment out the graphical function, specified as a numeric or logical 1 (`true`) or 0 (`false`). Setting this property to `true` is equivalent to right-clicking the graphical function and selecting **Comment Out**. For more information, see “Comment Out Objects in a Stateflow Chart”.

IsImplicitlyCommented — Whether graphical function is implicitly commented out`true or 1 | false or 0`

This property is read-only.


Whether the graphical function is implicitly commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). The graphical function is implicitly commented out when you explicitly comment out an object that contains it. If the graphical function is contained in an atomic subchart or an atomic box, this property is `false` unless the explicitly commented object is also contained in the atomic subchart or atomic box.

IsCommented — Whether graphical function is commented out`true or 1 | false or 0`

This property is read-only.

Whether the graphical function is commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property is `true` when either `IsExplicitlyCommented` or `IsImplicitlyCommented` is `true`.

CommentText — Comment text`"" (default) | string scalar | character vector`

Comment text for the graphical function, specified as a string scalar or character vector. This property applies only when the `IsExplicitlyCommented` property is `true`. In the Stateflow Editor, when you point to the comment badge  on the graphical function, the text appears as a tooltip. When you set the `IsExplicitlyCommented` property to `false`, the value of `CommentText` reverts to `""`.

Graphical Appearance**Position — Position and size of graphical function**`[0 0 90 60] (default) | [left top width height]`

Position and size of the graphical function, specified as a four-element numeric vector of the form `[left top width height]`.

BadIntersection — Whether graphical function intersects a box, state, or function`true or 1 | false or 0`

This property is read-only.

Whether the graphical function graphically intersects a box, state, or function, specified as a numeric or logical 1 (`true`) or 0 (`false`).

IsGrouped — Whether graphical function is a grouped function

`false` or 0 (default) | `true` or 1

Whether the function is a grouped function, specified as a numeric or logical 1 (`true`) or 0 (`false`). When you copy and paste a grouped function, you copy not only the function but all of its contents. For more information, see “Copy and Paste by Grouping” on page 2-28.

IsSubchart — Whether graphical function is a subchart

`false` or 0 (default) | `true` or 1

Whether the function is a subchart, specified as a numeric or logical 1 (`true`) or 0 (`false`).

ContentPreviewEnabled — Whether to display preview of graphical function contents

`false` or 0 (default) | `true` or 1

Whether to display a preview of the graphical function contents, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property applies only when the `IsSubchart` property is `true`.

FontSize — Font size for graphical function label

scalar

Font size for the graphical function label, specified as a scalar. The `StateFont.Size` property of the chart that contains the graphical function sets the initial value of this property.

Debugging

Debug — Debugger properties

`Stateflow.FunctionDebug` object

Debugger properties for the graphical function, specified as a `Stateflow.FunctionDebug` object with this property:

- **Breakpoints.OnDuring** — Whether to set the `During Function Call` breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).

For more information, see “Set Breakpoints to Debug Charts”.

Example: `function.Debug.Breakpoints.OnDuring = true;`

Code Generation

InlineOption — Appearance in generated code

`"Auto"` (default) | `"Function"` | `"Inline"`

Appearance of the graphical function in generated code, specified as one of these values:

- `"Auto"` — An internal calculation determines the appearance of the function in generated code.
- `"Function"` — The function is implemented as a separate C function.
- `"Inline"` — Calls to the function are replaced by code as long as the function is not part of a recursion.

For more information, see “Inline State Functions in Generated Code” (Simulink Coder).

Hierarchy

Chart — Chart that contains graphical function

`Stateflow.Chart` object

This property is read-only.

Chart that contains the graphical function, specified as a `Stateflow.Chart` object.

Subviewer — Subviewer for graphical function

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object

This property is read-only.

Subviewer for the graphical function, specified as a `Stateflow.Chart`, `Stateflow.State`, `Stateflow.Box`, or `Stateflow.Function` object. The subviewer is the chart or subchart where you can graphically view the graphical function.

Machine — Machine that contains graphical function

`Stateflow.Machine` object

This property is read-only.

Machine that contains the graphical function, specified as a `Stateflow.Machine` object.

Path — Location of parent in model hierarchy

character vector

This property is read-only.

Location of the parent of the graphical function in the model hierarchy, specified as a character vector.

Identification

Description — Description

`""` (default) | string scalar | character vector

Description for the graphical function, specified as a string scalar or character vector.

Document — Document link

`""` (default) | string scalar | character vector

Document link for the graphical function, specified as a string scalar or character vector.

Tag — User-defined tag

`[]` (default) | any data type

User-defined tag for the graphical function, specified as data of any type.

SSIDNumber — Session-independent identifier

scalar

This property is read-only.

Session-independent identifier, specified as an integer scalar. Use this property to distinguish the graphical function from other objects in the model.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Unlike `SSIdNumber`, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>find</code>	Identify specified objects in hierarchy
<code>getChildren</code>	Identify children of object
<code>getParent</code>	Identify parent of object
<code>defaultTransitions</code>	Identify default transitions in specified object
<code>getReferences</code>	Identify references to symbol name
<code>renameReferences</code>	Rename symbol and update references to symbol name
<code>commentedBy</code>	Identify objects that implicitly comment out a graphical object
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>highlight</code>	Highlight graphical object
<code>fitToView</code>	Zoom in on graphical object

Examples

Add Graphical Function to Chart

Add a graphical function in the chart `ch`. Set its label to "`[y1,y2] = f(x1,x2,x3)`".

```
function = Stateflow.Function(ch);
function.LabelString = "[y1,y2] = f(x1,x2,x3)";
```

Version History

Introduced before R2006a

R2023a: New object functions and properties

Errors starting in R2023a

`Stateflow.Function` objects have new object functions and properties:

- The object function `getReferences` returns the locations where a chart refers to the name of a graphical function.
- The object function `renameReferences` renames a graphical function and updates all references to the function name in the chart.
- The object function `commentedBy` identifies the explicitly commented objects that cause a graphical function to be commented out.
- The property `IsCommented` indicates whether a graphical function is commented out. This property replaces the object function `isCommented`.

See Also

Stateflow.Box | Stateflow.Chart | Stateflow.State

Topics

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

“Reuse Logic Patterns by Defining Graphical Functions”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.FunctionBreakpoint

Breakpoint properties for graphical or truth table function

Description

Use a `Stateflow.FunctionBreakpoint` object to specify the breakpoint properties for a graphical or truth table function. For more information, see “Set Breakpoints to Debug Charts”.

Creation

Each graphical and truth table function has its own `Stateflow.FunctionBreakpoint` object. To access the `Stateflow.FunctionBreakpoint` object, use the `Debug.Breakpoints` property of the `Stateflow.Function` or `Stateflow.TruthTable` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

OnDuring — Whether to set During Function Call breakpoint

false or 0 (default) | true or 1

Whether to set the `During Function Call` breakpoint, specified as a numeric or logical 1 (true) or 0 (false).

Examples

Set Breakpoint for Graphical Function

Access the `Stateflow.FunctionDebug` and `Stateflow.FunctionBreakpoints` objects for the `Stateflow.Function` object `f`.

```
debug = f.Debug;  
breakpoints = debug.Breakpoints;
```

Set the `During Function Call` breakpoint.

```
breakpoints.OnDuring = true;
```

Version History

Introduced before R2006a

See Also

Stateflow.Function | Stateflow.TruthTable

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Set Breakpoints to Debug Charts”

Stateflow.FunctionDebug

Debugger properties for graphical or truth table function

Description

Use a `Stateflow.FunctionDebug` object to specify the debugger properties for a graphical or truth table function.

Creation

Each graphical or truth table function has its own `Stateflow.FunctionDebug` object. To access the `Stateflow.FunctionDebug` object, use the `Debug` property for the `Stateflow.Function` or `Stateflow.TruthTable` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Breakpoints — Breakpoint properties

`Stateflow.FunctionBreakpoint` object

Breakpoint properties for the graphical or truth table function, specified as a `Stateflow.FunctionBreakpoint` object with this property:

- **OnDuring** — Whether to set the `During Function Call` breakpoint, specified as a numeric or logical 1 (true) or 0 (false).

For more information, see “Set Breakpoints to Debug Charts”.

Examples

Set Breakpoint for Graphical Function

Access the `Stateflow.FunctionDebug` and `Stateflow.FunctionBreakpoints` objects for the `Stateflow.Function` object `f`.

```
debug = f.Debug;  
breakpoints = debug.Breakpoints;
```

Set the `During Function Call` breakpoint.

```
breakpoints.OnDuring = true;
```

Version History

Introduced before R2006a

See Also

Stateflow.Function | Stateflow.TruthTable

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Set Breakpoints to Debug Charts”

Stateflow.Junction

Connective or history junction in chart, state, box, or function

Description

Use `Stateflow.Junction` objects to create junctions that:

- Represent decision points in a transition path
- Record the activity of substates inside a superstate

For more information, see “Combine Transitions and Junctions to Create Branching Paths” and “Resume Prior Substate Activity by Using History Junctions”.

Creation

Syntax

```
junction = Stateflow.Junction(parent)
```

Description

`junction = Stateflow.Junction(parent)` creates a `Stateflow.Junction` object in a parent chart, state, box, or graphical function.

Input Arguments

parent — Parent for new junction

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object

Parent for the new junction, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.Function`
- `Stateflow.State`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Type — Type of junction

"CONNECTIVE" (default) | "HISTORY"

Type of junction, specified as one of these values:

- "CONNECTIVE" — Connective junction that represents a decision point in a transition path
- "HISTORY" — History junction that records the activity of substates inside a superstate

IsExplicitlyCommented — Whether to comment out junction

false or 0 (default) | true or 1

Whether to comment out the junction, specified as a numeric or logical 1 (true) or 0 (false). Setting this property to true is equivalent to right-clicking the junction and selecting **Comment Out**. For more information, see “Comment Out Objects in a Stateflow Chart”.

IsImplicitlyCommented — Whether junction is implicitly commented out

true or 1 | false or 0

This property is read-only.

Whether the junction is implicitly commented out, specified as a numeric or logical 1 (true) or 0 (false). The junction is implicitly commented out when you explicitly comment out an object that contains it. If the junction is contained in an atomic subchart or an atomic box, this property is false unless the explicitly commented object is also contained in the atomic subchart or atomic box.

IsCommented — Whether junction is commented out


true or 1 | false or 0

This property is read-only.

Whether the junction is commented out, specified as a numeric or logical 1 (true) or 0 (false). This property is true when either `IsExplicitlyCommented` or `IsImplicitlyCommented` is true.

CommentText — Comment text

"" (default) | string scalar | character vector

Comment text added to the junction, specified as a string scalar or character vector. This property applies only when the `IsExplicitlyCommented` property is true. In the Stateflow Editor, when you point to the comment badge  on the junction, the text appears as a tooltip. When you set the `IsExplicitlyCommented` property to false, the value of `CommentText` reverts to "".

Graphical Appearance

Position — Position and size of junction

`Stateflow.JunctionPosition` object

Position and size of the junction, specified as a `Stateflow.JunctionPosition` object with these properties:

- **Center** — Position of the center of the junction, specified as a two-element numeric vector [x y] of coordinates relative to the upper left corner of the chart.
- **Radius** — Radius of the junction, specified as a scalar.

Example: `junction.Position.Center = [31.41 27.18];`

Example: `junction.Position.Radius = 16.18;`

ArrowSize — Size of incoming transition arrows

8 (default) | scalar

Size of incoming transition arrows, specified as a scalar.

Hierarchy

Chart — Chart that contains junction

`Stateflow.Chart` object

This property is read-only.

Chart that contains the junction, specified as a `Stateflow.Chart` object.

Subviewer — Subviewer for junction

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object

This property is read-only.

Subviewer for the junction, specified as a `Stateflow.Chart`, `Stateflow.State`, `Stateflow.Box`, or `Stateflow.Function` object. The subviewer is the chart or subchart where you can graphically view the junction.

Machine — Machine that contains junction

`Stateflow.Machine` object

This property is read-only.

Machine that contains the junction, specified as a `Stateflow.Machine` object.

Path — Location of parent in model hierarchy

character vector

This property is read-only.

Location of the parent of the junction in the model hierarchy, specified as a character vector.

Identification

Description — Description

"" (default) | string scalar | character vector

Description for the junction, specified as a string scalar or character vector.

Document — Document link

"" (default) | string scalar | character vector

Document link for the junction, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the junction, specified as data of any type.

SSIdNumber — Session-independent identifier

scalar

This property is read-only.

Session-independent identifier, specified as an integer scalar. Use this property to distinguish the junction from other objects in the model.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Unlike `SSIdNumber`, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>getParent</code>	Identify parent of object
<code>sinkedTransitions</code>	Identify transitions with specified destination
<code>sourcedTransitions</code>	Identify transitions with specified source
<code>commentedBy</code>	Identify objects that implicitly comment out a graphical object
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>highlight</code>	Highlight graphical object
<code>fitToView</code>	Zoom in on graphical object

Examples**Add Connective Junction to Chart**

Add a connective junction in the chart `ch`. Change its size and position.

```
junction = Stateflow.Junction(ch);  
junction.Position.Radius = 16.18;  
junction.Position.Center = [31.41 27.18];
```

Add History Junction to Chart

Add a history junction in the chart `ch`.

```
junction = Stateflow.Junction(ch);  
junction.Type = "HISTORY";
```

Version History

Introduced before R2006a

R2023a: New object function and property*Errors starting in R2023a*

Use the object function `commentedBy` and the property `IsCommented` to investigate commented-out junctions:

- The object function `commentedBy` identifies the explicitly commented objects that cause a junction to be commented out.
- The property `IsCommented` indicates whether a junction is commented out. This property replaces the object function `isCommented`.

See Also

`Stateflow.AtomicBox` | `Stateflow.AtomicSubchart` | `Stateflow.Box` | `Stateflow.Chart` | `Stateflow.Function` | `Stateflow.SimulinkBasedState` | `Stateflow.State`

Topics

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

“Combine Transitions and Junctions to Create Branching Paths”

“Resume Prior Substate Activity by Using History Junctions”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.JunctionPosition

Position and size of junctions

Description

Use a `Stateflow.JunctionPosition` object to control the position and size of a connective or history junction.

Creation

Each junction has its own `Stateflow.JunctionPosition` object. To access the `Stateflow.JunctionPosition` object, use the `Position` property for the `Stateflow.Junction` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Center — Position of center of junction

[7 7] (default) | [x y]

Position of the center of the junction, specified as a two-element numeric vector [x y] of coordinates relative to the upper left corner of the chart.

Radius — Radius of junction

7 (default) | scalar

Radius of the junction, specified as a scalar.

Examples

Change Size of Entry Junction

Set the radius of the connective junction `junction` to 10.

```
junction.Position.Radius = 10;
```

Version History

Introduced before R2006a

See Also

`Stateflow.Junction`

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.Machine

Container for Stateflow blocks in a Simulink model

Description

From a Stateflow perspective, `Stateflow.Machine` objects are equivalent to Simulink models. A `Stateflow.Machine` object contains `Stateflow.Chart`, `Stateflow.StateTransitionTableChart`, `Stateflow.TruthTableChart`, and `Stateflow.EMChart` objects that represent the Stateflow charts, State Transition Table blocks, Truth Table blocks, and MATLAB Function blocks in a Simulink model. For more information, see “Overview of the Stateflow API” on page 1-2.

Creation

You automatically create a `Stateflow.Machine` object when you load a model that contains a Stateflow block or call the function `sfnew`. To access the `Stateflow.Machine` object, call the `find` function for the `Simulink.Root` object. For example, if your Simulink model is named `myModel`, enter:

```
machine = find(sfroot, "-isa", "Stateflow.Machine", Name="myModel");
```

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Name — Name of Simulink model

character vector

This property is read-only.

Name of the Simulink model for the machine, specified as a character vector.

FullName — Full file path of Simulink model

character vector

This property is read-only.

Full file path of the Simulink model for the machine, specified as a character vector.

IsLibrary — Whether model builds library

false or 0 (default) | true or 1

This property is read-only.

Whether the Simulink model for the machine builds a library and not an application, specified as a numeric or logical 1 (`true`) or 0 (`false`).

Debugging

Debug — Debugger properties

`Stateflow.MachineDebug` object

Debugger properties for charts in the machine, specified as a `Stateflow.MachineDebug` object with these properties:

- **Animation.Enabled** — Whether to animate the charts in the machine during simulation, specified as a numeric or logical 1 (`true`) or 0 (`false`). Disabling this property is equivalent to selecting **None** in the **Animation Speed** drop-down list in the **Debug** tab.
- **Animation.Delay** — Delay that the chart animation uses for highlighting each transition segment in the machine, specified as a scalar. These values correspond to the settings of the **Animation Speed** drop-down list in the **Debug** tab:

Delay Value	Animation Speed
0.5	Slow
0.2	Medium
0	Fast
-1	Lightning Fast

- **Animation.MaintainHighlighting** — Whether to maintain the highlighting of active states in the machine after the simulation ends, specified as a numeric or logical 1 (`true`) or 0 (`false`).

Example: `machine.Debug.Animation.Enabled = true;`

Example: `machine.Debug.Animation.Delay = -1;`

Hierarchy

Path — Location of machine in model hierarchy

character vector

This property is read-only.

Location of the machine in the model hierarchy, specified as a character vector.

Dirty — Whether model has changed

`true` or 1 | `false` or 0

Whether the Simulink model for the machine has changed after being opened or saved, specified as a numeric or logical 1 (`true`) or 0 (`false`).

Locked — Whether machine is locked

`false` or 0 (default) | `true` or 1

Whether the machine is locked, specified as a numeric or logical 1 (`true`) or 0 (`false`). Enable this property to prevent changes in the Stateflow charts, state transition tables, and truth table blocks in this machine.

Iced — Whether machine is locked`false` or 0 (default) | `true` or 1

This property is read-only.

Whether the machine is locked, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property is equivalent to the property `Locked`, but is used internally to prevent changes in the machine during simulation.

Identification**Created — Date of creation**`character vector`

This property is read-only.

Date of the creation of the machine, specified as a character vector.

Creator — Creator`"Unknown"` (default) | string scalar | character vector

Creator of the machine, specified as a string scalar or character vector.

Modified — Record of modifications`""` (default) | string scalar | character vector

Record of modifications to the machine, specified as a string scalar or character vector.

Version — Version`"none"` (default) | string scalar | character vector

Version of the machine, specified as a string scalar or character vector.

Description — Description`""` (default) | string scalar | character vector

Description for the machine, specified as a string scalar or character vector.

Document — Document link`""` (default) | string scalar | character vector

Document link for the machine, specified as a string scalar or character vector.

Tag — User-defined tag`[]` (default) | any data type

User-defined tag for the machine, specified as data of any type.

Id — Unique identifier`scalar`

This property is read-only.

Unique identifier, specified as an integer scalar. Use this property to distinguish the machine from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

`find` Identify specified objects in hierarchy
`dialog` Open properties dialog box

Examples

Update Machine Version

Update the Modified and Version properties of machine machine.

```
machine.Modified = string(datetime);  
oldVersion = str2double(machine.Version);  
if isnan(oldVersion)  
    machine.Version = "1";  
else  
    machine.Version = string(oldVersion+1);  
end
```

Version History

Introduced before R2006a

See Also

Functions

`sfroot`

Objects

`Stateflow.Chart` | `Stateflow.StateTransitionTableChart` |
`Stateflow.TruthTableChart` | `Stateflow.EMChart`

Topics

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

“Machine Properties”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.MachineAnimation

Animation properties for charts in Stateflow machine

Description

Use a `Stateflow.MachineAnimation` object to specify the animation properties for the charts in a Stateflow machine.

Creation

Each Stateflow machine has its own `Stateflow.MachineAnimation` object. To access the `Stateflow.MachineAnimation` object, use the `Debug.Animation` property for the `Stateflow.Machine` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Enabled — Whether to animate charts during simulation

`true` or 1 (default) | `false` or 0

Whether to animate the charts in the machine during simulation, specified as a numeric or logical 1 (`true`) or 0 (`false`). Disabling this property is equivalent to selecting **None** in the **Animation Speed** drop-down list in the **Debug** tab.

Delay — Delay for highlighting transitions

0 (default) | scalar

Delay that the chart animation uses for highlighting each transition segment in the machine, specified as a scalar. These values correspond to the settings of the **Animation Speed** drop-down list in the **Debug** tab:

Delay Value	Animation Speed
0.5	Slow
0.2	Medium
0	Fast
-1	Lightning Fast

This property applies only when the `Enable` property of the machine is `true`.

MaintainHighlighting — Whether to maintain highlighting of active states

`false` or 0 (default) | `true` or 1

Whether to maintain the highlighting of active states in the machine after the simulation ends, specified as a numeric or logical 1 (`true`) or 0 (`false`).

Examples

Set Animation Speed to Lightning Fast

Access the `Stateflow.MachineDebug` and `Stateflow.MachineAnimation` objects for the `Stateflow.Machine` object `machine`.

```
debug = machine.Debug;  
animation = debug.Animation;
```

Enable animation and set delay to -1.

```
animation.Enabled = true;  
animation.Delay = -1;
```

Version History

Introduced before R2006a

See Also

`Stateflow.Machine`

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.MachineDebug

Debugger properties for charts in Stateflow machine

Description

Use a `Stateflow.MachineDebug` object to specify the debugger properties for the charts in a Stateflow machine.

Creation

Each Stateflow machine has its own `Stateflow.MachineDebug` object. To access the `Stateflow.MachineDebug` object, use the `Debug` property for the `Stateflow.Machine` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Animation — Animation properties

`Stateflow.MachineAnimation` object

Animation properties for charts in the machine, specified as a `Stateflow.MachineAnimation` object with these properties:

- **Enabled** — Whether to animate the charts in the machine during simulation, specified as a numeric or logical 1 (`true`) or 0 (`false`). Disabling this property is equivalent to selecting **None** in the **Animation Speed** drop-down list in the **Debug** tab.
- **Delay** — Delay that the chart animation uses for highlighting each transition segment in the machine, specified as a scalar. These values correspond to the settings of the **Animation Speed** drop-down list in the **Debug** tab:

Delay Value	Animation Speed
0.5	Slow
0.2	Medium
0	Fast
-1	Lightning Fast

- **MaintainHighlighting** — Whether to maintain the highlighting of active states in the machine after the simulation ends, specified as a numeric or logical 1 (`true`) or 0 (`false`).

Examples

Set Animation Speed to Lightning Fast

Access the `Stateflow.MachineDebug` and `Stateflow.MachineAnimation` objects for the `Stateflow.Machine` object `machine`.

```
debug = machine.Debug;  
animation = debug.Animation;
```

Enable animation and set delay to -1.

```
animation.Enabled = true;  
animation.Delay = -1;
```

Version History

Introduced before R2006a

See Also

`Stateflow.Machine`

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.Message

Message in chart, state, or box

Description

Use `Stateflow.Message` objects to communicate data locally or between Stateflow charts in Simulink models. For more information, see “Communicate with Stateflow Charts by Sending Messages”.

Creation

Syntax

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

Description

`message = Stateflow.Message(parent)` creates a `Stateflow.Message` object in a parent chart, state, or box.

Input Arguments

parent — Parent for new message

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object

Parent for the new message, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.State`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Interface

Name — Name of message

"message" (default) | string scalar | character vector

Name of the message, specified as a string scalar or character vector.

Scope — Scope of message

"Output" (default) | "Input" | "Local"

Scope of the message, specified as "Local", "Input", or "Output". For more information, see "Scope".

Port — Port index for message

scalar

Port index for the message, specified as an integer scalar. This property applies only to input and output messages. For more information, see "Port".

InitializeMethod — Method for initializing message data

"Expression" (default) | "Parameter" | "Not Needed"

Method for initializing the value of the message data, specified as a string scalar or character vector that depends on the scope of the message:

- For local and output messages, use "Expression" or "Parameter".
- For input messages, use "Not Needed".

To specify the initial value of the message data, use the `Props.InitialValue` property.

For more information, see "Initial Value".

Priority — Priority

"300" (default) | string scalar | character vector

Priority for the message, specified as a string scalar or character vector. If two distinct messages occur at the same time, this property determines which message is processed first. A smaller numeric value indicates a higher priority. This property applies only to local and output messages in discrete-event charts. For more information, see "Create Custom Queuing Systems Using Discrete-Event Stateflow Charts" (SimEvents).

Queue

UseInternalQueue — Whether chart maintains internal queue for message

true or 1 (default) | false or 0

Whether the Stateflow chart maintains an internal receiving queue for the input message, specified as a numeric or logical 1 (true) or 0 (false). This property applies only to input messages. For more information, see "Use Internal Queue".

QueueType — Order in which messages are removed from queue

"FIFO" (default) | "LIFO" | "Priority"

Order in which messages are removed from the receiving queue, specified as one of these values:

- "FIFO" — First in, first out.
- "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 local messages and to input messages that have `UseInternalQueue` set to true. For more information, see "Queue Type".

MessagePriorityOrder — Type of priority queue

"Ascending" (default) | "Descending"

Type of priority queue for the message, specified as one of these values:

- "Ascending" — Messages are received in ascending order of the message data value.
- "Descending" — Messages are received in descending order of the message data value.

This property applies only when the `QueueType` property of the message is "Priority". For more information, see "Queue Type".

QueueCapacity — Length of internal queue

10 (default) | scalar

Length of the internal queue for the message, specified as an integer scalar. This property applies only to local messages and to input messages that have `UseInternalQueue` set to `true`. For more information, see "Queue Capacity".

QueueOverflowDiagnostic — Level of diagnostic when number of messages exceeds queue capacity

"Error" (default) | "Warning" | "None"

Level of diagnostic action when the number of incoming messages exceeds the queue capacity for the message, specified as "Error", "Warning", or "None". This property applies only to local messages and to input messages that have `UseInternalQueue` set to `true`. For more information, see "Queue Overflow Diagnostic".

Data Specification

DataType — Data type of message

"Inherit: Same as Simulink" (default) | "double" | "single" | "int32" | "uint32" | "boolean" | ...

Data type of the message, specified as a string scalar or character vector that depends on the `Props.Type.Method` property of the message:

- If the `Props.Type.Method` property of the message is "Inherit", the value of this property is "Inherit: Same as Simulink".
- If the `Props.Type.Method` property of the message is "Built-in", you can specify this property with one of these options:
 - "double"
 - "single"
 - "int8"
 - "int16"
 - "int32"
 - "int64"
 - "uint8"
 - "uint16"
 - "uint32"
 - "uint64"
 - "boolean"
 - "string"

- "ml" (Supported only in charts that use C as the action language)
- Otherwise, the `Props.Type` properties of the message determine the value of this property.

For more information, see the section Add Data on page 1-0 in “Create Charts by Using the Stateflow API” on page 1-19.

Props — Data specification properties

`Stateflow.DataProps` object

Data specification properties, specified as a `Stateflow.DataProps` object with these properties:

- **Type.Method** — Method for setting the data type of the message, specified as "Inherited", "Built-in", "Bus Object", "Enumerated", "Expression", or "Fixed point". This property is equivalent to the **Mode** field of the Data Type Assistant in the Model Explorer and the Data properties dialog box. For more information, see “Specify Type of Stateflow Data”.
- **Type.BusObject** — Name of the Simulink.Bus object that defines the message data, specified as a string scalar or character vector. This property applies only when the `Type.Method` property of the data object is "Bus Object". For more information, see “Access Bus Signals Through Stateflow Structures”.
- **Type.EnumType** — Name of the enumerated type that defines the message data, specified as a string scalar or character vector. This property applies only when the `Type.Method` property of the data object is "Enumerated". For more information, see “Reference Values by Name by Using Enumerated Data”.
- **Type.Expression** — Expression that evaluates to the data type of the message data, specified as a string scalar or character vector. This property applies only when the `Type.Method` property of the data object is "Expression". For more information, see “Specify Data Properties by Using MATLAB Expressions”.
- **Type.Signed** — Signedness, specified as a numeric or logical 1 (true) or 0 (false). This property applies only when the `Type.Method` property of the data object is "Fixed point". For more information, see “Fixed-Point Data in Stateflow Charts”.
- **Type.WordLength** — Word length, in bits, specified as a string scalar or character vector. This property applies only when the `Type.Method` property of the data object is "Fixed point". For more information, see “Fixed-Point Data in Stateflow Charts”.
- **Type.Fixpt.ScalingMode** — Method for scaling the fixed-point message data, specified as "Binary point", "Slope and bias", or "None". This property applies only when the `Type.Method` property of the data object is "Fixed point". For more information, see “Fixed-Point Data in Stateflow Charts”.
- **Type.Fixpt.FractionLength** — Fraction length, in bits, specified as a string scalar or character vector. This property applies only when the `Type.Method` property is "Fixed point" and the `Type.Fixpt.ScalingMode` property is "Binary point".
- **Type.Fixpt.Slope** — Slope, specified as a string scalar or character vector. This property applies only when the `Type.Method` property is "Fixed point" and the `Type.Fixpt.ScalingMode` property is "Slope and bias".
- **Type.Fixpt.Bias** — Bias, specified as a string scalar or character vector. This property applies only to when the `Type.Method` property is "Fixed point" and the `Type.Fixpt.ScalingMode` property is "Slope and bias".
- **Type.Fixpt.Lock** — Whether to prevent replacement of the fixed-point type with an autoscaled type chosen by the Fixed-Point Tool (Fixed-Point Designer), specified as a numeric or logical 1 (true) or 0 (false). This property applies only when the `Type.Method` property of the data object is "Fixed point".

- **Array.Size** — Size of the message data, specified as a string scalar or character vector. For more information, see “Specify Size of Stateflow Data”.
- **Complexity** — Whether the message accepts complex values, specified as "On" or "Off". For more information, see “Complex Data in Stateflow Charts”.
- **InitialValue** — Initial value, specified as a string scalar or character vector.

CompiledSize — Message data size as determined by compiler

' ' (default) | character vector

This property is read-only.

Message data size as determined by the compiler, specified as a character vector.

CompiledType — Data type as determined by compiler

' unknown ' (default) | character vector

This property is read-only.

Data type as determined by the compiler, specified as a character vector.

Hierarchy**Machine — Machine that contains message**

Stateflow.Machine object

This property is read-only.

Machine that contains the message, specified as a Stateflow.Machine object.

Path — Location of parent in model hierarchy

character vector

This property is read-only.

Location of the parent of the message in the model hierarchy, specified as a character vector.

Identification**Description — Description**

"" (default) | string scalar | character vector

Description for the message, specified as a string scalar or character vector.

Document — Document link

"" (default) | string scalar | character vector

Document link for the message, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the message, specified as data of any type.

SSIdNumber — Session-independent identifier

scalar

This property is read-only.

Session-independent identifier, specified as an integer scalar. Use this property to distinguish the message from other objects in the model.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Unlike `SSIdNumber`, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>getParent</code>	Identify parent of object
<code>getReferences</code>	Identify references to symbol name
<code>renameReferences</code>	Rename symbol and update references to symbol name
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment

Examples

Add Message to Chart

Add a message to the chart `ch`. Specify its name, scope, and data type.

```
message = Stateflow.Message(ch);
message.Name = "M";
message.Scope = "Input";
message.Props.Type.Method = "Built-in";
message.DataType = "int32";
```

Version History

Introduced in R2015b

R2023a: New object functions

The object functions `getReferences` and `renameReferences` find and update the locations, such as state and transition actions, where the chart refers to the name of a message:

- The object function `getReferences` returns the locations where a chart refers to a message name.
- The object function `renameReferences` renames a message and updates all references to the message name in the chart.

See Also

`Stateflow.Box` | `Stateflow.Chart` | `Stateflow.State`

Topics

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

“Communicate with Stateflow Charts by Sending Messages”

“Set Properties for a Message”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.NoteFont

Font properties for annotations

Description

Use a `Stateflow.NoteFont` object to specify the font properties for an annotation.

Creation

Each annotation has its own `Stateflow.NoteFont` object. To access the `Stateflow.NoteFont` object, use the `Font` property for the `Stateflow.Annotation` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Name — Font name

'Helvetica' (default) | character vector

This property is read-only.

Font name, specified as a character vector. The `StateFont.Name` property of the chart that contains the annotation sets the value of this property.

Angle — Font angle

"NORMAL" (default) | "ITALIC"

Font angle, specified as "NORMAL" or "ITALIC".

Weight — Font weight

"NORMAL" (default) | "BOLD"

Font weight, specified as "NORMAL" or "BOLD".

Size — Font size

scalar

Font size, specified as a scalar. The `StateFont.Size` property of the chart that contains the annotation sets the default value of this property.

Examples

Change Font Properties for Annotation

Access the `Stateflow.NoteFont` object for the `Stateflow.Annotation` object `annotation`.

```
font = annotation.Font;
```

Set the font angle to italics, the font weight to bold, and the font size to 8.

```
font.Angle = "ITALIC";  
font.Weight = "BOLD";  
font.Size = 8;
```

Version History

Introduced before R2006a

See Also

`Stateflow.Annotation`

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.Port

Entry or exit port in state or atomic subchart

Description

Use `Stateflow.Port` objects to create ports and junctions that provide entry and exit connections across boundaries in the Stateflow hierarchy. Entry and exit ports improve componentization by isolating the transition logic for entering and exiting states. Unlike supertransitions, they can be used in atomic subcharts. For more information, see “Create Entry and Exit Connections Across State Boundaries”.

Entry and exit ports are located on the boundary of a state or atomic subchart. Each port has a matching junction that marks the entry or exit point inside the state or atomic subchart. The port and junction are represented by separate `Stateflow.Port` objects.

Creation

Syntax

```
port = Stateflow.Port(parent,portType)
```

Description

`port = Stateflow.Port(parent,portType)` creates a `Stateflow.Port` object of the specified port type in the parent. The function creates a second `Stateflow.Port` object for the matching entry or exit port on the boundary of the parent. To identify the matching `Stateflow.Port` object, use the function `Stateflow.findMatchingPort`.

Input Arguments

parent — Parent for new entry or exit junction

`Stateflow.State` object | `Stateflow.Chart` object

Parent for the new entry or exit junction, specified as a `Stateflow.State` object or as a `Stateflow.Chart` object that is the subchart of a `Stateflow.AtomicSubchart` object.

portType — Type of junction

"EntryJunction" | "ExitJunction"

Type of junction, specified as "EntryJunction" or "ExitJunction".

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

LabelString — Label for port or junction

"" (default) | string scalar | character vector

Label for the port or junction, specified as a string scalar or character vector. Changing this property automatically sets the `LabelString` property for the matching `Stateflow.Port` object to the same value.

PortType — Type of port or junction

'EntryJunction' | 'EntryPort' | 'ExitJunction' | 'ExitPort'

This property is read-only.

Type of port or junction, specified as one of these values:

- 'EntryJunction' — Entry junction inside a state or atomic subchart
- 'EntryPort' — Entry port on the boundary of a state or atomic subchart
- 'ExitJunction' — Exit junction inside a state or atomic subchart
- 'ExitPort' — Exit port on the boundary of a state or atomic subchart

IsExplicitlyCommented — Whether to comment out junction and port pair

false or 0 (default) | true or 1

Whether to comment out the junction and port pair, specified as a numeric or logical 1 (`true`) or 0 (`false`). Setting this property to `true` is equivalent to right-clicking the entry or exit junction and selecting **Comment Out**. This property applies only to entry and exit junctions. Attempting to set this property on an entry or exit port results in an error. For more information, see “Comment Out Objects in a Stateflow Chart”.

IsImplicitlyCommented — Whether port or junction is implicitly commented out

true or 1 | false or 0

This property is read-only.

Whether the port or junction is implicitly commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). The port or junction is implicitly commented out when you explicitly comment out an object that contains it. Additionally, an entry or exit port is implicitly commented out when you explicitly comment out the matching entry or exit junction. If the port or junction is contained in an atomic subchart or an atomic box, this property is `false` unless the explicitly commented object is also contained in the atomic subchart or atomic box.

IsCommented — Whether port or junction is commented out


true or 1 | false or 0

This property is read-only.

Whether the port or junction is commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property is `true` when either `IsExplicitlyCommented` or `IsImplicitlyCommented` is `true`.

CommentText — Comment text

"" (default) | string scalar | character vector

Comment text added to the entry or exit junction, specified as a string scalar or character vector. This property applies only to entry and exit junctions when the `IsExplicitlyCommented` property is `true`. In the Stateflow Editor, when you point to the comment badge  on the junction, the text appears as a tooltip. When you set the `IsExplicitlyCommented` property to `false`, the value of `CommentText` reverts to `""`.

Graphical Appearance

Position — Position and size of port or junction

`Stateflow.PortPosition` object

Position and size of the port or junction, specified as a `Stateflow.PortPosition` object with these properties:

- **Center** — Position of the center of the port or junction, specified as a two-element numeric vector `[x y]` of coordinates relative to the upper left corner of the chart.
- **Radius** — Radius of the port or junction, specified as a scalar.

Example: `port.Position.Center = [31.41 27.18];`

Example: `port.Position.Radius = 16.18;`

LabelPosition — Position and size of port or junction label

`[-10 -15 2 16]` (default) | `[left top width height]`

Position and size of the port or junction label, specified as a four-element numeric vector of the form `[left top width height]`.

ArrowSize — Size of incoming transition arrows

8 (default) | scalar

Size of incoming transition arrows, specified as a scalar.

Hierarchy

Chart — Chart that contains port or junction

`Stateflow.Chart` object

This property is read-only.

Chart that contains the port or junction, specified as a `Stateflow.Chart` object.

Subviewer — Subviewer for port or junction

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object

This property is read-only.

Subviewer for the port or junction, specified as a `Stateflow.Chart`, `Stateflow.State`, or `Stateflow.Box` object. The subviewer is the chart or subchart where you can graphically view the port.

Home — Home state or subchart

`Stateflow.State` object | `Stateflow.AtomicSubchart` object

This property is read-only.

Home state or subchart, specified as a `Stateflow.State` or `Stateflow.AtomicSubchart` object. The home of an entry or exit port is the state or subchart whose boundary contains the port. This property applies only to entry and exit ports.

Machine — Machine that contains port or junction

`Stateflow.Machine` object

This property is read-only.

Machine that contains the port or junction, specified as a `Stateflow.Machine` object.

Path — Location of parent in model hierarchy

character vector

This property is read-only.

Location of the parent of the port or junction in the model hierarchy, specified as a character vector.

Linked — Whether port or junction has matching junction or port

`true` or `1` (default) | `false` or `0`

This property is read-only.

Whether the port or junction has a matching junction or port, specified as a numeric or logical `1` (`true`) or `0` (`false`). This property is used to detect internal inconsistencies in the chart.

Identification**Description — Description**

`""` (default) | string scalar | character vector

Description for the port or junction, specified as a string scalar or character vector.

Document — Document link

`""` (default) | string scalar | character vector

Document link for the port or junction, specified as a string scalar or character vector.

Tag — User-defined tag

`[]` (default) | any data type

User-defined tag for the port or junction, specified as data of any type.

SSIdNumber — Session-independent identifier

scalar

This property is read-only.

Session-independent identifier, specified as an integer scalar. Use this property to distinguish the port or junction from other objects in the model.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Unlike `SSIdNumber`, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>getParent</code>	Identify parent of object
<code>sinkedTransitions</code>	Identify transitions with specified destination
<code>sourcedTransitions</code>	Identify transitions with specified source
<code>commentedBy</code>	Identify objects that implicitly comment out a graphical object
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>highlight</code>	Highlight graphical object
<code>fitToView</code>	Zoom in on graphical object

Examples

Add Exit Port and Junction to Atomic Subchart

In an atomic subchart called A, add an exit port and an exit junction with the label "exit".

Find the `Stateflow.AtomicSubchart` object that corresponds to the atomic subchart A in the chart `ch`.

```
atomicSubchart = find(ch, "-isa", "Stateflow.AtomicSubchart", Name="A");
```

Add an exit junction to the atomic subchart. Use the `Subchart` property of the atomic subchart as the parent of the exit junction. Display the value of the `PortType` property of the exit junction.

```
exitJunction = Stateflow.Port(atomicSubchart.Subchart, "ExitJunction");
exitJunction.PortType
```

```
ans =
```

```
    'ExitJunction'
```

Set the label of the exit junction to "exit".

```
exitJunction.labelString = "exit";
```

Find the `Stateflow.Port` object for the matching exit port. Display the value of the `PortType` property of the exit port.

```
exitPort = Stateflow.findMatchingPort(exitJunction);
exitPort.PortType
```

```
ans =
```

```
    'ExitPort'
```

Display the label of the exit port.

```
exitPort.labelString
```

```
ans =  
    'exit'
```

Tips

- If you move an entry or exit junction to a different parent, Stateflow deletes the `Stateflow.Port` object for the matching port and creates a `Stateflow.Port` object on the new parent. To identify the new matching port, use the `Stateflow.findMatchingPort` function.

Version History

Introduced in R2021b

R2023a: New object function and property

Errors starting in R2023a

Use the object function `commentedBy` and the property `IsCommented` to investigate commented-out ports and junctions:

- The object function `commentedBy` identifies the explicitly commented objects that cause a port or junction to be commented out.
- The property `IsCommented` indicates whether a port or junction is commented out. This property replaces the object function `isCommented`.

See Also

Functions

`find` | `Stateflow.findMatchingPort`

Objects

`Stateflow.AtomicSubchart` | `Stateflow.Chart` | `Stateflow.State`

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Create Entry and Exit Connections Across State Boundaries”

Stateflow.PortPosition

Position and size of entry or exit ports

Description

Use a `Stateflow.PortPosition` object to control the position and size of an entry or exit port.

Creation

Each entry or exit port has its own `Stateflow.PortPosition` object. To access the `Stateflow.PortPosition` object, use the `Position` property for the `Stateflow.Port` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Center — Position of center of port or junction

[x y]

Position of the center of the port or junction, specified as a two-element numeric vector [x y] of coordinates relative to the upper left corner of the chart.

Radius — Radius of port or junction

scalar

Radius of the port or junction, specified as a scalar.

Examples

Change Size of Entry Junction

Set the radius of the entry junction `entryJunction` to 10.

```
entryJunction.Position.Radius = 10;
```

Version History

Introduced in R2021b

See Also

`Stateflow.Port`

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.SigLoggingInfo

Signal logging properties for states and data

Description

Use a `Stateflow.SigLoggingInfo` object to specify the signal logging properties for a state or data object. Signal logging saves the self activity of a state or the values of a data object to the MATLAB workspace during simulation. For more information, see “Log Simulation Output for States and Data”.

Creation

Each state, atomic subchart, Simulink based state, and data object has its own `Stateflow.SigLoggingInfo` object. To access the `Stateflow.SigLoggingInfo` object, use the `LoggingInfo` property for the `Stateflow.State`, `Stateflow.AtomicSubchart`, `Stateflow.SimulinkBasedState`, or `Stateflow.Data` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

DataLogging — Whether to enable signal logging

`false` or 0 (default) | `true` or 1

Whether to enable signal logging for the state or data object, specified as a numeric or logical 1 (`true`) or 0 (`false`).

DecimateData — Whether to limit logged data

`false` or 0 (default) | `true` or 1

Whether to limit the amount of logged data, specified as a numeric or logical 1 (`true`) or 0 (`false`). When this property is `true`, signal logging skips samples by using the interval size specified by the `Decimation` property. For more information, see “Decimation”.

Decimation — Decimation interval

2 (default) | scalar

Decimation interval, specified as an integer scalar. This property applies only when the `DecimateData` property is `true`. The default value of 2 means that the chart logs every other sample.

LimitDataPoints — Whether to limit number of data points to log

`false` or 0 (default) | `true` or 1

Whether to limit the number of data points to log, specified as a numeric or logical 1 (`true`) or 0 (`false`). When this property is `true`, signal logging limits the number of data points by using the value specified by the `MaxPoints` property. For more information, see “Limit data points to last”.

MaxPoints — Maximum number of data points to log

5000 (default) | scalar

Maximum number of data points to log, specified as an integer scalar. This property applies only when the `LimitDataPoints` property is `true`. The default value of 5000 means the chart logs only the last 5000 data points generated by the simulation.

NameMode — Source of signal name

"SignalName" (default) | "Custom"

Source of the signal name used to log the state or data object, specified as one of these values:

- "SignalName" — Use the name of the state or data object.
- "Custom" — Use the custom signal name specified by the `LoggingName` property.

For more information, see “Logging name”.

LoggingName — Custom signal name

string scalar | character vector

Custom signal name for the state or data object, specified as a string scalar or character vector. This property applies only when the `NameMode` property is "Custom".

Examples

Enable Signal Logging for Data

Access the `SigLoggingInfo` object for the `Stateflow.Data` object `x`.

```
log = x.LoggingInfo;
```

Enable logging for the data object and specify a custom signal name.

```
log.DataLogging = true;  
log.NameMode = "Custom";  
log.LoggingName = "My Data";
```

Version History

Introduced before R2006a

See Also

`Stateflow.Data` | `Stateflow.State` | `Stateflow.AtomicSubchart` | `Stateflow.SimulinkBasedState`

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Log Simulation Output for States and Data”

Stateflow.SimulinkBasedState

Simulink based state in chart, state, or box

Description

Use `Stateflow.SimulinkBasedState` objects to create Simulink subsystems within a Stateflow state. With Simulink based states, you can model hybrid dynamic systems or systems that switch between periodic and continuous time dynamics. For more information, see “Simulink Subsystems as States”.

Creation

Syntax

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

Description

`simulinkBasedState = Stateflow.SimulinkBasedState(parent)` creates a `Stateflow.SimulinkBasedState` object in a parent chart, state, or box.

Input Arguments

parent — Parent for new Simulink based state

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object

Parent for the new Simulink based state, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.State`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Name — Name of Simulink based state

`""` (default) | string scalar | character vector

Name of the Simulink based state, specified as a string scalar or character vector.

IsExplicitlyCommented — Whether to comment out Simulink based state

`false` or `0` (default) | `true` or `1`

Whether to comment out the Simulink based state, specified as a numeric or logical 1 (`true`) or 0 (`false`). Setting this property to `true` is equivalent to right-clicking the Simulink based state and selecting **Comment Out**. For more information, see “Comment Out Objects in a Stateflow Chart”.

IsImplicitlyCommented — Whether Simulink based state is implicitly commented out

`true` or 1 | `false` or 0

This property is read-only.

Whether the Simulink based state is implicitly commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). The Simulink based state is implicitly commented out when you explicitly comment out an object that contains it. If the Simulink based state is contained in an atomic subchart, this property is `false` unless the explicitly commented object is also contained in the atomic subchart.

IsCommented — Whether Simulink based state is commented out


`true` or 1 | `false` or 0

This property is read-only.

Whether the Simulink based state is commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property is `true` when either `IsExplicitlyCommented` or `IsImplicitlyCommented` is `true`.

CommentText — Comment text

`""` (default) | string scalar | character vector

Comment text added to the Simulink based state, specified as a string scalar or character vector. This property applies only when the `IsExplicitlyCommented` property is `true`. In the Stateflow Editor, when you point to the comment badge  on the Simulink based state, the text appears as a tooltip. When you set the `IsExplicitlyCommented` property to `false`, the value of `CommentText` reverts to `""`.

Graphical Appearance

Position — Position and size of Simulink based state

`[0 0 90 60]` (default) | `[left top width height]`

Position and size of the Simulink based state, specified as a four-element numeric vector of the form `[left top width height]`.

BadIntersection — Whether Simulink based state intersects a box, state, or function

`true` or 1 | `false` or 0

This property is read-only.

Whether the Simulink based state graphically intersects a box, state, or function, specified as a numeric or logical 1 (`true`) or 0 (`false`).

ContentPreviewEnabled — Whether to display preview of Simulink based state contents

`true` or 1 (default) | `false` or 0

Whether to display a preview of the Simulink based state contents, specified as a numeric or logical 1 (`true`) or 0 (`false`).

ArrowSize — Size of incoming transition arrows

8 (default) | scalar

Size of incoming transition arrows, specified as a scalar.

FontSize — Font size for Simulink based state label

scalar

Font size for the Simulink based state label, specified as a scalar. The `StateFont.Size` property of the chart that contains the Simulink based state sets the initial value of this property.

State Decomposition**Type — Decomposition of sibling states**

'AND' | 'OR'

This property is read-only.

Decomposition of sibling states, specified as 'AND' or 'OR'. The Simulink based state inherits this property from the `Decomposition` property of its parent state or chart.

ExecutionOrder — Execution order in parallel (AND) decomposition

scalar

Execution order for the Simulink based state in parallel (AND) decomposition, specified as an integer scalar. This property applies only when both of these conditions are satisfied:

- The `Type` property of the Simulink based state is "AND".
- The `UserSpecifiedStateTransitionExecutionOrder` property of the chart that contains the Simulink based state is `true`.

Active State Output**HasOutputData — Whether to create active state data output**

false or 0 (default) | true or 1

Whether to create an active state data output port for the Simulink based state, specified as a numeric or logical 1 (`true`) or 0 (`false`). For more information, see “Monitor State Activity Through Active State Data”.

OutputData — Active state data object

Stateflow.Data object

This property is read-only.

Active state data object for the Simulink based state, specified as a `Stateflow.Data` object. This property applies only when the `HasOutputData` property for the Simulink based state is `true`.

OutputPortName — Name of active state data object

string scalar | character vector

Name of the active state data object for the Simulink based state, specified as a string scalar or character vector. This property applies only when the `HasOutputData` property for the Simulink based state is `true`.

OutputMonitoringMode — Monitoring mode for active state output

"SelfActivity"

Monitoring mode for the active state output data, specified as a string scalar or character vector. For Simulink based states, the only option is "SelfActivity".

Signal Logging and Test Point Monitoring**LoggingInfo — Signal logging properties**

Stateflow.SigLoggingInfo object

Signal logging properties for the Simulink based state, specified as a Stateflow.SigLoggingInfo object with these properties:

- **DataLogging** — Whether to enable signal logging, specified as a numeric or logical 1 (true) or 0 (false).
- **DecimateData** — Whether to limit the amount of logged data, specified as a numeric or logical 1 (true) or 0 (false).
- **Decimation** — Decimation interval, specified as an integer scalar. This property applies only when the DecimateData property is true.
- **LimitDataPoints** — Whether to limit the number of data points to log, specified as a numeric or logical 1 (true) or 0 (false).
- **MaxPoints** — Maximum number of data points to log, specified as an integer scalar. This property applies only when the LimitDataPoints property is true.
- **NameMode** — Source of the signal name, specified as "SignalName" or "Custom".
- **LoggingName** — Custom signal name, specified as a string scalar or character vector. This property applies only when the NameMode property is "Custom".

Signal logging saves the self activity of the Simulink based state to the MATLAB workspace during simulation. For more information, see "Log Simulation Output for States and Data".

Example: `state.LoggingInfo.DataLogging = true;`

TestPoint — Whether to set Simulink based state as test point

false or 0 (default) | true or 1

Whether to set the Simulink based state as a test point, specified as a numeric or logical 1 (true) or 0 (false). You can monitor testpoints with a floating scope during simulation. You can also log test point values to the MATLAB workspace. For more information, see "Monitor Test Points in Stateflow Charts".

Debugging**Debug — Debugger properties**

Stateflow.StateDebug object

Debugger properties for the Simulink based state, specified as a Stateflow.StateDebug object with these properties:

- **OnEntry** — Whether to set the On State Entry breakpoint, specified as a numeric or logical 1 (true) or 0 (false).
- **OnDuring** — Whether to set the During State breakpoint, specified as a numeric or logical 1 (true) or 0 (false).

- **OnExit** — Whether to set the On State Exit breakpoint, specified as a numeric or logical 1 (true) or 0 (false).

For more information, see “Set Breakpoints to Debug Charts”.

Example: `simulinkBasedState.Debug.Breakpoints.OnEntry = true;`

Example: `simulinkBasedState.Debug.Breakpoints.OnDuring = true;`

Example: `simulinkBasedState.Debug.Breakpoints.OnExit = true;`

Hierarchy

Chart — Chart that contains Simulink based state

`Stateflow.Chart` object

This property is read-only.

Chart that contains the Simulink based state, specified as a `Stateflow.Chart` object.

Subviewer — Subviewer for Simulink based state

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object

This property is read-only.

Subviewer for the Simulink based state, specified as a `Stateflow.Chart`, `Stateflow.State`, or `Stateflow.Box` object. The subviewer is the chart or subchart where you can graphically view the Simulink based state.

Machine — Machine that contains Simulink based state

`Stateflow.Machine` object

This property is read-only.

Machine that contains the Simulink based state, specified as a `Stateflow.Machine` object.

Path — Location of parent in model hierarchy

character vector

This property is read-only.

Location of the parent of the Simulink based state in the model hierarchy, specified as a character vector.

Identification

Description — Description

"" (default) | string scalar | character vector

Description for the Simulink based state, specified as a string scalar or character vector.

Document — Document link

"" (default) | string scalar | character vector

Document link for the Simulink based state, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the Simulink based state, specified as data of any type.

SSIdNumber — Session-independent identifier

scalar

This property is read-only.

Session-independent identifier, specified as an integer scalar. Use this property to distinguish the Simulink based state from other objects in the model.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Unlike `SSIdNumber`, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>getParent</code>	Identify parent of object
<code>getReferences</code>	Identify references to symbol name
<code>renameReferences</code>	Rename symbol and update references to symbol name
<code>commentedBy</code>	Identify objects that implicitly comment out a graphical object
<code>getMappingForSymbol</code>	Get mapping for symbol in atomic subchart, atomic box, or Simulink based state
<code>setMappingForSymbol</code>	Set mapping for symbol in atomic subchart, atomic box, or Simulink based state
<code>clearMappingForSymbol</code>	Clear mapping for symbol in atomic subchart, atomic box, or Simulink based state
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>highlight</code>	Highlight graphical object
<code>fitToView</code>	Zoom in on graphical object

Examples

Add Simulink Based State to Chart

Add a Simulink based state in the chart `ch`. Set its name to `A`.

```
simulinkBasedState = Stateflow.SimulinkBasedState(ch);
simulinkBasedState.Name = "A";
```

Map Variables in Simulink Based State

In a Simulink based state called `Locked`, modify the mapping for the output `we`.

Open the model `sf_clutch.slx`.

```
open_system("sf_clutch.slx")
```

Access the `Stateflow.SimulinkBasedState` object for the Simulink based state `Locked`.

```
subsystem = find(sfroot, "-isa", "Stateflow.SimulinkBasedState", ...  
    Name="Locked");
```

Check the mapping for Simulink based state output `we`.

```
getMappingForSymbol(subsystem, "we").Name
```

```
ans =  
'we'
```

Map the Simulink based state output `we` to chart output `wv`.

```
setMappingForSymbol(subsystem, "we", "wv")  
getMappingForSymbol(subsystem, "we").Name
```

```
ans =  
'wv'
```

Clear the mapping for Simulink based state output `we`.

```
clearMappingForSymbol(subsystem, "we")  
getMappingForSymbol(subsystem, "we").Name
```

```
ans =  
'we'
```

Version History

Introduced in R2017b

R2023a: New object functions and properties

Errors starting in R2023a

`Stateflow.SimulinkBasedState` objects have new object functions and properties:

- The object function `setMappingForSymbol` maps a Simulink based state symbol to a main chart symbol.
- The object function `clearMappingForSymbol` clears the mapping for a Simulink based state symbol.
- The object function `getMappingForSymbol` returns the main chart symbol that a Simulink based state symbol maps to.
- The object function `getReferences` returns the locations where a chart refers to the name of a Simulink based state.
- The object function `renameReferences` renames a Simulink based state and updates all references to the name of the Simulink based state in the chart.
- The object function `commentedBy` identifies the explicitly commented objects that cause a Simulink based state to be commented out.
- The property `IsCommented` indicates whether a Simulink based state is commented out. This property replaces the object function `isCommented`.

See Also

[Stateflow.Box](#) | [Stateflow.Chart](#) | [Stateflow.State](#)

Topics

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

“Simulink Subsystems as States”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.SLFunction

Simulink function in chart, state, box, or function

Description

Use `Stateflow.SLFunction` objects to create Simulink functions that enable you to call Simulink subsystems in the actions of states and transitions. Typical applications include:

- Defining a function that requires Simulink blocks
- Scheduling execution of multiple controllers

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

Creation

Syntax

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

Description

`function = Stateflow.SLFunction(parent)` creates a `Stateflow.SLFunction` object in a parent chart, state, box, or function.

Input Arguments

parent — Parent for new Simulink function

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object

Parent for the new Simulink function, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.Function`
- `Stateflow.State`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Name — Name of Simulink function

"" (default) | string scalar | character vector

Name of the Simulink function, specified as a string scalar or character vector.

LabelString — Label for Simulink function

"?" (default) | string scalar | character vector

Label for the Simulink function, specified as a string scalar or character vector.

IsExplicitlyCommented — Whether to comment out Simulink function

false or 0 (default) | true or 1

Whether to comment out the Simulink function, specified as a numeric or logical 1 (**true**) or 0 (**false**). Setting this property to **true** is equivalent to right-clicking the Simulink function and selecting **Comment Out**. For more information, see “Comment Out Objects in a Stateflow Chart”.

IsImplicitlyCommented — Whether Simulink function is implicitly commented out

true or 1 | false or 0

This property is read-only.

Whether the Simulink function is implicitly commented out, specified as a numeric or logical 1 (**true**) or 0 (**false**). The Simulink function is implicitly commented out when you explicitly comment out an object that contains it. If the Simulink function is contained in an atomic subchart or an atomic box, this property is **false** unless the explicitly commented object is also contained in the atomic subchart or atomic box.

IsCommented — Whether Simulink function is commented out


true or 1 | false or 0

This property is read-only.

Whether the Simulink function is commented out, specified as a numeric or logical 1 (**true**) or 0 (**false**). This property is **true** when either **IsExplicitlyCommented** or **IsImplicitlyCommented** is **true**.

CommentText — Comment text

"" (default) | string scalar | character vector

Comment text added to the Simulink function, specified as a string scalar or character vector. This property applies only when the **IsExplicitlyCommented** property is **true**. In the Stateflow Editor, when you point to the comment badge  on the Simulink function, the text appears as a tooltip. When you set the **IsExplicitlyCommented** property to **false**, the value of **CommentText** reverts to "".

Graphical Appearance

Position — Position and size of Simulink function

[0 0 90 60] (default) | [left top width height]

Position and size of the Simulink function, specified as a four-element numeric vector of the form [left top width height].

BadIntersection — Whether Simulink function intersects a box, state, or function`true or 1 | false or 0`

This property is read-only.

Whether the Simulink function graphically intersects a box, state, or function, specified as a numeric or logical 1 (`true`) or 0 (`false`).

ContentPreviewEnabled — Whether to display preview of Simulink function contents`true or 1 (default) | false or 0`

Whether to display a preview of the Simulink function contents, specified as a numeric or logical 1 (`true`) or 0 (`false`).

FontSize — Font size for Simulink function label`scalar`

Font size for the Simulink function label, specified as a scalar. The `StateFont.Size` property of the chart that contains the Simulink function sets the initial value of this property.

Hierarchy**Chart — Chart that contains Simulink function**`Stateflow.Chart` object

This property is read-only.

Chart that contains the Simulink function, specified as a `Stateflow.Chart` object.

Subviewer — Subviewer for Simulink function`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object

This property is read-only.

Subviewer for the Simulink function, specified as a `Stateflow.Chart`, `Stateflow.State`, `Stateflow.Box`, or `Stateflow.Function` object. The subviewer is the chart or subchart where you can graphically view the Simulink function.

Machine — Machine that contains Simulink function`Stateflow.Machine` object

This property is read-only.

Machine that contains the Simulink function, specified as a `Stateflow.Machine` object.

Path — Location of parent in model hierarchy`character vector`

This property is read-only.

Location of the parent of the Simulink function in the model hierarchy, specified as a character vector.

Identification

Description — Description

"" (default) | string scalar | character vector

Description for the Simulink function, specified as a string scalar or character vector.

Document — Document link

"" (default) | string scalar | character vector

Document link for the Simulink function, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the Simulink function, specified as data of any type.

SSIdNumber — Session-independent identifier

scalar

This property is read-only.

Session-independent identifier, specified as an integer scalar. Use this property to distinguish the Simulink function from other objects in the model.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Unlike `SSIdNumber`, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>find</code>	Identify specified objects in hierarchy
<code>getChildren</code>	Identify children of object
<code>getParent</code>	Identify parent of object
<code>getReferences</code>	Identify references to symbol name
<code>renameReferences</code>	Rename symbol and update references to symbol name
<code>commentedBy</code>	Identify objects that implicitly comment out a graphical object
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>highlight</code>	Highlight graphical object
<code>fitToView</code>	Zoom in on graphical object

Examples

Add Simulink Function to Chart

Add a Simulink function in the chart `ch`. Set its label to "[y1,y2] = f(x1,x2,x3)".

```
function = Stateflow.SLFunction(ch);  
function.LabelString = "[y1,y2] = f(x1,x2,x3)";
```

Version History

Introduced in R2008b

R2023a: New object functions and properties

Errors starting in R2023a

Stateflow.SLFunction objects have new object functions and properties:

- The object function `getReferences` returns the locations where a chart refers to the name of a Simulink function.
- The object function `renameReferences` renames a Simulink function and updates all references to the function name in the chart.
- The object function `commentedBy` identifies the explicitly commented objects that cause a Simulink function to be commented out.
- The property `IsCommented` indicates whether a Simulink function is commented out. This property replaces the object function `isCommented`.

See Also

[Stateflow.Box](#) | [Stateflow.Chart](#) | [Stateflow.Function](#) | [Stateflow.State](#)

Topics

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

“Reuse Simulink Functions in Stateflow Charts”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.State

State in chart, state, or box

Description

Use `Stateflow.State` objects to describe an operating mode of a reactive system. For more information, see “Represent Operating Modes by Using States”.

Creation

Syntax

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

Description

`state = Stateflow.State(parent)` creates a `Stateflow.State` object in a parent chart, state, or box.

Input Arguments

parent — Parent for new state

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object

Parent for the new state, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.State`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Name — Name of state

`""` (default) | string scalar | character vector

Name of the state, specified as a string scalar or character vector.

LabelString — Label for state

`"?"` (default) | string scalar | character vector

Label for the state, specified as a string scalar or character vector. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16.

DuringAction — State during action

character vector

This property is read-only.

State during action, specified as a character vector. The value of this property depends on the `LabelString` property for the state. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16. This property is not supported in Moore charts.

EntryAction — State entry action

character vector

This property is read-only.

State entry action, specified as a character vector. The value of this property depends on the `LabelString` property for the state. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16. This property is not supported in Moore charts.

ExitAction — State exit action

character vector

This property is read-only.

State exit action, specified as a character vector. The value of this property depends on the `LabelString` property for the state. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16. This property is not supported in Moore charts.

MooreAction — State action in Moore chart

character vector

This property is read-only.

State action in a Moore chart, specified as a character vector. The value of this property depends on the `LabelString` property for the state. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16. This property is supported only in Moore charts. For more information, see “Design Guidelines for Moore Charts”.

OnAction — State on actions

cell array of character vectors

This property is read-only.

State on actions, specified as a cell array of character vectors in the form

```
{'trigger1','action1',...,'triggerN','actionN'}
```

The value of this property depends on the `LabelString` property for the state. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16. This property is not supported in Moore charts.

IsExplicitlyCommented — Whether to comment out state

false or 0 (default) | true or 1

Whether to comment out the state, specified as a numeric or logical 1 (`true`) or 0 (`false`). Setting this property to `true` is equivalent to right-clicking the state and selecting **Comment Out**. For more information, see “Comment Out Objects in a Stateflow Chart”.

IsImplicitlyCommented — Whether state is implicitly commented out

`true` or 1 | `false` or 0

This property is read-only.

Whether the state is implicitly commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). The state is implicitly commented out when you explicitly comment out an object that contains it. If the state is contained in an atomic subchart, this property is `false` unless the explicitly commented object is also contained in the atomic subchart.

IsCommented — Whether state is commented out


`true` or 1 | `false` or 0

This property is read-only.

Whether the state is commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property is `true` when either `IsExplicitlyCommented` or `IsImplicitlyCommented` is `true`.

CommentText — Comment text

`""` (default) | string scalar | character vector

Comment text added to the state, specified as a string scalar or character vector. This property applies only when the `IsExplicitlyCommented` property is `true`. In the Stateflow Editor, when you point to the comment badge  on the state, the text appears as a tooltip. When you set the `IsExplicitlyCommented` property to `false`, the value of `CommentText` reverts to `""`.

Graphical Appearance

Position — Position and size of state

`[0 0 90 60]` (default) | `[left top width height]`

Position and size of the state, specified as a four-element numeric vector of the form `[left top width height]`.

BadIntersection — Whether state intersects a box, state, or function

`true` or 1 | `false` or 0

This property is read-only.

Whether the state graphically intersects a box, state, or function, specified as a numeric or logical 1 (`true`) or 0 (`false`).

IsGrouped — Whether state is a grouped state

`false` or 0 (default) | `true` or 1

Whether the state is a grouped state, specified as a numeric or logical 1 (`true`) or 0 (`false`). When you copy and paste a grouped state, you copy not only the state but all of its contents. For more information, see “Copy and Paste by Grouping” on page 2-28.

IsSubchart — Whether state is a subchart

`false` or 0 (default) | `true` or 1

Whether the state is a subchart, specified as a numeric or logical 1 (`true`) or 0 (`false`).

ContentPreviewEnabled — Whether to display preview of state contents

`false` or 0 (default) | `true` or 1

Whether to display a preview of the state contents, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property applies only when the `IsSubchart` property is `true`.

ArrowSize — Size of incoming transition arrows

8 (default) | scalar

Size of incoming transition arrows, specified as a scalar.

FontSize — Font size for state label

scalar

Font size for the state label, specified as a scalar. The `StateFont.Size` property of the chart that contains the state sets the initial value of this property.

State Decomposition**Decomposition — Decomposition of substates**

"EXCLUSIVE_OR" (default) | "PARALLEL_AND"

Decomposition of substates at the top level of containment in the state, specified as "EXCLUSIVE_OR" or "PARALLEL_AND". For more information, see "Specify Substate Decomposition".

Type — Decomposition of sibling states

'AND' | 'OR'

This property is read-only.

Decomposition of sibling states, specified as 'AND' or 'OR'. The state inherits this property from the `Decomposition` property of its parent state or chart.

ExecutionOrder — Execution order in parallel (AND) decomposition

scalar

Execution order for the state in parallel (AND) decomposition, specified as an integer scalar. This property applies only when both of these conditions are satisfied:

- The `Type` property of the state is "AND".
- The `UserSpecifiedStateTransitionExecutionOrder` property of the chart that contains the state is `true`.

Active State Output**HasOutputData — Whether to create active state data output**

`false` or 0 (default) | `true` or 1

Whether to create an active state data output port for the state, specified as a numeric or logical 1 (`true`) or 0 (`false`). For more information, see "Monitor State Activity Through Active State Data".

OutputData — Active state data object

`Stateflow.Data` object

This property is read-only.

Active state data object for the state, specified as a `Stateflow.Data` object. This property applies only when the `HasOutputData` property for the state is `true`.

OutputPortName — Name of active state data object

string scalar | character vector

Name of the active state data object for the state, specified as a string scalar or character vector. This property applies only when the `HasOutputData` property for the state is `true`.

OutputMonitoringMode — Monitoring mode for active state output

"SelfActivity" (default) | "ChildActivity" | "LeafStateActivity"

Monitoring mode for the active state output data, specified as "SelfActivity", "ChildActivity", or "LeafStateActivity".

EnumTypeName — Name of enumerated data type for active state data object

string scalar | character vector

Name of the enumerated data type for the active state data object for the state, specified as a string scalar or character vector. This property applies only when the `OutputMonitoringMode` property for the state is "ChildActivity" or "LeafStateActivity". For more information, see "Enum Name".

DoNotAutogenerateEnum — Whether to define enumerated data type manually

false or 0 (default) | true or 1

Whether to define the enumerated data type for the active state data output manually, specified as a numeric or logical 1 (true) or 0 (false). This property applies only when the `OutputMonitoringMode` property for the state is "ChildActivity" or "LeafStateActivity". For more information, see "Define State Activity Enumeration Type".

Signal Logging and Test Point Monitoring

LoggingInfo — Signal logging properties

`Stateflow.SigLoggingInfo` object

Signal logging properties for the state, specified as a `Stateflow.SigLoggingInfo` object with these properties:

- **DataLogging** — Whether to enable signal logging, specified as a numeric or logical 1 (true) or 0 (false).
- **DecimateData** — Whether to limit the amount of logged data, specified as a numeric or logical 1 (true) or 0 (false).
- **Decimation** — Decimation interval, specified as an integer scalar. This property applies only when the `DecimateData` property is `true`.
- **LimitDataPoints** — Whether to limit the number of data points to log, specified as a numeric or logical 1 (true) or 0 (false).
- **MaxPoints** — Maximum number of data points to log, specified as an integer scalar. This property applies only when the `LimitDataPoints` property is `true`.
- **NameMode** — Source of the signal name, specified as "SignalName" or "Custom".

- **LoggingName** — Custom signal name, specified as a string scalar or character vector. This property applies only when the `NameMode` property is "Custom".

Signal logging saves the self activity of the state to the MATLAB workspace during simulation. For more information, see "Log Simulation Output for States and Data".

Example: `state.LoggingInfo.DataLogging = true;`

TestPoint — Whether to set state as test point

false or 0 (default) | true or 1

Whether to set the state as a test point, specified as a numeric or logical 1 (true) or 0 (false). You can monitor testpoints with a floating scope during simulation. You can also log test point values to the MATLAB workspace. For more information, see "Monitor Test Points in Stateflow Charts".

Debugging

Debug — Debugger properties

`Stateflow.StateDebug` object

Debugger properties for the state, specified as a `Stateflow.StateDebug` object with these properties:

- **OnEntry** — Whether to set the `On State Entry` breakpoint, specified as a numeric or logical 1 (true) or 0 (false).
- **OnDuring** — Whether to set the `During State` breakpoint, specified as a numeric or logical 1 (true) or 0 (false).
- **OnExit** — Whether to set the `On State Exit` breakpoint, specified as a numeric or logical 1 (true) or 0 (false).

For more information, see "Set Breakpoints to Debug Charts".

Example: `state.Debug.Breakpoints.OnEntry = true;`

Example: `state.Debug.Breakpoints.OnDuring = true;`

Example: `state.Debug.Breakpoints.OnExit = true;`

Code Generation

InlineOption — Appearance in generated code

"Auto" (default) | "Function" | "Inline"

Appearance of the state functions in generated code, specified as one of these values:

- "Auto" — An internal calculation determines the appearance of state functions in generated code.
- "Function" — State functions are implemented as separate static functions.
- "Inline" — Calls to state functions are replaced by code as long as the function is not part of a recursion.

For more information, see "Inline State Functions in Generated Code" (Simulink Coder).

Hierarchy

Chart — Chart that contains state

Stateflow.Chart object

This property is read-only.

Chart that contains the state, specified as a Stateflow.Chart object.

Subviewer — Subviewer for state

Stateflow.Chart object | Stateflow.State object | Stateflow.Box object | Stateflow.Function object

This property is read-only.

Subviewer for the state, specified as a Stateflow.Chart, Stateflow.State, or Stateflow.Box object. The subviewer is the chart or subchart where you can graphically view the state.

Machine — Machine that contains state

Stateflow.Machine object

This property is read-only.

Machine that contains the state, specified as a Stateflow.Machine object.

Path — Location of parent in model hierarchy

character vector

This property is read-only.

Location of the parent of the state in the model hierarchy, specified as a character vector.

Identification

Description — Description

"" (default) | string scalar | character vector

Description for the state, specified as a string scalar or character vector.

Document — Document link

"" (default) | string scalar | character vector

Document link for the state, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the state, specified as data of any type.

SSIdNumber — Session-independent identifier

scalar

This property is read-only.

Session-independent identifier, specified as an integer scalar. Use this property to distinguish the state from other objects in the model.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Unlike `SSIdNumber`, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>find</code>	Identify specified objects in hierarchy
<code>getChildren</code>	Identify children of object
<code>getParent</code>	Identify parent of object
<code>defaultTransitions</code>	Identify default transitions in specified object
<code>innerTransitions</code>	Identify inner transitions with specified source
<code>outerTransitions</code>	Identify outgoing transitions with specified source
<code>sinkedTransitions</code>	Identify transitions with specified destination
<code>sourcedTransitions</code>	Identify transitions with specified source
<code>getReferences</code>	Identify references to symbol name
<code>renameReferences</code>	Rename symbol and update references to symbol name
<code>commentedBy</code>	Identify objects that implicitly comment out a graphical object
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>highlight</code>	Highlight graphical object
<code>fitToView</code>	Zoom in on graphical object

Examples**Add State to Chart**

Add a state in the chart `ch`. Set its name to `A`.

```
state = Stateflow.State(ch);  
state.Name = "A";
```

Enter Multiline Label in State

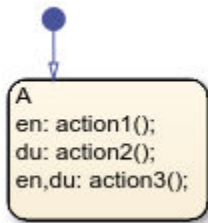
To enter a multiline label in the state `state`, you can:

- Call the MATLAB function `sprintf` and use the escape sequence `\n` to insert newline characters:

```
str = sprintf("A\nen: action1();\ndu: action2();\nen,du: action3();");  
sA.LabelString = str;
```

- Enter a concatenated text expression that uses the function `newline` to create newline characters:

```
str = "A" + newline + ...  
      "en: action1();" + newline + ...  
      "du: action2();" + newline + ...  
      "en,du: action3();"  
sA.LabelString = str;
```



To extract the state name, entry action, and during action specified by the state label, enter:

```
name = state.Name
name =
    'A'
entry = state.EntryAction
entry =
    ' action1();
    action3();'
during = state.DuringAction
during =
    ' action2();
    action3();'
```

For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16.

Add Supertransition from Subchart

Create a supertransition that connects a junction inside a subchart to a junction outside the subchart.



Open the model and access the `Stateflow.Chart` object for the chart.

```
open_system("sfSupertransitionAPIExample")
ch = find(sfroot, "-isa", "Stateflow.Chart");
```

Access the `Stateflow.State` object for the subchart and the `Stateflow.Junction` objects for the junctions.

```
st = find(ch, "-isa", "Stateflow.State");
j1 = find(st, "-isa", "Stateflow.Junction");
j2 = find(ch, "-isa", "Stateflow.Junction", "-depth", 1);
```

Save the original position of the subchart to a temporary workspace variable `subchartPosition`.

```
subchartPosition = st.Position;
```

Convert the subchart to a normal state by setting its `IsSubchart` and `IsGrouped` properties to `false`.

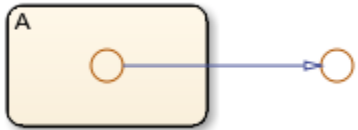
```
st.IsSubchart = false;
st.IsGrouped = false;
```

When you convert a subchart to a normal state, it may change size to display its contents.



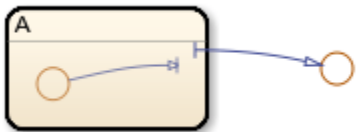
Add a transition that connects junction `j1` to junction `j2`.

```
tr = Stateflow.Transition(ch);
tr.Source = j1;
tr.Destination = j2;
```



Revert the state to a subchart by setting its `IsSubchart` property to `true`. Restore the subchart to its original position.

```
st.IsSubchart = true;
st.Position = subchartPosition;
```



The transition between the junctions is now a supertransition that crosses the boundary of the subchart.

Version History

Introduced before R2006a

R2023a: New object functions and properties

Errors starting in R2023a

`Stateflow.State` objects have new object functions and properties:

- The object function `getReferences` returns the locations where a chart refers to the name of a state.
- The object function `renameReferences` renames a state and updates all references to the state name in the chart.
- The object function `commentedBy` identifies the explicitly commented objects that cause a state to be commented out.
- The property `IsCommented` indicates whether a state is commented out. This property replaces the object function `isCommented`.

See Also

`Stateflow.Box` | `Stateflow.Chart` | `Stateflow.Transition`

Topics

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

“Represent Operating Modes by Using States”

“Specify Labels in States and Transitions Programmatically” on page 1-16

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.StateBreakpoints

Breakpoint properties for state

Description

Use a `Stateflow.StateBreakpoints` object to specify the breakpoint properties for a state, atomic subchart, or Simulink based state. For more information, see “Set Breakpoints to Debug Charts”.

Creation

Each state, atomic subchart, and Simulink based state has its own `Stateflow.StateBreakpoints` object. To access the `Stateflow.StateBreakpoints` object, use the `Debug.Breakpoints` property of the `Stateflow.State`, `Stateflow.AtomicSubchart`, or `Stateflow.SimulinkBasedState` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

OnEntry — Whether to set On State Entry breakpoint

false or 0 (default) | true or 1

Whether to set the On State Entry breakpoint, specified as a numeric or logical 1 (true) or 0 (false).

OnDuring — Whether to set During State breakpoint

false or 0 (default) | true or 1

Whether to set the During State breakpoint, specified as a numeric or logical 1 (true) or 0 (false).

OnExit — Whether to set On State Exit breakpoint

false or 0 (default) | true or 1

Whether to set the On State Exit breakpoint, specified as a numeric or logical 1 (true) or 0 (false).

Examples

Set Breakpoints for State

Access the `Stateflow.StateDebug` and `Stateflow.StateBreakpoints` objects for the `Stateflow.State` object state.


```
debug = state.Debug;  
breakpoints = debug.Breakpoints;
```

Set the On State Entry, During State, and On State Exit breakpoints.

```
breakpoints.OnEntry = true;  
breakpoints.OnDuring = true;  
breakpoints.OnExit = true;
```

Version History

Introduced before R2006a

See Also

[Stateflow.State](#) | [Stateflow.SimulinkBasedState](#) | [Stateflow.AtomicSubchart](#)

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Set Breakpoints to Debug Charts”

Stateflow.StateDebug

Debugger properties for state

Description

Use a `Stateflow.StateDebug` object to specify the debugger properties for a state, atomic subchart, or Simulink based state.

Creation

Each state, atomic subchart, and Simulink based state has its own `Stateflow.StateDebug` object. To access the `Stateflow.StateDebug` object, use the `Debug` property for the `Stateflow.State`, `Stateflow.AtomicSubchart`, or `Stateflow.SimulinkBasedState` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Breakpoints — Breakpoint properties

`Stateflow.StateBreakpoints` object

Breakpoint properties for the state, atomic subchart, or Simulink based state, specified as a `Stateflow.StateBreakpoints` object with these properties:

- **OnEntry** — Whether to set the `On State Entry` breakpoint, specified as a numeric or logical 1 (true) or 0 (false).
- **OnDuring** — Whether to set the `During State` breakpoint, specified as a numeric or logical 1 (true) or 0 (false).
- **OnExit** — Whether to set the `On State Exit` breakpoint, specified as a numeric or logical 1 (true) or 0 (false).

For more information, see “Set Breakpoints to Debug Charts”.

Examples

Set Breakpoints for State

Access the `Stateflow.StateDebug` and `Stateflow.StateBreakpoints` objects for the `Stateflow.State` object state.

```
debug = state.Debug;  
breakpoints = debug.Breakpoints;
```

Set the On State Entry, During State, and On State Exit breakpoints.

```
breakpoints.OnEntry = true;  
breakpoints.OnDuring = true;  
breakpoints.OnExit = true;
```

Version History

Introduced before R2006a

See Also

[Stateflow.State](#) | [Stateflow.SimulinkBasedState](#) | [Stateflow.AtomicSubchart](#)

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Set Breakpoints to Debug Charts”

Stateflow.StateFont

Font for box, function, and state labels

Description

Use a `Stateflow.StateFont` object to specify the font properties for box, function, and state labels in a chart.

Creation

Each chart has its own `Stateflow.StateFont` object. To access the `Stateflow.StateFont` object, use the `StateFont` property for the `Stateflow.Chart` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Name — Font name

"Helvetica" (default) | string scalar | character vector

Font name, specified as a string scalar or character vector. This property also determines the font for annotations in the chart.

Angle — Font angle

"NORMAL" (default) | "ITALIC"

Font angle, specified as "NORMAL" or "ITALIC".

Weight — Font weight

"NORMAL" (default) | "BOLD"

Font weight, specified as "NORMAL" or "BOLD".

Size — Default font size

12 (default) | scalar

Default font size for new boxes, functions, and states in the chart, specified as a scalar. This property also determines the default font size for new annotations in the chart.

Examples

Change Font Properties for State Labels

Access the `Stateflow.StateFont` object for the `Stateflow.Chart` object `ch`.

```
font = ch.StateFont;
```

Set the font for box, function, and state labels to Arial. Set the font angle to italics and the font weight to bold. Set the default font size to 8.

```
font.Name = "Arial";  
font.Angle = "ITALIC";  
font.Weight = "BOLD";  
font.Size = 8;
```

Version History

Introduced before R2006a

See Also

Stateflow.Chart

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.StateTransitionTableChart

Tabular representation of state machine for modal logic

Description

Use a `Stateflow.StateTransitionTableChart` object to represent a finite state machine for sequential modal logic in tabular format. Instead of drawing states and transitions in a Stateflow chart, you can use a State Transition Table block to model a state machine in a concise, compact format that requires minimal maintenance of graphical objects. For more information, see “Use State Transition Tables to Express Sequential Logic in Tabular Form”.

Creation

To create a `Stateflow.StateTransitionTableChart` object, call the function `sfnew` with the `-STT` argument. For example, to create a State Transition Table block in a new Simulink model called `myModel`, enter:

```
sfnew -STT myModel
```

Alternatively, you can add a new State Transition Table block to an existing model by using the function `add_block`:

```
add_block("sflib/State Transition Table", ...  
         "myModel/State Transition Table")
```

Then, to access the `Stateflow.StateTransitionTableChart` object, call the `find` function for the `Simulink.Root` object:

```
table = find(sfroot, "-isa", "Stateflow.StateTransitionTableChart", ...  
            Path="myModel/State Transition Table");
```

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Name — Name of state transition table

"State Transition Table" (default) | string scalar | character vector

Name of the state transition table, specified as a string scalar or character vector.

ActionLanguage — Action language

"MATLAB" (default) | "C"

Action language used to program the state transition table, specified as "MATLAB" or "C". For more information, see "Differences Between MATLAB and C as Action Language Syntax".

StateMachineType — State machine semantics

"Classic" (default) | "Mealy" | "Moore"

State machine semantics implemented by the state transition table, specified as "Classic", "Mealy", or "Moore". For more information, see "Overview of Mealy and Moore Machines".

SupportVariableSizing — Whether state transition table supports variable-size data

true or 1 (default) | false or 0

Whether the state transition table supports variable-size data, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Declare Variable-Size Data in Stateflow Charts".

Chart Initialization

ExecuteAtInitialization — Whether to initialize state configuration

false or 0 (default) | true or 1

Whether to initialize the state configuration of the state transition table at time zero instead of at the first input event, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Execution of a Chart at Initialization".

StatesWhenEnabling — Behavior of states when event reenables state transition table

"" (default) | "held" | "reset"

Behavior of the states when a function-call input event reenables the state transition table, specified as one of these values:

- "" — The state transition table does not contain function-call input events.
- "held" — The state transition table maintains the most recent values of the states.
- "reset" — The state transition table reverts to the initial conditions of the states.

For more information, see "Control States in Charts Enabled by Function-Call Input Events".

InitializeOutput — Whether to initialize output data

false or 0 (default) | true or 1

Whether to initialize the output data every time the state transition table wakes up, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Initialize outputs every time chart wakes up".

Active State Output

HasOutputData — Whether to create active state data output

false or 0 (default) | true or 1

Whether to create an active state data output port for the state transition table, specified as a numeric or logical 1 (true) or 0 (false). For more information, see "Monitor State Activity Through Active State Data".

OutputData — Active state data object

Stateflow.Data object

This property is read-only.

Active state data object for the state transition table, specified as a `Stateflow.Data` object. This property applies only when the `HasOutputData` property for the state transition table is `true`.

OutputPortName — Name of active state data object

string scalar | character vector

Name of the active state data object for the state transition table, specified as a string scalar or character vector. This property applies only when the `HasOutputData` property for the state transition table is `true`.

OutputMonitoringMode — Monitoring mode for active state output

"ChildActivity" (default) | "LeafStateActivity"

Monitoring mode for the active state output data, specified as "ChildActivity" or "LeafStateActivity".

EnumTypeName — Name of enumerated data type for active state data object

string scalar | character vector

Name of the enumerated data type for the active state data object for the state transition table, specified as a string scalar or character vector. For more information, see "Enum Name".

DoNotAutogenerateEnum — Whether to define enumerated data type manually

false or 0 (default) | true or 1

Whether to define the enumerated data type for the active state data output manually, specified as a numeric or logical 1 (`true`) or 0 (`false`). For more information, see "Define State Activity Enumeration Type".

Discrete and Continuous-Time Semantics**ChartUpdate — Activation method for state transition table**

"INHERITED" (default) | "CONTINUOUS" | "DISCRETE"

Activation method for the state transition table, specified as "CONTINUOUS", "DISCRETE", or "INHERITED". For more information, see "Update Method".

SampleTime — Sample time for activating state transition table

"-1" (default) | string scalar | character vector

Sample time for activating the state transition table, specified as a string scalar or character vector. This property applies only when the `ChartUpdate` property for the state transition table is "DISCRETE".

EnableZeroCrossings — Whether to enable zero-crossing detection

true or 1 (default) | false or 0

Whether to enable zero-crossing detection on state transitions in the state transition table, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property applies only when the `ChartUpdate` property for the state transition table is set to "CONTINUOUS". For more information, see "Disable Zero-Crossing Detection".

Super Step Semantics

EnableNonTerminalStates — Whether to enable super step semantics

false or 0 (default) | true or 1

Whether to enable super step semantics for the state transition table, specified as a numeric or logical 1 (true) or 0 (false). For more information, see “Super Step Semantics”.

NonTerminalMaxCounts — Maximum number of transitions in one super step

1000 (default) | scalar

Maximum number of transitions the state transition table can take in one super step, specified as an integer scalar. This property applies only when the EnableNonTerminalStates property for the state transition table is true.

NonTerminalUnstableBehavior — Behavior if super step exceeds maximum number of transitions

"Proceed" (default) | "Throw Error"

Behavior if a super step for the state transition table exceeds the maximum number of transitions specified in the NonTerminalMaxCounts property before reaching a stable state, specified as one of these values:

- "Proceed" — The state transition table goes to sleep with the last active state configuration.
- "Throw Error" — The state transition table generates an error.

This property applies only when the EnableNonTerminalStates property for the state transition table is true.

Integer and Fixed-Point Data

SaturateOnIntegerOverflow — Whether data saturates on integer overflow

true or 1 (default) | false or 0

Whether the data in the state transition table saturates on integer overflow, specified as a numeric or logical 1 (true) or 0 (false). When this property is disabled, the data in the state transition table wraps on integer overflow. For more information, see “Handle Integer Overflow for Chart Data”.

TreatAsFi — Inherited Simulink signals to treat as fi objects

"Fixed-point" (default) | "Fixed-point & Integer"

Inherited Simulink signals to treat as Fixed-Point Designer fi objects, specified as one of these values:

- "Fixed-point" — The state transition table treats all fixed-point inputs as fi objects.
- "Fixed-point & Integer" — The state transition table treats all fixed-point and integer inputs as fi objects.

This property applies only when the ActionLanguage property of the state transition table is "MATLAB".

EmfDefaultFimath — Default fimath properties

"Same as MATLAB Default" (default) | "Other:UserSpecified"

Default fimath properties for the state transition table, specified as one of these values:

- "Same as MATLAB Default" — Use the same `fimath` properties as the current default `fimath` object.
- "Other:UserSpecified" — Use the `InputFimath` property to specify the default `fimath` object.

This property applies only when the `ActionLanguage` property of the state transition table is "MATLAB".

InputFimath — Default fimath object

string scalar | character vector

Default `fimath` object, specified as a string scalar or character vector. When the `EmfDefaultFimath` property for the state transition table is "Other:UserSpecified", you can use this property to:

- 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 `ActionLanguage` property of the state transition table is "MATLAB".

C Action Language

EnableBitOps — Whether to use bit operations

false or 0 (default) | true or 1

Whether to use bit operations in state and transition actions in the state transition table, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property applies only to state transition tables that use C as the action language. For more information, see "Enable C-bit operations".

Debugging

Debug — Debugger properties

`Stateflow.ChartDebug` object

Debugger properties for the state transition table, specified as a `Stateflow.ChartDebug` object with this property:

- **Breakpoints.OnEntry** — Whether to set the `On Chart Entry` breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).

For more information, see "Set Breakpoints to Debug Charts".

Example: `table.Debug.Breakpoints.OnEntry = true;`

Graphical Appearance

Editor — Editor

`Stateflow.Editor` object

This property is read-only.

Editor for the state transition table, specified as a `Stateflow.Editor` object. You can use this object to control the position, size, and magnification level of the Stateflow Editor window.

Visible — Whether editor is displaying state transition table

true or 1 | false or 0

Whether the Stateflow Editor window is displaying the state transition table, specified as a numeric or logical 1 (true) or 0 (false).

ChartColor — Background color

[1 0.9608 0.8824] (default) | [red green blue]

Background color for the chart that is automatically generated for the state transition table, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.

StateColor — Color for states

[0 0 0] (default) | [red green blue]

Color for the states in the chart that is automatically generated for the state transition table, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.

TransitionColor — Color for transitions

[0.2902 0.3294 0.6039] (default) | [red green blue]

Color for transitions in the chart that is automatically generated for the state transition table, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.

JunctionColor — Color for junctions

[0.6824 0.3294 0] (default) | [red green blue]

Color for junctions in the chart that is automatically generated for the state transition table, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.

StateFont — Font for state labels

Stateflow.STTStateFont object

Font for the state labels in the chart that is automatically generated for the state transition table, specified as a Stateflow.STTStateFont object with these properties:

- **Name** — Font name, specified as a string scalar or character vector.
- **Angle** — Font angle, specified as "NORMAL" or "ITALIC".
- **Weight** — Font weight, specified as "NORMAL" or "BOLD".
- **Size** — Default font size for new states, specified as a scalar.

Example: table.StateFont.Name = "Arial";

Example: table.StateFont.Angle = "ITALIC";

Example: table.StateFont.Weight = "BOLD";

Example: table.StateFont.Size = 8;

StateLabelColor — Color for state labels

[0 0 0] (default) | [red green blue]

Color for the state labels in the chart that is automatically generated for the state transition table, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.

TransitionFont — Font for transition labels

Stateflow.STTTransFont object

Font for the transition labels in the chart that is automatically generated for the state transition table, specified as a Stateflow.STTTransFont object with these properties:

- **Name** — Font name, specified as a string scalar or character vector.
- **Angle** — Font angle, specified as "NORMAL" or "ITALIC".
- **Weight** — Font weight, specified as "NORMAL" or "BOLD".
- **Size** — Default font size for new transitions, specified as a scalar.

Example: table.TransitionFont.Name = "Arial";

Example: table.TransitionFont.Angle = "ITALIC";

Example: table.TransitionFont.Weight = "BOLD";

Example: table.TransitionFont.Size = 8;

TransitionLabelColor — Color for transition labels

[0.2902 0.3294 0.6039] (default) | [red green blue]

Color for the transition labels in the chart that is automatically generated for the state transition table, specified as a three-element numeric vector of the form [red green blue] that specifies the red, green, and blue values. Each element must be in the range between 0 and 1.

Hierarchy**Machine — Machine that contains state transition table**

Stateflow.Machine object

This property is read-only.

Machine that contains the state transition table, specified as a Stateflow.Machine object.

Path — Location of state transition table in model hierarchy

character vector

This property is read-only.

Location of the state transition table in the model hierarchy, specified as a character vector.

Dirty — Whether state transition table has changed

true or 1 | false or 0

Whether the state transition table has changed after being opened or saved, specified as a numeric or logical 1 (true) or 0 (false).

Locked — Whether state transition table is locked

false or 0 (default) | true or 1

Whether the state transition table is locked, specified as a numeric or logical 1 (`true`) or 0 (`false`). Enable this property to prevent changes in the state transition table.

Iced — Whether state transition table is locked

`false` or 0 (default) | `true` or 1

This property is read-only.

Whether the state transition table is locked, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property is equivalent to the property `Locked`, but is used internally to prevent changes in the state transition table during simulation.

Identification

Description — Description

`""` (default) | string scalar | character vector

Description for the state transition table, specified as a string scalar or character vector.

Document — Document link

`""` (default) | string scalar | character vector

Document link for the state transition table, specified as a string scalar or character vector.

Tag — User-defined tag

`[]` (default) | any data type

User-defined tag for the state transition table, specified as data of any type.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Use this property to distinguish the state transition table from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>find</code>	Identify specified objects in hierarchy
<code>getChildren</code>	Identify children of object
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>exportAsStruct</code>	Export contents of state transition table as structure array

Examples

Create Empty State Transition Table

Call the function `sfnew` with the `-STT` argument to open a new Simulink model that contains an empty State Transition Table block.

```
sfnew -STT
```

Access the `Simulink.Root` object by calling the `sfroot` function.

```
rt = sfroot;
```

Access the `Stateflow.StateTransitionTableChart` object by calling the `find` function for the `Simulink.Root` object.

```
table = find(rt, "-isa", "Stateflow.StateTransitionTableChart");
```

Create Array of Structures

Export the contents of the state transition table in “Model Bang-Bang Controller by Using a State Transition Table” as an array of structures. This state transition table contains two top-level states and three substates.

Access `Stateflow.StateTransitionTableChart` for the state transition table.

```
table = find(sfroot, "-isa", "Stateflow.StateTransitionTableChart");
```

Export the contents of the state transition table as an array of structures.

```
structure = exportAsStruct(table)
```

```
structure =
```

```
1x5 struct array with fields:
```

```
    rowText
    depth
    rowType
    isDefaultTransitionOwner
    isWhenState
    hasHistory
    isExpanded
    outlinedTransitionIdxs
    sfObjectInfo
    aslInfo
    decompositionInfo
```

View the contents of a top-level state.

```
structure(1)
```

```
ans =
```

```
struct with fields:
```

```
    rowText: {'Normal' {3x1 cell} {3x1 cell}}
    depth: 1
    rowType: 0
    isDefaultTransitionOwner: 1
    isWhenState: 0
    hasHistory: 0
    isExpanded: 1
    outlinedTransitionIdxs: [0 0 0]
    sfObjectInfo: [1x3 struct]
```

```

        aslInfo: [1×1 struct]
    decompositionInfo: [1×1 struct]

```

View the contents of a child state.

```
structure(4)
```

```
ans =
```

```
    struct with fields:
```

```

        rowText: {'OnEntry:boiler_cmd = 1;' {3×1 cell} {3×1 cell}}
        depth: 2
        rowType: 0
    isDefaultTransitionOwner: 0
        isWhenState: 0
        hasHistory: 0
        isExpanded: 0
    outlinedTransitionIdxs: [0 0 0]
        sfObjectInfo: [1×3 struct]
        aslInfo: [1×1 struct]
    decompositionInfo: [1×1 struct]

```

Version History

Introduced in R2012b

R2022b: Exports contents of state transition tables

Export the contents of a state transition table as a structure array by calling the object function `exportAsStruct`.

See Also

Blocks

State Transition Table

Functions

`sfnew` | `sfroot` | `add_block`

Topics

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

“Model Finite State Machines by Using Stateflow Charts”

“Use State Transition Tables to Express Sequential Logic in Tabular Form”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.STTStateFont

Font for state labels in state transition tables

Description

Use a `Stateflow.STTStateFont` object to specify the font properties for state labels in the chart that is automatically generated for a state transition table.

Creation

Each state transition table has its own `Stateflow.STTStateFont` object. To access the `Stateflow.STTStateFont` object, use the `StateFont` property for the `Stateflow.StateTransitionTableChart` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Name — Font name

"Helvetica" (default) | string scalar | character vector

Font name, specified as a string scalar or character vector.

Angle — Font angle

"NORMAL" (default) | "ITALIC"

Font angle, specified as "NORMAL" or "ITALIC".

Weight — Font weight

"NORMAL" (default) | "BOLD"

Font weight, specified as "NORMAL" or "BOLD".

Size — Default font size

12 (default) | scalar

Default font size for new states in the state transition table, specified as a scalar.

Examples

Change Font Properties for State Labels

Access the `Stateflow.STTStateFont` object for the `Stateflow.StateTransitionTableChart` object `stt`.


```
font = stt.StateFont;
```

Set the font for state labels to Arial. Set the font angle to italics and the font weight to bold. Set the default font size to 8.

```
font.Name = "Arial";  
font.Angle = "ITALIC";  
font.Weight = "BOLD";  
font.Size = 8;
```

Version History

Introduced before R2006a

See Also

Stateflow.StateTransitionTableChart

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.STTTransFont

Font properties for transition labels in state transition tables

Description

Use a `Stateflow.STTTransFont` object to specify the font properties for transition labels in the chart that is automatically generated for a state transition table.

Creation

Each state transition table has its own `Stateflow.STTTransFont` object. To access the `Stateflow.STTTransFont` object, use the `TransitionFont` property for the `Stateflow.StateTransitionTableChart` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Name — Font name

"Helvetica" (default) | string scalar | character vector

Font name, specified as a string scalar or character vector.

Angle — Font angle

"NORMAL" (default) | "ITALIC"

Font angle, specified as "NORMAL" or "ITALIC".

Weight — Font weight

"NORMAL" (default) | "BOLD"

Font weight, specified as "NORMAL" or "BOLD".

Size — Default font size

12 (default) | scalar

Default font size for new transitions in the state transition table, specified as a scalar.

Examples

Change Font Properties for Transition Labels

Access the `Stateflow.STTTransFont` object for the `Stateflow.StateTransitionTableChart` object `stt`.

```
font = stt.TransitionFont;
```

Set the font for transition labels to Arial. Set the font angle to italics and the font weight to bold. Set the default font size to 8.

```
font.Name = "Arial";  
font.Angle = "ITALIC";  
font.Weight = "BOLD";  
font.Size = 8;
```

Version History

Introduced before R2006a

See Also

Stateflow.StateTransitionTableChart

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.SymbolReference

Reference to symbol name

Description

`Stateflow.SymbolReference` objects indicate where a Stateflow chart refers to a symbol name. For example, a `Stateflow.SymbolReference` object can correspond to a state or transition action that includes a symbol name. Symbols include `Stateflow.Data`, `Stateflow.Event`, `Stateflow.Message`, and other Stateflow objects that have a `Name` property.

Creation

The Stateflow chart maintains an array of `Stateflow.SymbolReference` objects for each symbol. To access this array, call the object function `getReferences`.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

WhereUsed — Object that references symbol name

`Stateflow.State` object | `Stateflow.Transition` object | `Stateflow.EMFunction` object | ...

This property is read-only.

Object that references the symbol name, specified as a Stateflow API object of one of these types:

- `Stateflow.AtomicBox`
- `Stateflow.AtomicSubchart`
- `Stateflow.EMFunction`
- `Stateflow.SimulinkBasedState`
- `Stateflow.State`
- `Stateflow.Transition`
- `Stateflow.TruthTable`

Position — Position of symbol name

integer vector | []

This property is read-only.

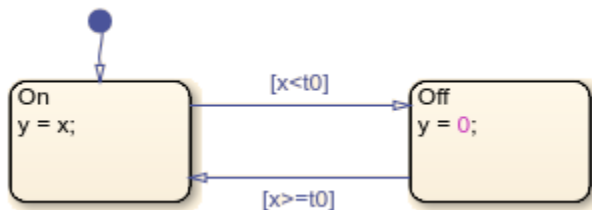
Position of the symbol name, specified as a two-element integer vector of the form [`start` `end`] or as an empty vector.

- If `WhereUsed` is a `Stateflow.State` or `Stateflow.Transition` object, the values of `start` and `end` indicate the positions of the first and last characters of the symbol name in the `LabelString` property of `WhereUsed`.
- If `WhereUsed` is a `Stateflow.EMFunction` object, the values of `start` and `end` indicate the positions of the first and last characters of the symbol name in the `Script` property of `WhereUsed`.
- If `WhereUsed` is an object of any other type, this property is an empty vector.

Examples

Rename Data and Update References in Chart

Rename and update the references to the chart output `y`.



Open the model and access the `Stateflow.Data` object for the chart output `y`.

```
open_system("sfRectifyAPIExample")
data = find(sfroot, "-isa", "Stateflow.Data", Scope="Output");
data.Name
```

```
ans =
'y'
```

Find the locations where the chart refers to the chart output.

```
references = getReferences(data)

references=2x1 object
2x1 SymbolReference array with properties:

    Position
    WhereUsed
```

Display the names and entry actions of the states that refer to the chart output.

```
whereUsed = [references.WhereUsed];
classes = arrayfun(@class,whereUsed,UniformOutput=false);
idx = (classes=="Stateflow.State");
states = whereUsed(idx);
get(states,{"Name" "EntryAction"})

ans = 2x2 cell
    {'On' }    {'y = x;'}
    {'Off'}    {'y = 0;'}
  
```

Change the Name property of the chart output to "z" and update the references to the output in the chart.

```
renameReferences(data, "z")  
data.Name
```

```
ans =  
'z'
```

Display the names and modified entry actions of the states that refer to the chart output.

```
get(states, {"Name" "EntryAction"})
```

```
ans = 2x2 cell  
    {'On' }      {'z = x;'}  
    {'Off'}     {'z = 0;'}  
    
```

Version History

Introduced in R2023a

See Also

Functions

find | getReferences | renameReferences

Objects

Stateflow.Data | Stateflow.Event | Stateflow.Message | Stateflow.State

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.TransBreakpoints

Breakpoint properties for transition

Description

Use a `Stateflow.TransBreakpoints` object to specify the breakpoint properties for a transition. For more information, see “Set Breakpoints to Debug Charts”.

Creation

Each transition has its own `Stateflow.TransBreakpoints` object. To access the `Stateflow.TransBreakpoints` object, use the `Debug.Breakpoints` property of the `Stateflow.Transition` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

WhenTested — Whether to set When Transition is Tested breakpoint

`false` or 0 (default) | `true` or 1

Whether to set the `When Transition is Tested` breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).

WhenValid — Whether to set When Transition is Valid breakpoint

`false` or 0 (default) | `true` or 1

Whether to set the `When Transition is Valid` breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).

Examples

Set Breakpoints for Transition

Access the `Stateflow.TransDebug` and `Stateflow.TransBreakpoints` objects for the `Stateflow.Transition` object `transition`.

```
debug = transition.Debug;  
breakpoints = debug.Breakpoints;
```

Set the `When Transition is Tested` and `When Transition is Valid` breakpoints.

```
breakpoints.WhenTested = true;  
breakpoints.WhenValid = true;
```

Version History

Introduced before R2006a

See Also

Stateflow.Transition

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Set Breakpoints to Debug Charts”

Stateflow.TransDebug

Debugger properties for transition

Description

Use a `Stateflow.TransDebug` object to specify the debugger properties for a transition.

Creation

Each transition has its own `Stateflow.TransDebug` object. To access the `Stateflow.TransDebug` object, use the `Debug` property for the `Stateflow.Transition` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Breakpoints — Breakpoint properties

`Stateflow.TransBreakpoints` object

Breakpoint properties for the transition, specified as a `Stateflow.TransBreakpoints` object with these properties:

- **WhenTested** — Whether to set the `When Transition is Tested` breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).
- **WhenValid** — Whether to set the `When Transition is Valid` breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).

For more information, see “Set Breakpoints to Debug Charts”.

Examples

Set Breakpoints for Transition

Access the `Stateflow.TransDebug` and `Stateflow.TransBreakpoints` objects for the `Stateflow.Transition` object `transition`.

```
debug = transition.Debug;  
breakpoints = debug.Breakpoints;
```

Set the `When Transition is Tested` and `When Transition is Valid` breakpoints.

```
breakpoints.WhenTested = true;  
breakpoints.WhenValid = true;
```

Version History

Introduced before R2006a

See Also

Stateflow.Transition

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Set Breakpoints to Debug Charts”

Stateflow.TransFont

Font properties for transition labels

Description

Use a `Stateflow.TransFont` object to specify the font properties for transition labels in a chart.

Creation

Each chart has its own `Stateflow.TransFont` object. To access the `Stateflow.TransFont` object, use the `TransitionFont` property for the `Stateflow.Chart` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Name — Font name

"Helvetica" (default) | string scalar | character vector

Font name, specified as a string scalar or character vector.

Angle — Font angle

"NORMAL" (default) | "ITALIC"

Font angle, specified as "NORMAL" or "ITALIC".

Weight — Font weight

"NORMAL" (default) | "BOLD"

Font weight, specified as "NORMAL" or "BOLD".

Size — Default font size

12 (default) | scalar

Default font size for new transitions in the chart, specified as a scalar.

Examples

Change Font Properties for Transition Labels

Access the `Stateflow.TransFont` object for the `Stateflow.Chart` object `ch`.

```
font = ch.TransitionFont;
```

Set the font for transition labels to Arial. Set the font angle to italics and the font weight to bold. Set the default font size to 8.

```
font.Name = "Arial";  
font.Angle = "ITALIC";  
font.Weight = "BOLD";  
font.Size = 8;
```

Version History

Introduced before R2006a

See Also

Stateflow.Chart

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.Transition

Transition in chart, state, box, or function

Description

Use `Stateflow.Transition` objects to create transitions from one operating mode to another. For more information, see “Transition Between Operating Modes”.

Creation

Syntax

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

Description

`transition = Stateflow.Transition(parent)` creates a `Stateflow.Transition` object in a parent chart, state, box, or graphical function.

Input Arguments

parent — Parent for new transition

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object

Parent for the new transition, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.Function`
- `Stateflow.State`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

LabelString — Label for transition

`""` (default) | string scalar | character vector

Label for the transition, specified as a string scalar or character vector. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16.

Condition — Transition condition

character vector

This property is read-only.

Transition condition, specified as a character vector. The value of this property depends on the `LabelString` property for the transition. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16.

ConditionAction — Transition condition action

character vector

This property is read-only.

Transition condition action, specified as a character vector. The value of this property depends on the `LabelString` property for the transition. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16.

TransitionAction — Transition action

character vector

This property is read-only.

Transition action, specified as a character vector. The value of this property depends on the `LabelString` property for the transition. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16.

Trigger — Transition trigger

character vector

This property is read-only.

Transition trigger, specified as a character vector. The value of this property depends on the `LabelString` property for the transition. For more information, see “Specify Labels in States and Transitions Programmatically” on page 1-16.

ExecutionOrder — Execution order for transition

scalar

Execution order for the transition when its source is active, specified as an integer scalar. This property applies only when the `UserSpecifiedStateTransitionExecutionOrder` property of the chart that contains the transition is `true`. For more information, see “Transition Evaluation Order”.

IsExplicitlyCommented — Whether to comment out transition

false or 0 (default) | true or 1

Whether to comment out the transition, specified as a numeric or logical 1 (true) or 0 (false). Setting this property to `true` is equivalent to right-clicking the transition and selecting **Comment Out**. For more information, see “Comment Out Objects in a Stateflow Chart”.

IsImplicitlyCommented — Whether transition is implicitly commented out

true or 1 | false or 0

This property is read-only.

Whether the transition is implicitly commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). The transition is implicitly commented out when you explicitly comment out an object that contains it or when you comment out its source or destination. If the transition is contained in an atomic subchart or an atomic box, this property is `false` unless the explicitly commented object is also contained in the atomic subchart or atomic box.

IsCommented — Whether transition is commented out


`true` or 1 | `false` or 0

This property is read-only.

Whether the transition is commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property is `true` when either `IsExplicitlyCommented` or `IsImplicitlyCommented` is `true`.

CommentText — Comment text

`" "` (default) | string scalar | character vector

Comment text added to the transition, specified as a string scalar or character vector. This property applies only when the `IsExplicitlyCommented` property is `true`. In the Stateflow Editor, when you point to the comment badge  on the transition, the text appears as a tooltip. When you set the `IsExplicitlyCommented` property to `false`, the value of `CommentText` reverts to `" "`.

Graphical Appearance

Source — Source of transition

`[]` (default) | `Stateflow.AtomicSubchart` object | `Stateflow.Junction` object | `Stateflow.SimulinkBasedState` object | `Stateflow.State` object

Source of the transition, specified as an empty array or a Stateflow API object of one of these types:

- `Stateflow.AtomicSubchart`
- `Stateflow.Junction`
- `Stateflow.SimulinkBasedState`
- `Stateflow.State`

SourceEndPoint — Position of transition endpoint at source

`[2 2]` (default) | `[x y]`

Position of the transition endpoint at its source, specified as a two-element numeric vector `[x y]` of coordinates relative to the upper left corner of the chart.

SourceOClock — Location of transition endpoint at source

0 (default) | scalar between 0 and 12

Location of the transition endpoint at its source, specified as a scalar between 0 and 12 that describes a clock position.

Destination — Destination of transition

`[]` (default) | `Stateflow.AtomicSubchart` object | `Stateflow.Junction` object | `Stateflow.SimulinkBasedState` object | `Stateflow.State` object

Destination of the transition, specified as an empty array or a Stateflow API object of one of these types:

- `Stateflow.AtomicSubchart`
- `Stateflow.Junction`
- `Stateflow.SimulinkBasedState`
- `Stateflow.State`

DestinationEndPoint — Position of transition endpoint at destination`[40 40] (default) | [x y]`

Position of the transition endpoint at its destination, specified as a two-element numeric vector `[x y]` of coordinates relative to the upper left corner of the chart.

DestinationOClock — Location of transition endpoint at destination`0 (default) | scalar between 0 and 12`

Location of the transition endpoint at its destination, specified as a scalar between 0 and 12 that describes a clock position.

MidPoint — Position of midpoint of transition`[21 21] (default) | [x y]`

Position of the midpoint of the transition, specified as a two-element numeric vector `[x y]` of coordinates relative to the upper left corner of the chart.

LabelPosition — Position and size of transition label`[0 0 8 14] (default) | [left top width height]`

Position and size of the transition label, specified as a four-element numeric vector of the form `[left top width height]`.

ArrowSize — Size of transition arrow`scalar`

Size of the transition arrow at the destination, specified as a scalar. When you change the destination of the transition, this property resets to the value of the `ArrowSize` property of the new destination.

FontSize — Font size for transition label`scalar`

Font size for the transition label, specified as a scalar. The `TransitionFont.Size` property of the chart that contains the transition sets the initial value of this property.

Debugging**Debug — Debugger properties**`Stateflow.TransDebug object`

Debugger properties for the transition, specified as a `Stateflow.TransDebug` object with these properties:

- **Breakpoints.WhenTested** — Whether to set the `When Transition is Tested` breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).
- **Breakpoints.WhenValid** — Whether to set the `When Transition is Valid` breakpoint, specified as a numeric or logical 1 (`true`) or 0 (`false`).

For more information, see “Set Breakpoints to Debug Charts”.

Example: `transition.Debug.Breakpoints.WhenTested = true;`

Example: `transition.Debug.Breakpoints.WhenValid = true;`

Code Generation

IsVariant — Whether transition is a variant transition

false or 0 (default) | true or 1

Whether the transition is a variant transition, specified as a numeric or logical 1 (true) or 0 (false). For more information, see “Control Indicator Lamp Dimmer Using Variant Conditions”.

Hierarchy

Chart — Chart that contains transition

Stateflow.Chart object

This property is read-only.

Chart that contains the transition, specified as a Stateflow.Chart object.

Subviewer — Subviewer for transition

Stateflow.Chart object | Stateflow.State object | Stateflow.Box object | Stateflow.Function object

This property is read-only.

Subviewer for the transition, specified as a Stateflow.Chart, Stateflow.State, Stateflow.Box, or Stateflow.Function object. The subviewer is the chart or subchart where you can graphically view the transition.

Machine — Machine that contains transition

Stateflow.Machine object

This property is read-only.

Machine that contains the transition, specified as a Stateflow.Machine object.

Path — Location of parent in model hierarchy

character vector

This property is read-only.

Location of the parent of the transition in the model hierarchy, specified as a character vector.

Identification

Description — Description

"" (default) | string scalar | character vector

Description for the transition, specified as a string scalar or character vector.

Document — Document link

"" (default) | string scalar | character vector

Document link for the transition, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the transition, specified as data of any type.

SSIdNumber — Session-independent identifier

scalar

This property is read-only.

Session-independent identifier, specified as an integer scalar. Use this property to distinguish the transition from other objects in the model.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Unlike `SSIdNumber`, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>getParent</code>	Identify parent of object
<code>commentedBy</code>	Identify objects that implicitly comment out a graphical object
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>highlight</code>	Highlight graphical object
<code>fitToView</code>	Zoom in on graphical object

Examples**Add Transition to Chart**

Add a transition that connects state `s1` to state `s2` in the chart `ch`.

```
transition = Stateflow.Transition(ch);  
transition.Source = s1;  
transition.Destination = s2;
```

Label Transitions

Add a label that specifies a trigger, condition, and condition action on the transition `transition`.

```
transition.LabelString = "trigger[guard]{action()}";
```



To extract the trigger, condition, and condition action specified by the transition label, enter:

```

trigger = transition.Trigger
trigger =
    'trigger'
condition = transition.Condition
condition =
    'guard'
action = transition.ConditionAction
action =
    'action();'
  
```

Add a Default Transition

Create a `Stateflow.Transition` object in the `Stateflow.Chart` object `ch`.

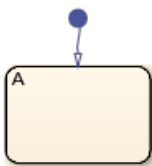
```
dt = Stateflow.Transition(ch);
```

Set the destination of the transition to the `Stateflow.State` object `st`.

```
dt.Destination = st;
dt.DestinationClock = 0;
```

Place the source endpoint for the transition 30 pixels above the destination endpoint. Place the midpoint for the transition 15 pixels above the destination endpoint.

```
dt.SourceEndPoint = dt.DestinationEndPoint-[0 30];
dt.MidPoint = dt.DestinationEndPoint-[0 15];
```



Add Supertransition from Subchart

Create a supertransition that connects a junction inside a subchart to a junction outside the subchart.



Open the model and access the `Stateflow.Chart` object for the chart.

```
open_system("sfSupertransitionAPIExample")
ch = find(sfroot, "-isa", "Stateflow.Chart");
```

Access the `Stateflow.State` object for the subchart and the `Stateflow.Junction` objects for the junctions.

```
st = find(ch, "-isa", "Stateflow.State");
j1 = find(st, "-isa", "Stateflow.Junction");
j2 = find(ch, "-isa", "Stateflow.Junction", "-depth", 1);
```

Save the original position of the subchart to a temporary workspace variable `subchartPosition`.

```
subchartPosition = st.Position;
```

Convert the subchart to a normal state by setting its `IsSubchart` and `IsGrouped` properties to `false`.

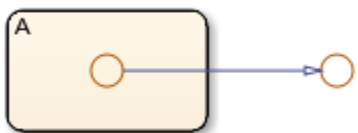
```
st.IsSubchart = false;
st.IsGrouped = false;
```

When you convert a subchart to a normal state, it may change size to display its contents.



Add a transition that connects junction `j1` to junction `j2`.

```
tr = Stateflow.Transition(ch);
tr.Source = j1;
tr.Destination = j2;
```



Revert the state to a subchart by setting its `IsSubchart` property to `true`. Restore the subchart to its original position.

```
st.IsSubchart = true;
st.Position = subchartPosition;
```



The transition between the junctions is now a supertransition that crosses the boundary of the subchart.

Version History

Introduced before R2006a

R2023a: New object function and property

Errors starting in R2023a

Use the object function `commentedBy` and the property `IsCommented` to investigate commented-out transitions:

- The object function `commentedBy` identifies the explicitly commented objects that cause a transition to be commented out.
- The property `IsCommented` indicates whether a transition is commented out. This property replaces the object function `isCommented`.

See Also

[Stateflow.AtomicSubchart](#) | [Stateflow.Chart](#) | [Stateflow.Function](#) | [Stateflow.Junction](#) | [Stateflow.SimulinkBasedState](#) | [Stateflow.State](#)

Topics

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

“Transition Between Operating Modes”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.TruthTable

Truth table function in chart, state, box, or function

Description

Use `Stateflow.TruthTable` objects to create truth table functions that implement combinatorial logic design in a tabular format. You can use truth table functions to model decision making for fault detection and management and mode switching. For more information, see “Use Truth Tables to Model Combinatorial Logic”.

Creation

Syntax

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

Description

`function = Stateflow.TruthTable(parent)` creates a `Stateflow.TruthTable` object in a parent chart, state, box, or function.

Input Arguments

parent — Parent for new truth table

`Stateflow.Chart` object | `Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object

Parent for the new truth table, specified as a Stateflow API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.Function`
- `Stateflow.State`

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Name — Name of truth table

`" "` (default) | string scalar | character vector

Name of the truth table, specified as a string scalar or character vector.

LabelString — Label for truth table

"?" (default) | string scalar | character vector

Label for the truth table, specified as a string scalar or character vector.

ActionTable — Action table

cell array of character vectors

Action table for the truth table, specified as a cell array of character vectors.

ConditionTable — Condition table

cell array of character vectors

Condition table for the truth table, specified as a cell array of character vectors.

Language — Action language

"MATLAB" (default) | "C"

Action language used to program the truth table, specified as "MATLAB" or "C". The option "C" is supported only in truth tables in charts that use C as the action language. For more information, see "Differences Between MATLAB and C as Action Language Syntax".

IsExplicitlyCommented — Whether to comment out truth table

false or 0 (default) | true or 1

Whether to comment out the truth table, specified as a numeric or logical 1 (`true`) or 0 (`false`). Setting this property to `true` is equivalent to right-clicking the truth table and selecting **Comment Out**. For more information, see "Comment Out Objects in a Stateflow Chart".

IsImplicitlyCommented — Whether truth table is implicitly commented out

true or 1 | false or 0

This property is read-only.

Whether the truth table is implicitly commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). The truth table is implicitly commented out when you explicitly comment out an object that contains it. If the truth table is contained in an atomic subchart or an atomic box, this property is `false` unless the explicitly commented object is also contained in the atomic subchart or atomic box.

IsCommented — Whether truth table is commented out


true or 1 | false or 0

This property is read-only.

Whether the truth table is commented out, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property is `true` when either `IsExplicitlyCommented` or `IsImplicitlyCommented` is `true`.

CommentText — Comment text

"" (default) | string scalar | character vector

Comment text added to the truth table, specified as a string scalar or character vector. This property applies only when the `IsExplicitlyCommented` property is `true`. In the Stateflow Editor, when you point to the comment badge  on the truth table, the text appears as a tooltip. When you set the `IsExplicitlyCommented` property to `false`, the value of `CommentText` reverts to "".

Graphical Appearance

Position — Position and size of truth table

[0 0 90 60] (default) | [left top width height]

Position and size of the truth table, specified as a four-element numeric vector of the form [left top width height].

BadIntersection — Whether truth table intersects a box, state, or function

true or 1 | false or 0

This property is read-only.

Whether the truth table graphically intersects a box, state, or function, specified as a numeric or logical 1 (true) or 0 (false).

FontSize — Font size for truth table label

scalar

Font size for the truth table label, specified as a scalar. The `StateFont.Size` property of the chart that contains the truth table sets the initial value of this property.

Debugging

OverSpecDiagnostic — Level of diagnostic when truth table is overspecified

"Error" (default) | "Warning" | "None"

Level of diagnostic action when the truth table is overspecified, specified as "Error", "Warning", or "None". For more information, see "Correct Overspecified and Underspecified Truth Tables".

UnderSpecDiagnostic — Level of diagnostic when truth table is underspecified

"Error" (default) | "Warning" | "None"

Level of diagnostic action when the truth table is underspecified, specified as "Error", "Warning", or "None". For more information, see "Correct Overspecified and Underspecified Truth Tables".

Debug — Debugger properties

`Stateflow.FunctionDebug` object

Debugger properties for the truth table, specified as a `Stateflow.FunctionDebug` object with this property:

- **Breakpoints.OnDuring** — Whether to set the During Function Call breakpoint, specified as a numeric or logical 1 (true) or 0 (false).

This property applies only when both the `Language` property of the truth table and the `ActionLanguage` of the chart that contains the truth table are "C". For more information, see "Set Breakpoints to Debug Charts".

Example: `function.Debug.Breakpoints.OnDuring = true;`

Integer and Fixed-Point Data

SaturateOnIntegerOverflow — Whether data saturates on integer overflow

true or 1 (default) | false or 0

Whether the data in the truth table saturates on integer overflow, specified as a numeric or logical 1 (`true`) or 0 (`false`). When this property is disabled, the data in the truth table wraps on integer overflow. This property applies only when the `Language` property of the truth table is "MATLAB" and the `ActionLanguage` of the chart that contains the truth table is "C". For more information, see "Handle Integer Overflow for Chart Data".

EmlDefaultFimath — Default fimath properties

"Same as MATLAB Default" (default) | "Other:UserSpecified"

Default `fimath` properties for the truth table, specified as one of these values:

- "Same as MATLAB Default" — Use the same `fimath` properties as the current default `fimath` object.
- "Other:UserSpecified" — Use the `InputFimath` property to specify the default `fimath` object.

This property applies only when the `Language` property of the truth table is "MATLAB" and the `ActionLanguage` of the chart that contains the truth table is "C".

InputFimath — Default fimath object

string scalar | character vector

Default `fimath` object, specified as a string scalar or character vector. When the `EmlDefaultFimath` property for the truth table is "Other:UserSpecified", you can use this property to:

- 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 `Language` property of the truth table is "MATLAB" and the `ActionLanguage` of the chart that contains the truth table is "C".

Code Generation

InlineOption — Appearance in generated code

"Auto" (default) | "Function" | "Inline"

Appearance of the truth table in generated code, specified as one of these values:

- "Auto" — An internal calculation determines the appearance of the truth table in generated code.
- "Function" — The truth table is implemented as a separate C function.
- "Inline" — Calls to the truth table are replaced by code as long as the truth table is not part of a recursion.

For more information, see "Inline State Functions in Generated Code" (Simulink Coder).

Hierarchy

Chart — Chart that contains truth table

Stateflow.Chart object

This property is read-only.

Chart that contains the truth table, specified as a `Stateflow.Chart` object.

Subviewer — Subviewer for truth table

Stateflow.Chart object | Stateflow.State object | Stateflow.Box object | Stateflow.Function object

This property is read-only.

Subviewer for the truth table, specified as a Stateflow.Chart, Stateflow.State, Stateflow.Box, or Stateflow.Function object. The subviewer is the chart or subchart where you can graphically view the truth table.

Machine — Machine that contains truth table

Stateflow.Machine object

This property is read-only.

Machine that contains the truth table, specified as a Stateflow.Machine object.

Path — Location of parent in model hierarchy

character vector

This property is read-only.

Location of the parent of the truth table in the model hierarchy, specified as a character vector.

Identification**Description — Description**

"" (default) | string scalar | character vector

Description for the truth table, specified as a string scalar or character vector.

Document — Document link

"" (default) | string scalar | character vector

Document link for the truth table, specified as a string scalar or character vector.

Tag — User-defined tag

[] (default) | any data type

User-defined tag for the truth table, specified as data of any type.

SSIdNumber — Session-independent identifier

scalar

This property is read-only.

Session-independent identifier, specified as an integer scalar. Use this property to distinguish the truth table from other objects in the model.

Id — Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Unlike `SSIdNumber`, the value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>find</code>	Identify specified objects in hierarchy
<code>getChildren</code>	Identify children of object
<code>getParent</code>	Identify parent of object
<code>getReferences</code>	Identify references to symbol name
<code>renameReferences</code>	Rename symbol and update references to symbol name
<code>commentedBy</code>	Identify objects that implicitly comment out a graphical object
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment
<code>highlight</code>	Highlight graphical object
<code>fitToView</code>	Zoom in on graphical object

Examples

Add Truth Table Function to Chart

Add a truth table function in the chart `ch`. Set its label to "`[y1,y2] = f(x1,x2,x3)`".

```
function = Stateflow.TruthTable(ch);
function.LabelString = "[y1,y2] = f(x1,x2,x3)";
```

Version History

Introduced before R2006a

R2023a: New object functions and properties

Errors starting in R2023a

`Stateflow.TruthTable` objects have new object functions and properties:

- The object function `getReferences` returns the locations where a chart refers to the name of a truth table function.
- The object function `renameReferences` renames a truth table function and updates all references to the function name in the chart.
- The object function `commentedBy` identifies the explicitly commented objects that cause a truth table function to be commented out.
- The property `IsCommented` indicates whether a truth table function is commented out. This property replaces the object function `isCommented`.

See Also

`Stateflow.Box` | `Stateflow.Chart` | `Stateflow.Function` | `Stateflow.State`

Topics

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

“Use Truth Tables to Model Combinatorial Logic”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.TruthTableChart

Tabular representation of state machine for decision logic

Description

Use `Stateflow.TruthTableChart` objects to implement combinatorial logic design in a tabular format. You can use Truth Table blocks to model decision making for fault detection and management and mode switching. For more information, see “Use Truth Tables to Model Combinatorial Logic”.

Creation

To create a `Stateflow.TruthTableChart` object, call the function `sfnew` with the `-TT` argument. For example, to create a Truth Table block in a new Simulink model called `myModel`, enter:

```
sfnew -TT myModel
```

Alternatively, you can add a new Truth Table block to an existing model by using the function `add_block`:

```
add_block("sflib/Truth Table", ...
    "myModel/Truth Table")
```

Then, to access the `Stateflow.TruthTableChart` object, call the `find` function for the `Simulink.Root` object:

```
table = find(sfroot, "-isa", "Stateflow.TruthTableChart", ...
    Path="myModel/Truth Table");
```

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Content

Name — Name of truth table

"Truth Table" (default) | string scalar | character vector

Name of the truth table, specified as a string scalar or character vector.

ActionTable — Action table

cell array of character vectors

Action table for the truth table, specified as a cell array of character vectors.

ConditionTable — Condition table

cell array of character vectors

Condition table for the truth table, specified as a cell array of character vectors.

SupportVariableSizing — Whether truth table supports variable-size data

true or 1 (default) | false or 0

Whether the truth table supports variable-size data, specified as a numeric or logical 1 (true) or 0 (false). For more information, see “Declare Variable-Size Data in Stateflow Charts”.

Discrete and Continuous-Time Semantics**ChartUpdate — Activation method for truth table**

"INHERITED" (default) | "CONTINUOUS" | "DISCRETE"

Activation method for the truth table, specified as "CONTINUOUS", "DISCRETE", or "INHERITED". For more information, see “Update Method”.

SampleTime — Sample time for activating truth table

"-1" (default) | string scalar | character vector

Sample time for activating the truth table, specified as a string scalar or character vector. This property applies only when the ChartUpdate property for the truth table is "DISCRETE".

Integer and Fixed-Point Data**SaturateOnIntegerOverflow — Whether data saturates on integer overflow**

true or 1 (default) | false or 0

Whether the data in the truth table saturates on integer overflow, specified as a numeric or logical 1 (true) or 0 (false). When this property is disabled, the data in the truth table wraps on integer overflow. For more information, see “Handle Integer Overflow for Chart Data”.

TreatAsFi — Inherited Simulink signals to treat as fi objects

"Fixed-point" (default) | "Fixed-point & Integer"

Inherited Simulink signals to treat as Fixed-Point Designer fi objects, specified as one of these values:

- "Fixed-point" — The truth table treats all fixed-point inputs as fi objects.
- "Fixed-point & Integer" — The truth table treats all fixed-point and integer inputs as fi objects.

Em1DefaultFimath — Default fimath properties

"Same as MATLAB Default" (default) | "Other:UserSpecified"

Default fimath properties for the truth table, specified as one of these values:

- "Same as MATLAB Default" — Use the same fimath properties as the current default fimath object.
- "Other:UserSpecified" — Use the InputFimath property to specify the default fimath object.

InputFimath — Default fimath object

string scalar | character vector

Default `fimath` object, specified as a string scalar or character vector. When the `EmfDefaultFimath` property for the truth table is "Other:UserSpecified", you can use this property to:

- Enter an expression that constructs a `fimath` object.
- Enter the variable name for a `fimath` object in the MATLAB or model workspace.

Debugging

OverSpecDiagnostic — Level of diagnostic when truth table is overspecified

"Error" (default) | "Warning" | "None"

Level of diagnostic action when the truth table is overspecified, specified as "Error", "Warning", or "None". For more information, see "Correct Overspecified and Underspecified Truth Tables".

UnderSpecDiagnostic — Level of diagnostic when truth table is underspecified

"Error" (default) | "Warning" | "None"

Level of diagnostic action when the truth table is underspecified, specified as "Error", "Warning", or "None". For more information, see "Correct Overspecified and Underspecified Truth Tables".

Hierarchy

Machine — Machine that contains truth table

`Stateflow.Machine` object

This property is read-only.

Machine that contains the truth table, specified as a `Stateflow.Machine` object.

Path — Location of truth table in model hierarchy

character vector

This property is read-only.

Location of the truth table in the model hierarchy, specified as a character vector.

Dirty — Whether truth table has changed

true or 1 | false or 0

Whether the truth table has changed after being opened or saved, specified as a numeric or logical 1 (true) or 0 (false).

Locked — Whether truth table is locked

false or 0 (default) | true or 1

Whether the truth table is locked, specified as a numeric or logical 1 (true) or 0 (false). Enable this property to prevent changes in the truth table.

Iced — Whether truth table is locked

false or 0 (default) | true or 1

This property is read-only.

Whether the truth table is locked, specified as a numeric or logical 1 (`true`) or 0 (`false`). This property is equivalent to the property `Locked`, but is used internally to prevent changes in the truth table during simulation.

Identification

Description – Description

"" (default) | string scalar | character vector

Description for the truth table, specified as a string scalar or character vector.

Document – Document link

"" (default) | string scalar | character vector

Document link for the truth table, specified as a string scalar or character vector.

Tag – User-defined tag

[] (default) | any data type

User-defined tag for the truth table, specified as data of any type.

Id – Unique identifier

scalar

This property is read-only.

Unique identifier, specified as an integer scalar. Use this property to distinguish the truth table from other objects in the model. The value of this property is reassigned every time you start a new MATLAB session and may be recycled after an object is deleted.

Object Functions

<code>find</code>	Identify specified objects in hierarchy
<code>getChildren</code>	Identify children of object
<code>dialog</code>	Open properties dialog box
<code>view</code>	Display object in editing environment

Examples

Create Empty Truth Table

Call the function `sfnew` with the `-TT` argument to open a new Simulink model that contains an empty Truth Table block.

```
sfnew -TT
```

Access the `Simulink.Root` object by calling the `sfroot` function.

```
rt = sfroot;
```

Access the `Stateflow.TruthTableChart` object by calling the `find` function for the `Simulink.Root` object.


```
table = find(rt, "-isa", "Stateflow.TruthTableChart");
```

Version History

Introduced before R2006a

See Also

Blocks

Truth Table

Functions

sfnew | sfroot | add_block

Topics

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

“Model Finite State Machines by Using Stateflow Charts”

“Use Truth Tables to Model Combinatorial Logic”

“Summary of Stateflow API Objects and Properties” on page 1-36

Stateflow.Unit

Unit of measurement for input and output data

Description

Use a `Stateflow.Unit` object to specify the unit of measurement for an input or output data object. For more information, see “Specify Units for Stateflow Data”.

Creation

Each data object and message has its own `Stateflow.Unit` object. However, the object only applies for `Stateflow.Data` objects when the `Scope` property is set to "Input" or "Output". To access the `Stateflow.Unit` object, use the `Props.Unit` property for the `Stateflow.Data` object.

Properties

Stateflow API objects have properties that correspond to the values you set in the Stateflow Editor. To access or modify a property, use dot notation. To access or modify multiple properties for multiple API objects, use the `get` and `set` functions, respectively. For more information, see “Modify Properties and Call Functions of Stateflow Objects” on page 1-11.

Name — Name of unit of measurement

"inherit" (default) | string scalar | character vector

Name of unit of measurement, specified as a string scalar or character vector. This property applies only to input and output data.

Examples

Specify Units for Data

Access the `Stateflow.Props` and `Stateflow.Unit` objects for the `Stateflow.Data` object `x`.

```
properties = x.Props;  
unit = properties.Unit;
```

Specify the units as meters.

```
unit.Name = "m";
```

Version History

Introduced before R2006a

See Also

`Stateflow.Data`

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Specify Units for Stateflow Data”

API Object Function Reference

clearMappingForSymbol

Package: Stateflow

Clear mapping for symbol in atomic subchart, atomic box, or Simulink based state

Syntax

```
clearMappingForSymbol(subsystem, subsystemSymbol)
```

Description

`clearMappingForSymbol(subsystem, subsystemSymbol)` clears the mapping for the subsystem symbol `subsystemSymbol`, where `subsystem` is an atomic subchart, atomic box, or Simulink based state. After you clear the mapping, the subsystem symbol maps to a main chart symbol with the same name. For more information, see “Map Variables for Atomic Subcharts and Boxes” and “Map Variables for Simulink Based States”.

Examples

Map Variables in Atomic Subchart

In an atomic subchart called A, modify the mapping for the subchart input u1.

Open the model `sf_atomic_io_data_fixed.slx`.

```
open_system("sf_atomic_io_data_fixed")
```

Access the `Stateflow.AtomicSubchart` object for the atomic subchart A.

```
subsystem = find(sfroot, "-isa", "Stateflow.AtomicSubchart", ...  
    Name="A");
```

Use the `Subchart` property to access the `Stateflow.Data` object for subchart input u1.

```
subsystemSymbol = find(subsystem.Subchart, ...  
    "-isa", "Stateflow.Data", Name="u1");
```

Use the `Chart` property to access the `Stateflow.Data` object for chart input u2.

```
chartSymbol = find(subsystem.Chart, ...  
    "-isa", "Stateflow.Data", Name="u2");
```

Check the mapping for subchart input u1.

```
getMappingForSymbol(subsystem, subsystemSymbol).Name  
  
ans =  
'u1'
```

Map subchart input u1 to chart input u2.

```
setMappingForSymbol(subsystem, subsystemSymbol, chartSymbol)
getMappingForSymbol(subsystem, subsystemSymbol).Name
```

```
ans =
'u2'
```

Clear the mapping for subchart input u1.

```
clearMappingForSymbol(subsystem, subsystemSymbol)
getMappingForSymbol(subsystem, subsystemSymbol).Name
```

```
ans =
'u1'
```

Map Variables in Simulink Based State

In a Simulink based state called Locked, modify the mapping for the output we.

Open the model sf_clutch.slx.

```
open_system("sf_clutch.slx")
```

Access the Stateflow.SimulinkBasedState object for the Simulink based state Locked.

```
subsystem = find(sfroot, "-isa", "Stateflow.SimulinkBasedState", ...
    Name="Locked");
```

Check the mapping for Simulink based state output we.

```
getMappingForSymbol(subsystem, "we").Name
```

```
ans =
'we'
```

Map the Simulink based state output we to chart output wv.

```
setMappingForSymbol(subsystem, "we", "wv")
getMappingForSymbol(subsystem, "we").Name
```

```
ans =
'wv'
```

Clear the mapping for Simulink based state output we.

```
clearMappingForSymbol(subsystem, "we")
getMappingForSymbol(subsystem, "we").Name
```

```
ans =
'we'
```

Input Arguments

subsystem — Atomic subchart, atomic box, or Simulink based state

Stateflow.AtomicSubchart object | Stateflow.AtomicBox object |
Stateflow.SimulinkBasedState object

Atomic subchart, atomic box, or Simulink based state, specified as a `Stateflow.AtomicSubchart`, `Stateflow.AtomicBox`, or `Stateflow.SimulinkBasedState` object.

subsystemSymbol — Subsystem symbol

`Stateflow.Data` object | `Stateflow.Event` object | string scalar | character vector

Subsystem symbol, specified as a `Stateflow.Data` object, a `Stateflow.Event` object, a string scalar, or a character vector.

Note If the `subsystem` argument is a `Stateflow.SimulinkBasedState` object, this argument must be a string scalar or character vector.

Version History

Introduced in R2022b

R2023a: Map variables for Simulink based states

Edit the mapping of symbols in Simulink based states by calling the object functions `getMappingForSymbol`, `setMappingForSymbol`, and `clearMappingForSymbol` on `Stateflow.SimulinkBasedState` objects.

See Also

Functions

`disableMappingForSymbol` | `getMappingForSymbol` | `setMappingForSymbol`

Objects

`Stateflow.AtomicBox` | `Stateflow.AtomicSubchart` | `Stateflow.Data` | `Stateflow.Event` | `Stateflow.SimulinkBasedState`

Topics

“Map Variables for Atomic Subcharts and Boxes”

“Map Variables for Simulink Based States”

commentedBy

Package: Stateflow

Identify objects that implicitly comment out a graphical object

Syntax

```
objArray = commentedBy(graphicalObject)
```

Description

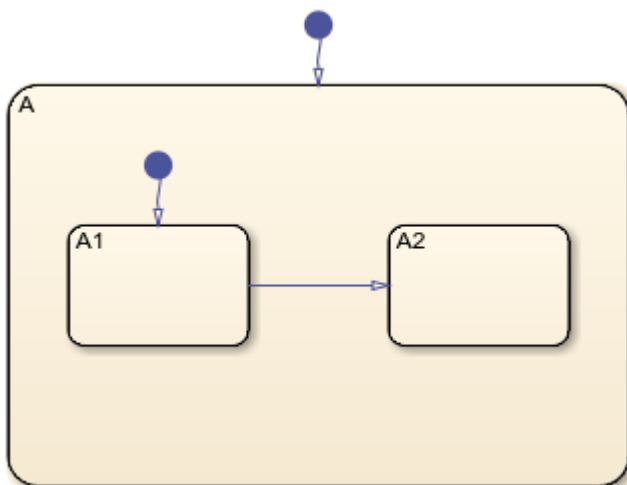
`objArray = commentedBy(graphicalObject)` returns an array of the explicitly commented objects that cause the graphical object `graphicalObject` to be commented out. The graphical object is commented out when:

- The `IsExplicitlyCommented` property has a value of `true`. In this case, `objArray` includes the graphical object.
- The `IsImplicitlyCommented` property has a value of `true`. In this case, `objArray` includes the explicitly commented states, boxes, or functions that contain the object. Additionally,
 - If `graphicalObject` is a transition, `objArray` includes the explicitly commented objects that cause the source or destination to be commented out.
 - If `graphicalObject` is an entry or exit port, `objArray` includes the explicitly commented objects that cause the matching entry or exit junction to be commented out.

Examples

Comment Out State

When you explicitly comment out a state, box, or function, you implicitly comment out all the graphical objects that it contains. For example, when you comment out state A in this chart, you also comment out its substates, A1 and A2.



Open the model.

```
open_system("sfHierarchyAPIExample")
```

Find the Stateflow.State objects named A, A1, and A2.

```
sA = find(sfroot, "-isa", "Stateflow.State", Name="A");
sA1 = find(sfroot, "-isa", "Stateflow.State", Name="A1");
sA2 = find(sfroot, "-isa", "Stateflow.State", Name="A2");
```

Check that state A and its substates are not commented out.

```
get([sA sA1 sA2], ...
    {"Name", "isCommented", "isExplicitlyCommented", "isImplicitlyCommented"})
```

```
ans=3x4 cell array
    {'A' }      {[0]}      {[0]}      {[0]}
    {'A1' }     {[0]}      {[0]}      {[0]}
    {'A2' }     {[0]}      {[0]}      {[0]}
```

Explicitly comment out states A and A2.

```
sA.IsExplicitlyCommented = true;
sA2.IsExplicitlyCommented = true;
```

Verify that state A and its substates are commented out.

```
get([sA sA1 sA2], ...
    {"Name", "isCommented", "isExplicitlyCommented", "isImplicitlyCommented"})
```

```
ans=3x4 cell array
    {'A' }      {[1]}      {[1]}      {[0]}
    {'A1' }     {[1]}      {[0]}      {[1]}
    {'A2' }     {[1]}      {[1]}      {[1]}
```

Identify the explicitly commented objects that cause each state to be commented out.

```
get(commentedBy(sA), {"Name"})
```

```
ans = 1x1 cell array
    {'A' }
```

```
get(commentedBy(sA1), {"Name"})
```

```
ans = 1x1 cell array
    {'A' }
```

```
get(commentedBy(sA2), {"Name"})
```

```
ans = 2x1 cell
    {'A' }
    {'A2' }
```

Input Arguments

graphicalObject — Graphical object

Stateflow.State object | Stateflow.Box object | Stateflow.Function object | ...

Graphical object, specified as a Stateflow API object of one of these types:

- Stateflow.AtomicBox
- Stateflow.AtomicSubchart
- Stateflow.Box
- Stateflow.EMFunction
- Stateflow.Function
- Stateflow.Junction
- Stateflow.Port
- Stateflow.SimulinkBasedState
- Stateflow.SLFunction
- Stateflow.State
- Stateflow.Transition
- Stateflow.TruthTable

Limitations

- When a graphical object is contained in an atomic subchart or an atomic box, the `commentedBy` function returns only explicitly commented objects that are also contained in the subchart or box.

Version History

Introduced in R2023a

See Also

Functions

`find`

Objects

Stateflow.State | Stateflow.Box | Stateflow.Function

Topics

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

“Comment Out Objects in a Stateflow Chart”

“Summary of Stateflow API Objects and Properties” on page 1-36

“Modify Properties and Call Functions of Stateflow Objects” on page 1-11

copy

Package: Stateflow

Copy array of objects to clipboard

Syntax

```
copy(clipboard,objArray)
```

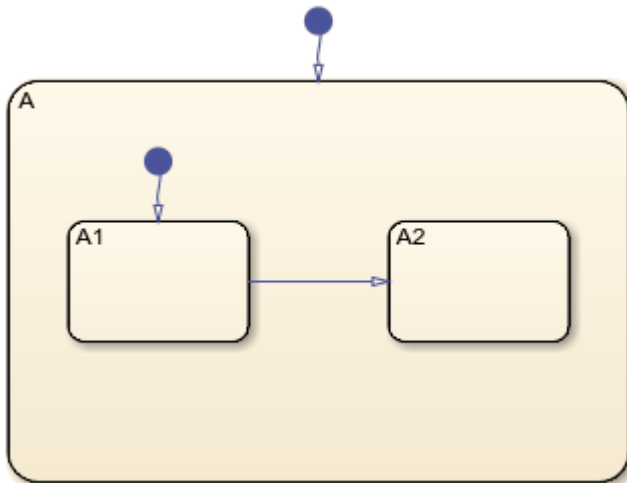
Description

`copy(clipboard,objArray)` copies the objects in the array `objArray` to the clipboard. To paste the copied objects, use the `pasteTo` function.

Examples

Copy and Paste by Grouping

Group a state and copy its contents to the chart. When you group a state, box, or graphical function, you can copy and paste all the objects contained in the grouped object, as well as all the relationships among these objects. This method is the simplest way of copying and pasting objects programmatically. If a state is not grouped, copying the state does not copy any of its contents.



Open the model and access the `Stateflow.Chart` object for the chart.

```
open_system("sfHierarchyAPIExample")
ch = find(sfroot, "-isa", "Stateflow.Chart");
```

Find the `Stateflow.State` object named A.

```
sA = find(ch, "-isa", "Stateflow.State", Name="A");
```

Group state A and its contents by setting the `IsGrouped` property for `sA` to `true`. Save the previous setting of this property so you can revert to it later.

```
prevGrouping = sA.IsGrouped;
sA.IsGrouped = true;
```

Change the name of the state to `Copy_of_A`. Save the previous name so you can revert to it later.

```
prevName = sA.Name;
newName = "Copy_of_" + prevName;
sA.Name = newName;
```

Access the clipboard object.

```
cb = sfclipboard;
```

Copy the grouped state to the clipboard.

```
copy(cb, sA);
```

Restore the state properties to their original settings.

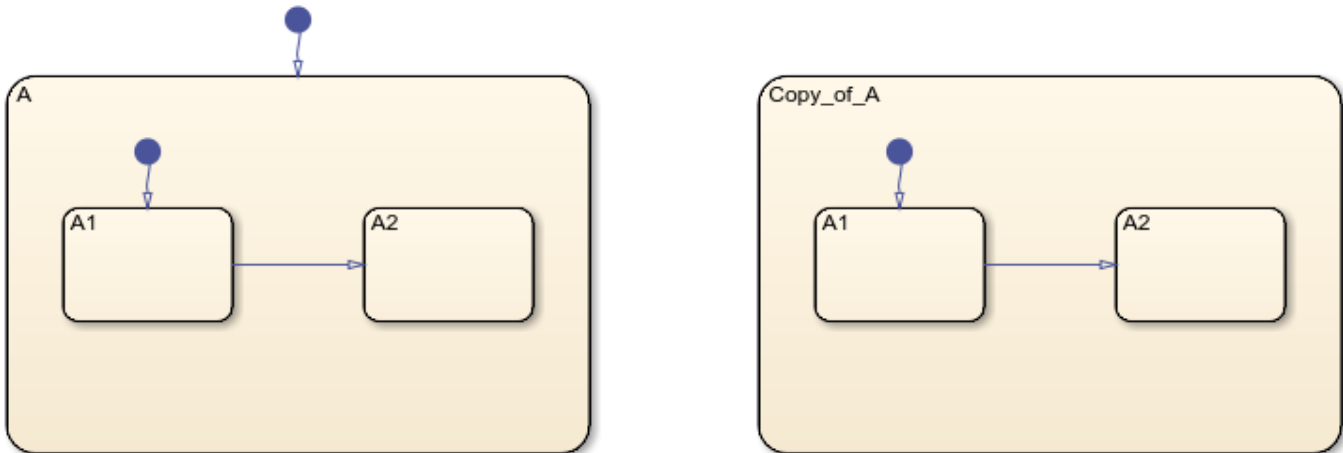
```
sA.IsGrouped = prevGrouping;
sA.Name = prevName;
```

Paste a copy of the objects from the clipboard to the chart.

```
pasteTo(cb, ch);
```

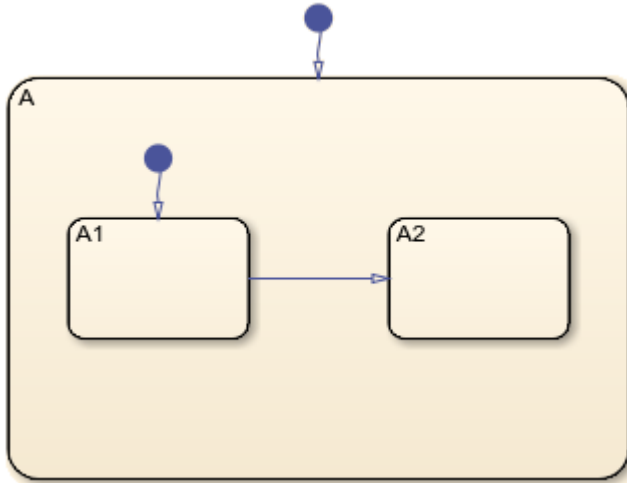
Adjust the state properties of the new state.

```
sNew = find(ch, "-isa", "Stateflow.State", Name=newName);
sNew.Position = sA.Position + [400 0 0 0];
sNew.IsGrouped = prevGrouping;
```



Copy and Paste Array of Objects

Copy states A1 and A2, along with the transition between them, to a new state in the chart. To preserve transition connections and containment relationships between objects, copy all the connected objects at once.



Open the model and access the `Stateflow.Chart` object for the chart.

```
open_system("sfHierarchyAPIExample")
ch = find(sfroot, "-isa", "Stateflow.Chart");
```

Find the `Stateflow.State` object named A.

```
sA = find(ch, "-isa", "Stateflow.State", Name="A");
```

Add a new state called B. To enable pasting of other objects inside B, convert the new state to a subchart.

```
sB = Stateflow.State(ch);
sB.Name = "B";
sB.Position = sA.Position + [400 0 0 0];
sB.IsSubchart = true;
```

Create an array called `objArray` that contains the states and transitions in state A. Use the function `setdiff` to remove state A from the array of objects to copy.

```
objArrayS = find(sA, "-isa", "Stateflow.State");
objArrayS = setdiff(objArrayS, sA);
objArrayT = find(sA, "-isa", "Stateflow.Transition");
objArray = [objArrayS objArrayT];
```

Access the clipboard object.

```
cb = sfclipboard;
```

Copy the objects in `objArray` and paste them in subchart B.

```
copy(cb, objArray);
pasteTo(cb, sB);
```

Revert B to a state.

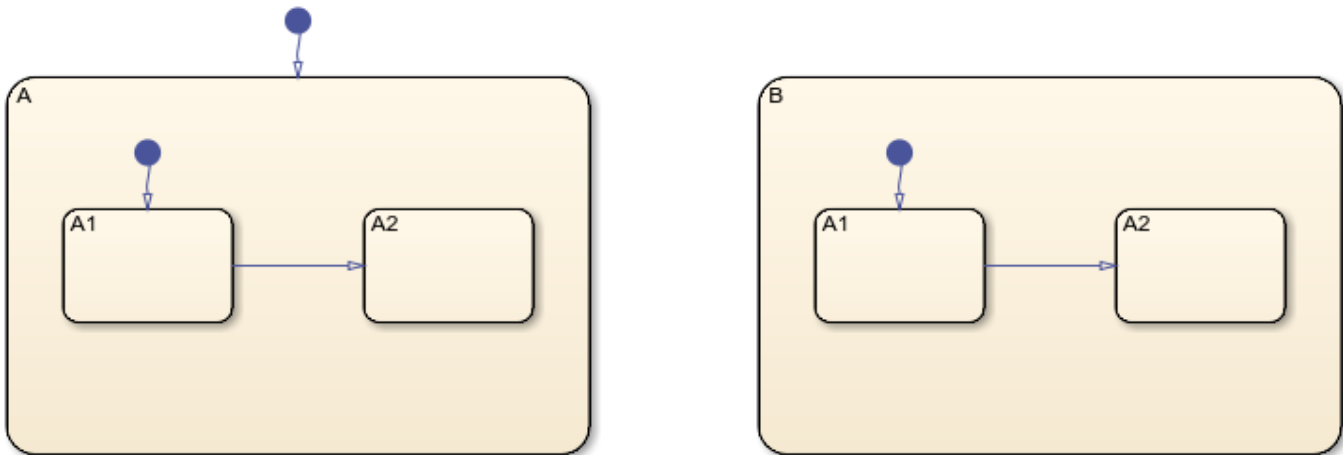
```
sB.IsSubchart = false;
sB.IsGrouped = false;
```

Reposition the states and transitions in B.

```
newStates = find(sB, "-isa", "Stateflow.State");
newStates = setdiff(newStates, sB);

newTransitions = find(sB, "-isa", "Stateflow.Transition");
newOClocks = get(newTransitions, {"SourceClock", "DestinationClock"});

for i = 1:numel(newStates)
    newStates(i).Position = newStates(i).Position + [25 35 0 0];
end
set(newTransitions, {"SourceClock", "DestinationClock"}, newOClocks);
```



Input Arguments

clipboard – Clipboard

Stateflow.Clipboard object

Clipboard, specified as a Stateflow.Clipboard object.

objArray – Objects to copy

array of Stateflow objects

Objects to copy, specified as an array of Stateflow API objects. The array must contain only graphical objects or only nongraphical objects.

Graphical objects include:

- Stateflow.Annotation
- Stateflow.AtomicBox
- Stateflow.AtomicSubchart

- `Stateflow.Box`
- `Stateflow.EMFunction`
- `Stateflow.Function`
- `Stateflow.Junction`
- `Stateflow.SimulinkBasedState`
- `Stateflow.SLFunction`
- `Stateflow.State`
- `Stateflow.Transition`
- `Stateflow.TruthTable`

Nongraphical objects include:

- `Stateflow.Data`
- `Stateflow.Event`
- `Stateflow.Message`

Copying graphical objects also copies the `Stateflow.Data`, `Stateflow.Event`, and `Stateflow.Message` objects that the graphical objects contain. When you copy multiple graphical objects, the value of their `Subviewer` property must be the same.

Tips

To maintain the transition connections and containment relationships between copied objects, you must:

- Copy a grouped object to the clipboard. When you group a state, box, or graphical function, you can copy and paste all the objects contained in the grouped object, as well as all the relationships among these objects. For more information, see “Copy and Paste by Grouping” on page 3-8.
- Copy all the related objects. For example, to copy two states connected by a transition to another container, create an array that contains both the states and the transition. Then you can copy the array to the clipboard. For more information, see “Copy and Paste Array of Objects” on page 3-9.

Version History

Introduced before R2006a

See Also

Functions

`find` | `pasteTo` | `setdiff` | `sfclipboard`

Objects

`Stateflow.State` | `Stateflow.Clipboard`

Topics

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

defaultTransitions

Package: Stateflow

Identify default transitions in specified object

Syntax

```
transitions = defaultTransitions(parent)
```

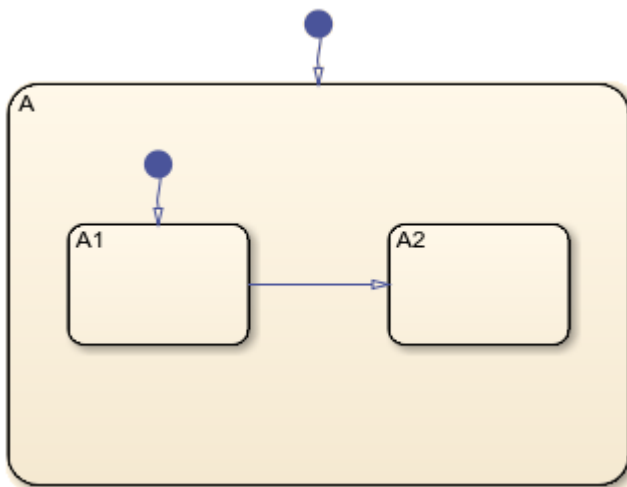
Description

`transitions = defaultTransitions(parent)` returns an array of `Stateflow.Transition` objects that correspond to the default transitions at the top level of the specified parent object. For more information, see “Use Default Transitions to Specify Initial Substate Activity”.

Examples

Identify Default Transitions

Find the default transitions in this chart and in state A.



Open the model and access the `Stateflow.Chart` object for the chart.

```
open_system("sfHierarchyAPIExample")
ch = find(sfroot, "-isa", "Stateflow.Chart");
```

Identify the default transition at the top level of the chart. Display the name of the destination.

```
tr1 = defaultTransitions(ch);
tr1.Destination.Name
```

```
ans =  
'A'
```

Save the `Stateflow.State` object that corresponds to state A.

```
state = tr1.Destination;
```

Identify the default transition at the top level of the state A. Display the name of the destination.

```
tr2 = defaultTransitions(state);  
tr2.Destination.Name
```

```
ans =  
'A1'
```

Input Arguments

parent — Parent object

`Stateflow.Chart` object | `Stateflow.Function` object | `Stateflow.State` object

Parent object, specified as a Stateflow API object of one of these types:

- `Stateflow.Chart`
- `Stateflow.Function`
- `Stateflow.State`

Tips

- To identify default transitions inside a `Stateflow.Box`, call the `defaultTransitions` function on the `Stateflow.Chart`, `Stateflow.Function`, or `Stateflow.State` object that contains the box.

Version History

Introduced before R2006a

See Also

Functions

`find` | `getChildren` | `innerTransitions` | `outerTransitions` | `sinkedTransitions` | `sourcedTransitions`

Objects

`Stateflow.Box` | `Stateflow.Chart` | `Stateflow.Function` | `Stateflow.State` | `Stateflow.Transition`

Topics

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

“Access Objects in Your Stateflow Chart” on page 1-6

“Use Default Transitions to Specify Initial Substate Activity”

“Group and Execute Transitions”

dialog

Package: Stateflow

Open properties dialog box

Syntax

```
dialog(object)
```

Description

`dialog(object)` opens the properties dialog box of an object.

Examples

Open Chart Properties Dialog Box

Open a Simulink model called `myModel`. Suppose that the model contains a Stateflow chart named `My Chart`.

```
open_system("myModel")
```

Find the chart named `My Chart`.

```
ch = find(sfroot,"-isa","Stateflow.Chart", ...  
    Name="My Chart");
```

Open the properties dialog box for the chart.

```
dialog(ch);
```

Input Arguments

object – Object to inspect

Stateflow.Chart object | Stateflow.State object | Stateflow.Box object |
Stateflow.Function object | ...

Object to inspect, specified as a Stateflow API object of one of these types:

- Stateflow.Annotation
- Stateflow.AtomicBox
- Stateflow.AtomicSubchart
- Stateflow.Box
- Stateflow.Chart
- Stateflow.Data
- Stateflow.EMChart

- Stateflow.EMFunction
- Stateflow.Event
- Stateflow.Function
- Stateflow.Junction
- Stateflow.Machine
- Stateflow.Message
- Stateflow.Port
- Stateflow.SimulinkBasedState
- Stateflow.SLFunction
- Stateflow.State
- Stateflow.StateTransitionTableChart
- Stateflow.Transition
- Stateflow.TruthTable
- Stateflow.TruthTableChart

Version History

Introduced before R2006a

See Also

[view](#) | [highlight](#) | [fitToView](#)

Topics

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

disableMappingForSymbol

Package: Stateflow

Disable input event in atomic subchart or box

Syntax

```
disableMappingForSymbol(subsystem, subsystemEvent)
```

Description

`disableMappingForSymbol(subsystem, subsystemEvent)` disables the input event `subsystemEvent` in the atomic subchart or atomic box `subsystem`. For more information, see “Map Input Events for an Atomic Subchart”.

Examples

Disable Input Event

In an atomic subchart called A, disable the input event E.

Access the `Stateflow.AtomicSubchart` object for the atomic subchart A.

```
subchart = find(sfroot, "-isa", "Stateflow.AtomicSubchart", Name="A");
```

Use the `Subchart` property to access the `Stateflow.Event` object for the input event E.

```
subsystemEvent = find(subchart.Subchart, ...
    "-isa", "Stateflow.Event", Name="E");
```

Check the mapping for input event E.

```
getMappingForSymbol(subchart, subsystemEvent).Name
```

```
ans =
```

```
    'E'
```

Disable input event E.

```
disableMappingForSymbol(subchart, subsystemEvent)
getMappingForSymbol(subchart, subsystemEvent)
```

```
ans =
```

```
    []
```

Input Arguments

subsystem — Atomic subchart or atomic box

`Stateflow.AtomicSubchart` object | `Stateflow.AtomicBox` object

Atomic subchart or atomic box, specified as a `Stateflow.AtomicSubchart` or `Stateflow.AtomicBox` object.

subsystemEvent — Input event in atomic subchart or atomic box

`Stateflow.Event` object | string scalar | character vector

Input event in atomic subchart or atomic box, specified as a `Stateflow.Event` object, a string scalar, or a character vector.

Version History

Introduced in R2022b

See Also

Functions

`clearMappingForSymbol` | `getMappingForSymbol` | `setMappingForSymbol`

Objects

`Stateflow.AtomicBox` | `Stateflow.AtomicSubchart` | `Stateflow.Event`

Topics

“Map Variables for Atomic Subcharts and Boxes”

exportAsStruct

Package: Stateflow

Export contents of state transition table as structure array

Syntax

```
structure = exportAsStruct(table)
structure = exportAsStruct(table,hierarchical)
```

Description

`structure = exportAsStruct(table)` returns a structure array that contains the contents of the state transition table `table`.

`structure = exportAsStruct(table,hierarchical)` specifies whether the function returns an array of structures or a hierarchy of structures based on the value of `hierarchical`.

Examples

Create Array of Structures

Export the contents of the state transition table in “Model Bang-Bang Controller by Using a State Transition Table” as an array of structures. This state transition table contains two top-level states and three substates.

Access `Stateflow.StateTransitionTableChart` for the state transition table.

```
table = find(sfroot,"-isa","Stateflow.StateTransitionTableChart");
```

Export the contents of the state transition table as an array of structures.

```
structure = exportAsStruct(table)
```

```
structure =
```

```
1x5 struct array with fields:
```

```
    rowText
    depth
    rowType
    isDefaultTransitionOwner
    isWhenState
    hasHistory
    isExpanded
    outlinedTransitionIdxs
    sfObjectInfo
    aslInfo
    decompositionInfo
```

View the contents of a top-level state.

```
structure(1)
```

```
ans =
```

```
struct with fields:
```

```
        rowText: {'Normal' {3×1 cell} {3×1 cell}}
        depth: 1
        rowType: 0
isDefaultTransitionOwner: 1
        isWhenState: 0
        hasHistory: 0
        isExpanded: 1
outlinedTransitionIdxs: [0 0 0]
        sfObjectInfo: [1×3 struct]
        aslInfo: [1×1 struct]
        decompositionInfo: [1×1 struct]
```

View the contents of a child state.

```
structure(4)
```

```
ans =
```

```
struct with fields:
```

```
        rowText: {'On-entry: boiler_cmd = 1;' {3×1 cell} {3×1 cell}}
        depth: 2
        rowType: 0
isDefaultTransitionOwner: 0
        isWhenState: 0
        hasHistory: 0
        isExpanded: 0
outlinedTransitionIdxs: [0 0 0]
        sfObjectInfo: [1×3 struct]
        aslInfo: [1×1 struct]
        decompositionInfo: [1×1 struct]
```

Create Hierarchy of Structures

Export the contents of the state transition table in “Model Bang-Bang Controller by Using a State Transition Table” as a hierarchy of structures. This state transition table contains two top-level states and three substates.

Access `Stateflow.StateTransitionTableChart` for the state transition table.

```
table = find(sfroot, "-isa", "Stateflow.StateTransitionTableChart");
```

Export the contents of the state transition table as a hierarchy of structures.

```
structure = exportAsStruct(table, true)
```

```
structure =
```

```
struct with fields:
```

```
        tableData: [1×2 struct]
```



```
columnWidths: [209 175 174]
tableId: 172
```

View the contents of a top-level state.

```
structure.tableData(1)
```

```
ans =
```

```
struct with fields:
```

```
stateLabel: 'Normal'
transitions: [1x2 struct]
rowHeights: {[30] [30] [30]}
hasHistory: 0
isDefaultTransitionOwner: 1
isWhenState: 0
isExpanded: 1
hasRequirements: 0
hasBreakpoints: 0
hasEnabledBreakpoints: 0
children: [1x3 struct]
rowType: 0
possibleDestinations: {' ' 'Alarm' '$NEXT' '$SELF' '% IGNORE %'}
decompositionInfo: [1x1 struct]
```

View the contents of a child state.

```
structure.tableData(1).children(3)
```

```
ans =
```

```
struct with fields:
```

```
stateLabel: 'On-entry: boiler_cmd = 1;'
transitions: [1x2 struct]
rowHeights: {[53] [30] [30]}
hasHistory: 0
isDefaultTransitionOwner: 0
isWhenState: 0
isExpanded: 0
hasRequirements: 0
hasBreakpoints: 0
hasEnabledBreakpoints: 0
children: []
rowType: 0
possibleDestinations: {' '$PREV' 'Off' 'Warmup' '$SELF' '% IGNORE %'}
decompositionInfo: [1x1 struct]
```

Input Arguments

table — State transition table

Stateflow.StateTransitionTableChart object

State transition table, specified as a Stateflow.StateTransitionTableChart object.

hierarchical — Whether to create hierarchy of structures

false or 0 (default) | true or 1

Whether to create a hierarchy of structures, specified as a numeric or logical 1 (`true`) or 0 (`false`). This argument determines the format of the output argument `structure`.

Output Arguments

structure – Contents of state transition table

structure array

Contents of state transition table, returned as a structure array. The format of `structure` depends on the input argument `hierarchical`:

- If `hierarchical` is `false`, `structure` is an array that contains a structure for each state in the state transition table. In each of these structures, the field `depth` indicates the level of each state in the hierarchy. See “Create Array of Structures” on page 3-19.
- If `hierarchical` is `true`, `structure` is a structure with the fields `tableData`, `columnWidths`, and `tableId`. `tableData` is an array that contains a structure for each top-level state in the state transition table. In each of these structures, the field `children` is an array that contains a structure for each substate. This pattern continues down the hierarchy of states in the state transition table. See “Create Hierarchy of Structures” on page 3-20.

Version History

Introduced in R2022b

See Also

`Stateflow.StateTransitionTableChart`

Topics

“Model Bang-Bang Controller by Using a State Transition Table”

find

Package: Stateflow

Identify specified objects in hierarchy

Syntax

```
objArray = find(location,propertyName,propertyValue)
objArray = find(location,"-regexp",propertyName,propertyValue)
```

```
objArray = find(location,"-isa",objectType)
objArray = find(location,"-depth",depth)
objArray = find(location,"-property",propertyName)
objArray = find(location,"-method",functionName)
objArray = find(location,"-function",function)
```

```
objArray = find(location,"-not",___)
objArray = find(location,___,logicalOp,___)
```

Description

`objArray = find(location,propertyName,propertyValue)` returns an array of Stateflow API objects in the hierarchy of `location` that have a property called `propertyName` with a value of `propertyValue`.

`objArray = find(location,"-regexp",propertyName,propertyValue)` returns the objects that have a property called `propertyName` with a value that matches the regular expression specified by `propertyValue`. For more information, see "Regular Expressions".

`objArray = find(location,"-isa",objectType)` returns the objects in the hierarchy of `location` that have the type specified by `objectType`.

`objArray = find(location,"-depth",depth)` returns the objects at or above the specified depth in the hierarchy of `location`.

`objArray = find(location,"-property",propertyName)` returns the objects that have a property with the specified name.

`objArray = find(location,"-method",functionName)` returns the objects that have an object function with the specified name.

`objArray = find(location,"-function",function)` returns the objects for which the specified function returns true.

`objArray = find(location,"-not",___)` returns the objects that do not match the specified search criterion. You can specify search criteria by using one of the previous syntaxes except "-depth" and "-regexp".

`objArray = find(location,___,logicalOp,___)` combines search criteria specified by using the previous syntaxes. Use one of these logical operations:

- "-and" — Results must match both search criteria.
- "-or" — Results must match at least one of the criteria.
- "-xor" — Results must match exactly one of the criteria.

When using multiple logical operations, -and has the highest precedence, while -or and -xor are right-associative. If no logical operation is specified, -and is assumed.

Examples

Find Objects Named A

Find the objects in the chart ch whose Name property is A.

```
namedIsA = find(ch, "Name", "A")
```

You can specify property names and values as `PropertyName=PropertyValue`, where `PropertyName` is not enclosed in quotes. This syntax must appear after other arguments that use quotes.

```
namedIsA = find(ch, Name="A")
```

Find Objects with Names That Start with Letter A

Find the objects in the chart ch whose Name property starts with the letter A.

```
nameStartsWithA = find(ch, "-regex", "Name", "^A")
```

You can specify property names and values as `PropertyName=PropertyValue`, where `PropertyName` is not enclosed in quotes. This syntax must appear after other arguments that use quotes.

```
nameStartsWithA = find(ch, "-regex", Name="^A")
```

Find States in Chart

Find the states in the chart ch.

```
states = find(ch, "-isa", "Stateflow.State")
```

Find States with Names That Start with Letter A

Find the states in the chart ch whose Name property starts with the letter A.

```
stateNameStartsWithA = find(ch, "-isa", "Stateflow.State", ...  
    "-and", "-regex", "Name", "^A")
```

You can specify property names and values as `PropertyName=PropertyValue`, where `PropertyName` is not enclosed in quotes. This syntax must appear after other arguments that use quotes.

```
stateNameStartsWithA = find(ch, "-isa", "Stateflow.State", ...
    "-and", "-regexp", Name="^A")
```

Find Objects in Top Two Levels of Chart

Find the objects in the top two levels of the hierarchy of the chart `ch`.

```
depthTwoObjects = find(ch, "-depth", 2)
```

Find Symbols in Chart

Find the symbols in the chart `ch`. Symbols include `Stateflow.Data`, `Stateflow.Event`, `Stateflow.Message`, and other non-chart objects that have a `Name` property.

```
symbols = find(ch, "-property", "Name", ...
    "-not", "-isa", "Stateflow.Chart")
```

Find Graphical Objects in Chart

Find the graphical objects in the chart `ch`. Graphical objects are non-chart objects that have an object function called `fitToView`.

```
graphicalObjects = find(ch, "-method", "fitToView", ...
    "-not", "-isa", "Stateflow.Chart")
```

Find Objects for Which Signal Logging Is Enabled

Find the objects in the chart `ch` for which signal logging is enabled. Signal logging is enabled when the subproperty `LoggingInfo.DataLogging` is `true`.

```
f = @(h) h.LoggingInfo.DataLogging;
signalLoggingOn = find(ch, "-function", f);
```

Input Arguments

Location — Location to search

`Simulink.Root` object | `Stateflow.Chart` object | `Stateflow.State` object | ...

Location to search, specified as a `Simulink.Root` object, a `Simulink.BlockDiagram` object, or a `Stateflow` API object of one of these types:

- `Stateflow.Box`
- `Stateflow.Chart`
- `Stateflow.EMChart`
- `Stateflow.EMFunction`
- `Stateflow.Function`

- `Stateflow.Machine`
- `Stateflow.State`
- `Stateflow.SLFunction`
- `Stateflow.StateTransitionTableChart`
- `Stateflow.TruthTable`
- `Stateflow.TruthTableChart`

propertyName — Name of property

string scalar | character vector

Name of property, specified as a string scalar or character vector.

propertyValue — Value of property

any data type, depending on the property

Value of property, specified in the format determined by the property.

objectType — Type of object

string scalar | character vector | class handle

Type of object for which to search, specified as a string scalar, character vector, or a class handle for an object.

Example: `find(ch, "-isa", "Stateflow.State")` finds the states in the chart `ch`.

Example: `find(ch, "-isa", class(object))` finds the objects of the same type as `object`.

depth — Depth of search

`inf` (default) | nonnegative integer scalar

Depth of search in the object hierarchy, specified as a nonnegative integer scalar or `inf`. Use `inf` to search all levels of the hierarchy.

functionName — Name of object function

string scalar | character vector

Name of object function, specified as a string scalar or character vector.

function — Filtering function

function handle

Filtering function, specified as a function handle. The function must return a logical scalar value that indicates whether an object is a match.

Output Arguments

objArray — Search results

array

Search results, returned as an array of Stateflow API objects.

Tips

- To limit search results based on the value of a subproperty, call `find` using "- function" and a function handle. For an example, see "Find Objects for Which Signal Logging Is Enabled" on page 3-25.
- Using the `find` function on `Simulink.Root`, `Simulink.BlockDiagram`, or `Stateflow.Machine` objects can return Simulink objects that match the search criteria you specify. For example, this command can return a Simulink subsystem or block named ABC:

```
find(sfroot, "Name", "ABC")
```

- Opening a main model that refers to a linked Stateflow chart does not guarantee that the Stateflow API can find the linked chart. To access the objects in a linked library chart, first load the library model into the Simulink workspace by performing one of these tasks:
 - Load the library model by calling the function `load_system`.
 - Call the function `find_system` with the `FollowLinks` argument set to on:

```
find_system(FollowLinks="on");
```
 - View a linked subsystem or block in the main model.
 - Compile or simulate the model.

Version History

Introduced before R2006a

See Also

`getChildren` | `getParent`

Topics

"Access Objects in Your Stateflow Chart" on page 1-6

"Summary of Stateflow API Objects and Properties" on page 1-36

"Regular Expressions"

fitToView

Package: Stateflow

Zoom in on graphical object

Syntax

```
fitToView(graphicalObject)
```

Description

`fitToView(graphicalObject)` zooms in on a graphical object in the Stateflow Editor.

Examples

Zoom in on State in Chart

Open a Simulink model called `myModel`. Suppose that the model contains a Stateflow chart with a state named A.

```
open_system("myModel")
```

Find the state named A.

```
st = find(sfroot, "-isa", "Stateflow.State", Name="A");
```

Zoom in on the state in the Stateflow Editor.

```
fitToView(st);
```

Input Arguments

graphicalObject — Graphical object

Stateflow.State object | Stateflow.Box object | Stateflow.Function object | ...

Graphical object, specified as a Stateflow API object of one of these types:

- Stateflow.Annotation
- Stateflow.AtomicBox
- Stateflow.AtomicSubchart
- Stateflow.Box
- Stateflow.Chart
- Stateflow.EMFunction
- Stateflow.Function
- Stateflow.Junction
- Stateflow.Port

- `Stateflow.SimulinkBasedState`
- `Stateflow.SLFunction`
- `Stateflow.State`
- `Stateflow.Transition`
- `Stateflow.TruthTable`

Version History

Introduced in R2008a

See Also

`view` | `highlight` | `zoomIn` | `zoomOut`

Topics

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

getChildren

Package: Stateflow

Identify children of object

Syntax

```
objArray = getChildren(parent)
```

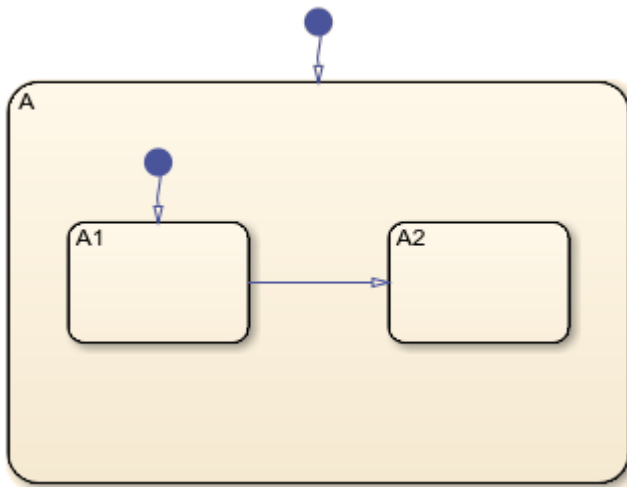
Description

`objArray = getChildren(parent)` returns an array of objects that have the specified parent.

Examples

Identify Children of Chart

This chart has two children, state A and a default transition. State A has four children, state A1, state A2, and two transitions.



Open the model and access the `Stateflow.Chart` object for the chart.

```
open_system("sfHierarchyAPIExample")
ch = find(sfroot, "-isa", "Stateflow.Chart");
```

Identify the children of the chart. Display the object types of the children.

```
children = getChildren(ch);
classes = arrayfun(@class, children, UniformOutput=false)

classes = 2x1 cell
    {'Stateflow.State' }
```

```
{'Stateflow.Transition'}
```

One element in `children` is a state. Display the name of the state.

```
idx = (classes=="Stateflow.State");
state = children(idx);
state.Name
```

```
ans =
'A'
```

Identify the children of state A. Display the object types of the children.

```
grandchildren = getChildren(state);
classes = arrayfun(@class,grandchildren,UniformOutput=false)
```

```
classes = 4x1 cell
    {'Stateflow.State'    }
    {'Stateflow.State'    }
    {'Stateflow.Transition'}
    {'Stateflow.Transition'}
```

Two elements in `grandchildren` are states. Display the names of the states.

```
idx = (classes=="Stateflow.State");
grandchildren(idx).Name
```

```
ans =
'A1'
```

```
ans =
'A2'
```

Input Arguments

parent — Parent object

Stateflow.Chart object | Stateflow.State object | Stateflow.Box object | Stateflow.Function object | ...

Parent object, specified as a Stateflow API object of one of these types:

- Stateflow.Box
- Stateflow.Chart
- Stateflow.EMChart
- Stateflow.EMFunction
- Stateflow.Function
- Stateflow.SimulinkBasedState
- Stateflow.State
- Stateflow.SLFunction
- Stateflow.StateTransitionTableChart

- Stateflow.TruthTable
- Stateflow.TruthTableChart

Version History

Introduced before R2006a

See Also

Functions

find | getParent | arrayfun | class

Objects

Stateflow.State | Stateflow.Box | Stateflow.Function

Topics

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

“Access Objects in Your Stateflow Chart” on page 1-6

getMappingForSymbol

Package: Stateflow

Get mapping for symbol in atomic subchart, atomic box, or Simulink based state

Syntax

```
chartSymbol = getMappingForSymbol(subsystem,subsystemSymbol)
[chartSymbol,expression] = getMappingForSymbol(subsystem,subsystemSymbol)
```

Description

`chartSymbol = getMappingForSymbol(subsystem,subsystemSymbol)` returns the main chart symbol to which the subsystem symbol `subsystemSymbol` maps, where `subsystem` is an atomic subchart, atomic box, or Simulink based state. For more information, see “Map Variables for Atomic Subcharts and Boxes” and “Map Variables for Simulink Based States”.

`[chartSymbol,expression] = getMappingForSymbol(subsystem,subsystemSymbol)` returns the main chart symbol and the nontrivial expression to which the subsystem symbol maps.

Examples

Map Variables in Atomic Subchart

In an atomic subchart called A, modify the mapping for the subchart input u1.

Open the model `sf_atomic_iodata_fixed.slx`.

```
open_system("sf_atomic_iodata_fixed")
```

Access the `Stateflow.AtomicSubchart` object for the atomic subchart A.

```
subsystem = find(sfroot,"-isa","Stateflow.AtomicSubchart", ...
    Name="A");
```

Use the `Subchart` property to access the `Stateflow.Data` object for subchart input u1.

```
subsystemSymbol = find(subsystem.Subchart, ...
    "-isa","Stateflow.Data",Name="u1");
```

Use the `Chart` property to access the `Stateflow.Data` object for chart input u2.

```
chartSymbol = find(subsystem.Chart, ...
    "-isa","Stateflow.Data",Name="u2");
```

Check the mapping for subchart input u1.

```
getMappingForSymbol(subsystem,subsystemSymbol).Name
```

```
ans =
'u1'
```

Map subchart input u1 to chart input u2.

```
setMappingForSymbol(subsystem, subsystemSymbol, chartSymbol)  
getMappingForSymbol(subsystem, subsystemSymbol).Name
```

```
ans =  
'u2'
```

Clear the mapping for subchart input u1.

```
clearMappingForSymbol(subsystem, subsystemSymbol)  
getMappingForSymbol(subsystem, subsystemSymbol).Name
```

```
ans =  
'u1'
```

Map Variables in Simulink Based State

In a Simulink based state called Locked, modify the mapping for the output we.

Open the model sf_clutch.slx.

```
open_system("sf_clutch.slx")
```

Access the Stateflow.SimulinkBasedState object for the Simulink based state Locked.

```
subsystem = find(sfroot, "-isa", "Stateflow.SimulinkBasedState", ...  
    Name="Locked");
```

Check the mapping for Simulink based state output we.

```
getMappingForSymbol(subsystem, "we").Name
```

```
ans =  
'we'
```

Map the Simulink based state output we to chart output wv.

```
setMappingForSymbol(subsystem, "we", "wv")  
getMappingForSymbol(subsystem, "we").Name
```

```
ans =  
'wv'
```

Clear the mapping for Simulink based state output we.

```
clearMappingForSymbol(subsystem, "we")  
getMappingForSymbol(subsystem, "we").Name
```

```
ans =  
'we'
```

Access Mapping When Parameter Maps to Expression

In an atomic subchart called A, find the expression that the parameter T maps to.

Open the model `sf_atomic_parameter_fixed.slx`.

```
open_system("sf_atomic_parameter_fixed")
```

Access the `Stateflow.AtomicSubchart` object for the atomic subchart A.

```
subsystem = find(sfroot, "-isa", "Stateflow.AtomicSubchart", ...
    Name="A");
```

Use the `Subchart` property to access the `Stateflow.Data` object for subchart parameter T.

```
subsystemSymbol = find(subsystem.Subchart, ...
    "-isa", "Stateflow.Data", Name="T");
```

Check the mapping for subchart parameter T.

```
[~,expression] = getMappingForSymbol(subsystem,subsystemSymbol)
```

```
expression =
'-1'
```

Input Arguments

subsystem — Atomic subchart, atomic box, or Simulink based state

`Stateflow.AtomicSubchart` object | `Stateflow.AtomicBox` object | `Stateflow.SimulinkBasedState` object

Atomic subchart, atomic box, or Simulink based state, specified as a `Stateflow.AtomicSubchart`, `Stateflow.AtomicBox`, or `Stateflow.SimulinkBasedState` object.

subsystemSymbol — Subsystem symbol

`Stateflow.Data` object | `Stateflow.Event` object | string scalar | character vector

Subsystem symbol, specified as a `Stateflow.Data` object, a `Stateflow.Event` object, a string scalar, or a character vector.

Note If the `subsystem` argument is a `Stateflow.SimulinkBasedState` object, this argument must be a string scalar or character vector.

Output Arguments

chartSymbol — Main chart symbol

`Stateflow.Data` object | `Stateflow.Event` object | []

Main chart symbol, returned as a `Stateflow.Data` object, a `Stateflow.Event` object, or an empty array []. If `subsystemSymbol` maps to a nontrivial expression, `chartSymbol` is an empty array.

expression — Mapping expression

character vector

Mapping expression, returned as a character vector. This expression can specify:

- A field of a `Stateflow` structure

- An element of a vector or matrix
- Any combination of structure fields or matrix indices

If `subsystemSymbol` maps to a main chart symbol, `expression` is an empty character vector, `''`.

Version History

Introduced in R2022b

R2023a: Map variables for Simulink based states

Edit the mapping of symbols in Simulink based states by calling the object functions `getMappingForSymbol`, `setMappingForSymbol`, and `clearMappingForSymbol` on `Stateflow.SimulinkBasedState` objects.

See Also

Functions

`clearMappingForSymbol` | `disableMappingForSymbol` | `setMappingForSymbol`

Objects

`Stateflow.AtomicBox` | `Stateflow.AtomicSubchart` | `Stateflow.Data` | `Stateflow.Event` | `Stateflow.SimulinkBasedState`

Topics

“Map Variables for Atomic Subcharts and Boxes”

“Map Variables for Simulink Based States”

getParent

Package: Stateflow

Identify parent of object

Syntax

```
parent = getParent(object)
```

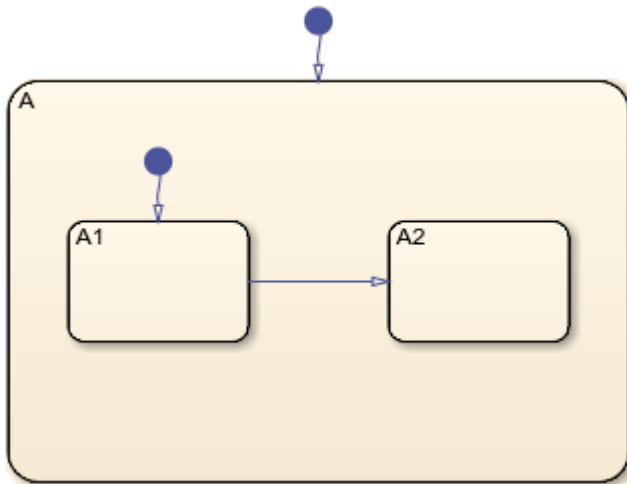
Description

`parent = getParent(object)` returns the parent of an object in a Stateflow chart, State Transition Table, Truth Table, or MATLAB Function block.

Examples

Identify Parent of State

In this chart, the parent of state A1 is state A. The parent of state A is the chart.



Open the model and access the `Stateflow.Chart` object for the chart.

```
open_system("sfHierarchyAPIExample")
ch = find(sfroot, "-isa", "Stateflow.Chart");
```

Find the `Stateflow.State` object named A1.

```
sA1 = find(sfroot, "-isa", "Stateflow.State", Name="A1");
```

Identify the parent of state A1. Display the name of the parent.

```
parent = getParent(sA1);  
parent.Name
```

```
ans =  
'A'
```

Identify the parent of state A. Display the name of the parent.

```
grandparent = getParent(parent);  
grandparent.Name
```

```
ans =  
'Chart'
```

Input Arguments

object – Object

Stateflow.State object | Stateflow.Box object | Stateflow.Function object | ...

Object in a Stateflow chart, State Transition Table, Truth Table, or MATLAB Function block, specified as a Stateflow API object of one of these types:

- Stateflow.Annotation
- Stateflow.AtomicBox
- Stateflow.AtomicSubchart
- Stateflow.Box
- Stateflow.Data
- Stateflow.EMFunction
- Stateflow.Event
- Stateflow.Function
- Stateflow.Junction
- Stateflow.Message
- Stateflow.Port
- Stateflow.SimulinkBasedState
- Stateflow.SLFunction
- Stateflow.State
- Stateflow.Transition
- Stateflow.TruthTable

Version History

Introduced before R2006a

See Also

Functions

find | getChildren

Objects

Stateflow.State | Stateflow.Box | Stateflow.Function

Topics

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

“Access Objects in Your Stateflow Chart” on page 1-6

getReferences

Package: Stateflow

Identify references to symbol name

Syntax

```
references = getReferences(symbol)
```

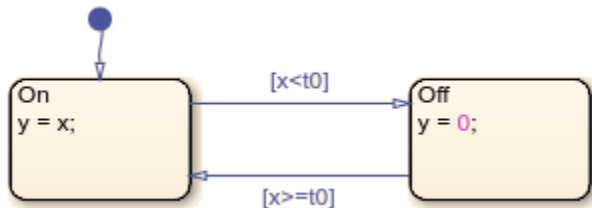
Description

`references = getReferences(symbol)` returns an array of `Stateflow.SymbolReference` objects that indicate where the chart refers to a symbol name. For example, a `Stateflow.SymbolReference` object can correspond to a state or transition action that includes the symbol name. Symbols include `Stateflow.Data`, `Stateflow.Event`, `Stateflow.Message`, and other Stateflow objects that have a `Name` property.

Examples

Rename Data and Update References in Chart

Rename and update the references to the chart output `y`.



Open the model and access the `Stateflow.Data` object for the chart output `y`.

```
open_system("sfRectifyAPIExample")
data = find(sfroot, "-isa", "Stateflow.Data", Scope="Output");
data.Name
```

```
ans =
'y'
```

Find the locations where the chart refers to the chart output.

```
references = getReferences(data)
```

```
references=2x1 object
  2x1 SymbolReference array with properties:
```

```
    Position
```

```
WhereUsed
```

Display the names and entry actions of the states that refer to the chart output.

```
whereUsed = [references.WhereUsed];
classes = arrayfun(@class,whereUsed,UniformOutput=false);
idx = (classes=="Stateflow.State");
states = whereUsed(idx);
get(states,{"Name" "EntryAction"})
```

```
ans = 2x2 cell
    {'On' }    {'y = x;'}
    {'Off'}    {'y = 0;'}

```

Change the Name property of the chart output to "z" and update the references to the output in the chart.

```
renameReferences(data,"z")
data.Name
```

```
ans =
    'z'
```

Display the names and modified entry actions of the states that refer to the chart output.

```
get(states,{"Name" "EntryAction"})
```

```
ans = 2x2 cell
    {'On' }    {'z = x;'}
    {'Off'}    {'z = 0;'}

```

Input Arguments

symbol — Symbol

Stateflow.Data object | Stateflow.Event object | Stateflow.Message object | ...

Symbol, specified as a Stateflow API object of one of these types:

- Stateflow.AtomicBox
- Stateflow.AtomicSubchart
- Stateflow.Box
- Stateflow.Data
- Stateflow.EMFunction
- Stateflow.Event
- Stateflow.Function
- Stateflow.Message
- Stateflow.SimulinkBasedState
- Stateflow.SLFunction
- Stateflow.State

- `Stateflow.TruthTable`

Limitations

- The functions `getReferences` and `renameReferences` do not find or update references to exported functions in other charts, atomic subcharts, or Function Caller blocks.

Version History

Introduced in R2023a

See Also

Functions

`find` | `renameReferences`

Objects

`Stateflow.Data` | `Stateflow.Event` | `Stateflow.Message` | `Stateflow.State` | `Stateflow.SymbolReference`

Topics

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

“Summary of Stateflow API Objects and Properties” on page 1-36

“Modify Properties and Call Functions of Stateflow Objects” on page 1-11

highlight

Package: Stateflow

Highlight graphical object

Syntax

```
highlight(graphicalObject)
```

Description

`highlight(graphicalObject)` highlights a graphical object in the Stateflow Editor.

Examples

Highlight State in Chart

Open a Simulink model called `myModel`. Suppose that the model contains a Stateflow chart with a state named A.

```
open_system("myModel")
```

Find the state named A.

```
st = find(sfroot, "-isa", "Stateflow.State", Name="A");
```

Highlight the state in the Stateflow Editor.

```
highlight(st);
```

Input Arguments

`graphicalObject` — Graphical object

`Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object | ...

Graphical object, specified as a Stateflow API object of one of these types:


- `Stateflow.AtomicBox`
- `Stateflow.AtomicSubchart`
- `Stateflow.Box`
- `Stateflow.EMFunction`
- `Stateflow.Function`
- `Stateflow.Junction`
- `Stateflow.Port`
- `Stateflow.SimulinkBasedState`
- `Stateflow.SLFunction`

- Stateflow.State
- Stateflow.Transition
- Stateflow.TruthTable

Tips

To clear the highlighting, use the `hilite_system` function. For example, to clear the highlighting in chart `ch`, enter:

```
hilite_system(ch.Path, "none")
```

Alternatively, you can use the Stateflow Editor. In the **Debug** tab, under **Animation**, click the Remove animation highlighting button .

Version History

Introduced in R2012a

See Also

`view` | `fitToView` | `hilite_system` | `zoomIn` | `zoomOut`

Topics

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

innerTransitions

Package: Stateflow

Identify inner transitions with specified source

Syntax

```
transitions = innerTransitions(source)
```

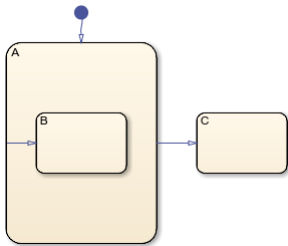
Description

`transitions = innerTransitions(source)` returns an array of `Stateflow.Transition` objects that correspond to the inner transitions of the specified source state. An inner transition is a transition that does not exit the source state. For more information, see “Control Chart Execution by Using Inner Transitions”.

Examples

Identify Inner Transitions

Suppose that `ch` is the `Stateflow.Chart` object that corresponds to this chart.



Find the `Stateflow.State` object named A.

```
sA = find(ch, "-isa", "Stateflow.State", Name="A");
```

Identify the transition whose source is state A and whose destination is inside state A. Display the name of the destination.

```
tr = innerTransitions(sA);
tr.Destination.Name
```

```
ans =
```

```
    'B'
```

Input Arguments

source — Source state
`Stateflow.State` object

Source state, specified as a `Stateflow.State` object.

Version History

Introduced before R2006a

See Also

Functions

`find` | `defaultTransitions` | `outerTransitions` | `sinkedTransitions` | `sourcedTransitions`

Objects

`Stateflow.State` | `Stateflow.Transition`

Topics

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

“Access Objects in Your Stateflow Chart” on page 1-6

“Control Chart Execution by Using Inner Transitions”

“Group and Execute Transitions”

isCommented

Package: Stateflow

(Removed) Determine if graphical object is commented out

Note `isCommented` has been removed. Use the property `IsCommented` instead. For information on updating your code, see “Version History”.

Syntax

```
tf = isCommented(graphicalObject)
```

Description

`tf = isCommented(graphicalObject)` returns a logical value that indicates if a graphical object is commented out. The function returns logical 1 (`true`) if:

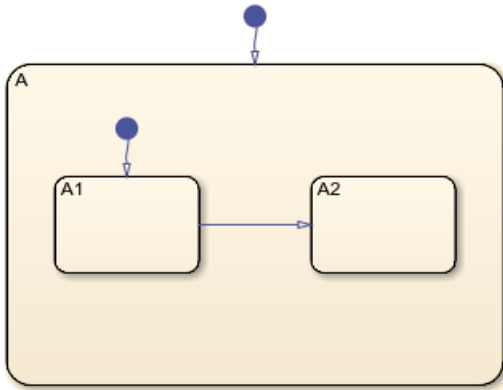
- The graphical object is explicitly commented out. To explicitly comment out an object, set its `IsExplicitlyCommented` property to `true`. Alternatively, you can right-click the graphical object and select **Comment Out**.
- The graphical object is implicitly commented out. In this case, its `IsImplicitlyCommented` property has a value of `true`. An object is implicitly commented out when you explicitly comment out a state, box, or function that contains the object. Additionally,
 - Transitions are implicitly commented out when you comment out their source or destination.
 - Entry and exit ports are implicitly commented out when you comment out their matching entry or exit junction.

Otherwise, the function returns logical 0 (`false`).

Examples

Comment Out State

When you explicitly comment out a state, box, or function, you implicitly comment out all the graphical objects that it contains. For example, when you comment out state A in this chart, you also comment out its substates, A1 and A2.



Find the `Stateflow.State` objects named A, A1, and A2.

```
sA = find(ch, "-isa", "Stateflow.State", Name="A");
sA1 = find(ch, "-isa", "Stateflow.State", Name="A1");
sA2 = find(ch, "-isa", "Stateflow.State", Name="A2");
```

Check that state A and its substates are not commented out.

```
[isCommented(sA), isCommented(sA1), isCommented(sA2)]
```

```
ans =
```

```
1x3 logical array
```

```
0 0 0
```

Explicitly comment out state A.

```
sA.IsExplicitlyCommented = true;
```

Check that state A and its substates are commented out.

```
[isCommented(sA), isCommented(sA1), isCommented(sA2)]
```

```
ans =
```

```
1x3 logical array
```

```
1 1 1
```

Input Arguments

graphicalObject — Graphical object

`Stateflow.State` object | `Stateflow.Box` object | `Stateflow.Function` object | ...

Graphical object, specified as a Stateflow API object of one of these types:

- `Stateflow.AtomicBox`
- `Stateflow.AtomicSubchart`
- `Stateflow.Box`

- Stateflow.EMFunction
- Stateflow.Function
- Stateflow.Junction
- Stateflow.Port
- Stateflow.SimulinkBasedState
- Stateflow.SLFunction
- Stateflow.State
- Stateflow.Transition
- Stateflow.TruthTable

Version History

Introduced in R2016a

R2023a: Removed

Errors starting in R2023a

isCommented has been removed. To determine whether a graphical object is commented out, use the property IsCommented instead:

```
tf = graphicalObject.IsCommented
```

For more information, see “Comment Out State” on page 3-5.

See Also

Functions

find

Objects

Stateflow.State | Stateflow.Box | Stateflow.Function

Topics

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

“Comment Out Objects in a Stateflow Chart”

“Summary of Stateflow API Objects and Properties” on page 1-36

outerTransitions

Package: Stateflow

Identify outgoing transitions with specified source

Syntax

```
transitions = outerTransitions(source)
```

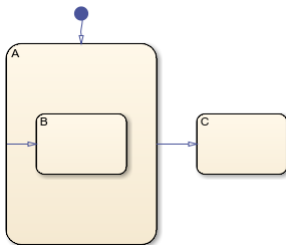
Description

`transitions = outerTransitions(source)` returns an array of `Stateflow.Transition` objects that correspond to the outer transitions of the specified source state. An outer transition is a transition that exits the source state.

Examples

Identify Outgoing Transitions

Suppose that `ch` is the `Stateflow.Chart` object that corresponds to this chart.



Find the `Stateflow.State` object named A.

```
sA = find(ch, "-isa", "Stateflow.State", Name="A");
```

Identify the transition whose source is state A and whose destination is outside of state A. Display the name of the destination.

```
tr = outerTransitions(sA);
tr.Destination.Name
```

```
ans =
```

```
    'C'
```

Input Arguments

source — Source state
`Stateflow.State` object

Source state, specified as a `Stateflow.State` object.

Version History

Introduced before R2006a

See Also

Functions

`find` | `defaultTransitions` | `innerTransitions` | `sinkedTransitions` | `sourcedTransitions`

Objects

`Stateflow.State` | `Stateflow.Transition`

Topics

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

“Access Objects in Your Stateflow Chart” on page 1-6

“Group and Execute Transitions”

pasteTo

Package: Stateflow

Paste objects in clipboard to specified container object

Syntax

```
pasteTo(clipboard,parent)
```

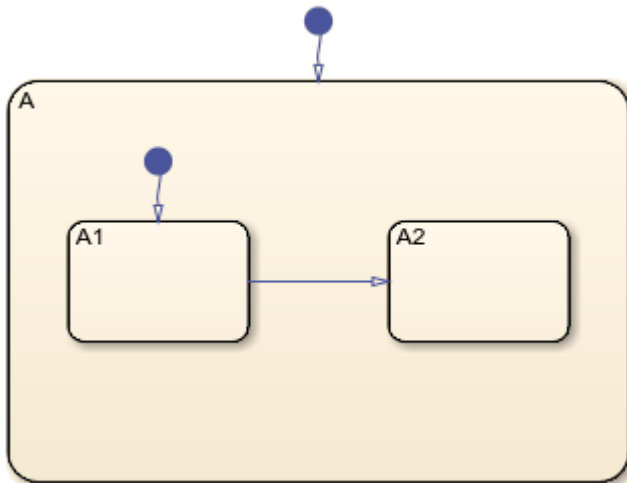
Description

`pasteTo(clipboard,parent)` pastes the contents of the clipboard to the specified parent. To copy objects to the clipboard, use the `copy` function.

Examples

Copy and Paste by Grouping

Group a state and copy its contents to the chart. When you group a state, box, or graphical function, you can copy and paste all the objects contained in the grouped object, as well as all the relationships among these objects. This method is the simplest way of copying and pasting objects programmatically. If a state is not grouped, copying the state does not copy any of its contents.



Open the model and access the `Stateflow.Chart` object for the chart.

```
open_system("sfHierarchyAPIExample")  
ch = find(sfroot, "-isa", "Stateflow.Chart");
```

Find the `Stateflow.State` object named A.

```
sA = find(ch, "-isa", "Stateflow.State", Name="A");
```


Group state A and its contents by setting the `IsGrouped` property for `sA` to `true`. Save the previous setting of this property so you can revert to it later.

```
prevGrouping = sA.IsGrouped;
sA.IsGrouped = true;
```

Change the name of the state to `Copy_of_A`. Save the previous name so you can revert to it later.

```
prevName = sA.Name;
newName = "Copy_of_" + prevName;
sA.Name = newName;
```

Access the clipboard object.

```
cb = sfclipboard;
```

Copy the grouped state to the clipboard.

```
copy(cb, sA);
```

Restore the state properties to their original settings.

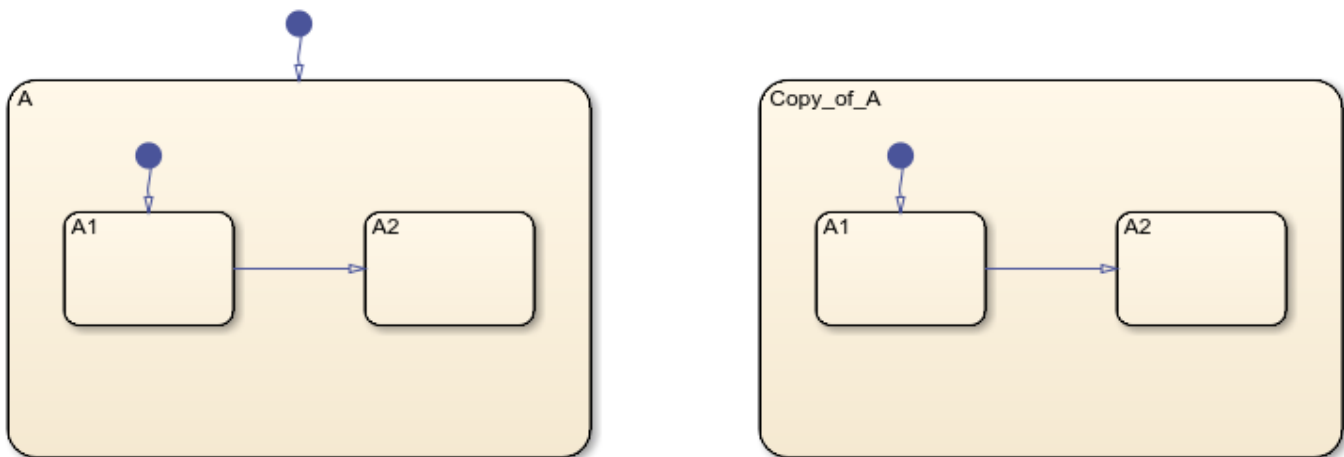
```
sA.IsGrouped = prevGrouping;
sA.Name = prevName;
```

Paste a copy of the objects from the clipboard to the chart.

```
pasteTo(cb, ch);
```

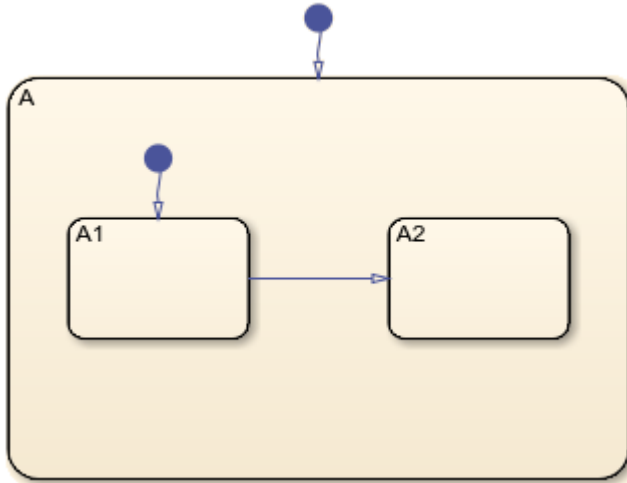
Adjust the state properties of the new state.

```
sNew = find(ch, "-isa", "Stateflow.State", Name=newName);
sNew.Position = sA.Position + [400 0 0 0];
sNew.IsGrouped = prevGrouping;
```



Copy and Paste Array of Objects

Copy states A1 and A2, along with the transition between them, to a new state in the chart. To preserve transition connections and containment relationships between objects, copy all the connected objects at once.



Open the model and access the `Stateflow.Chart` object for the chart.

```
open_system("sfHierarchyAPIExample")
ch = find(sfroot, "-isa", "Stateflow.Chart");
```

Find the `Stateflow.State` object named A.

```
sA = find(ch, "-isa", "Stateflow.State", Name="A");
```

Add a new state called B. To enable pasting of other objects inside B, convert the new state to a subchart.

```
sB = Stateflow.State(ch);
sB.Name = "B";
sB.Position = sA.Position + [400 0 0 0];
sB.IsSubchart = true;
```

Create an array called `objArray` that contains the states and transitions in state A. Use the function `setdiff` to remove state A from the array of objects to copy.

```
objArrayS = find(sA, "-isa", "Stateflow.State");
objArrayS = setdiff(objArrayS, sA);
objArrayT = find(sA, "-isa", "Stateflow.Transition");
objArray = [objArrayS objArrayT];
```

Access the clipboard object.

```
cb = sfclipboard;
```

Copy the objects in `objArray` and paste them in subchart B.

```
copy(cb, objArray);
pasteTo(cb, sB);
```

Revert B to a state.

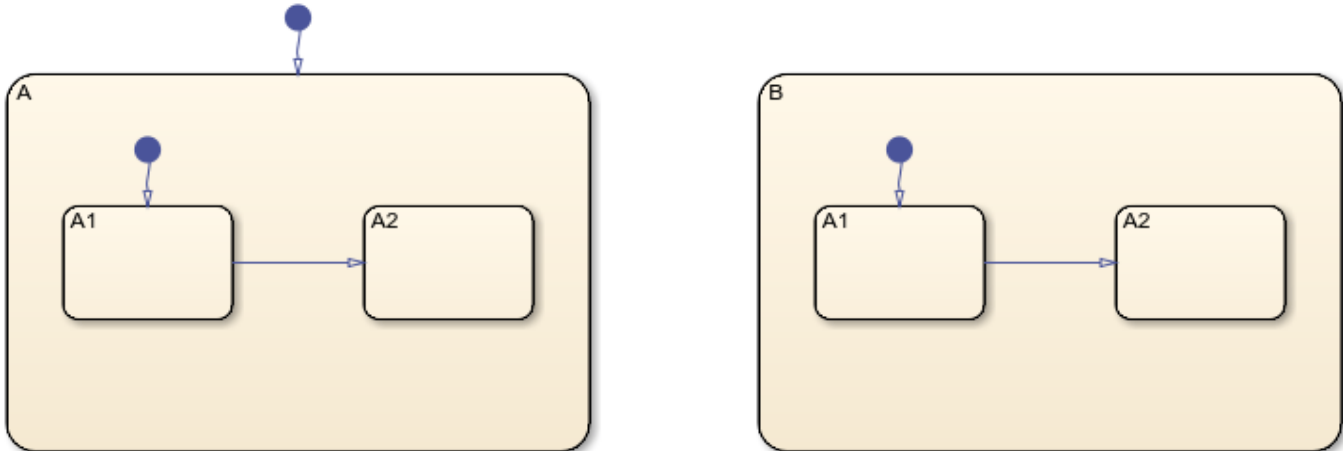
```
sB.IsSubchart = false;
sB.IsGrouped = false;
```

Reposition the states and transitions in B.

```
newStates = find(sB,"-isa","Stateflow.State");
newStates = setdiff(newStates,sB);

newTransitions = find(sB,"-isa","Stateflow.Transition");
newOClocks = get(newTransitions,{"SourceOClock","DestinationOClock"});

for i = 1:numel(newStates)
    newStates(i).Position = newStates(i).Position + [25 35 0 0];
end
set(newTransitions,{"SourceOClock","DestinationOClock"},newOClocks);
```



Input Arguments

clipboard – Clipboard

Stateflow.Clipboard object

Clipboard, specified as a Stateflow.Clipboard object.

parent – Parent for copied objects

Stateflow.Chart object | Stateflow.State object | Stateflow.Box object | Stateflow.Function object | ...

Parent for the copied objects, specified as a Stateflow API object of one of these types:

- Stateflow.Box
- Stateflow.Chart
- Stateflow.EMFunction
- Stateflow.Function

- `Stateflow.SimulinkBasedState`
- `Stateflow.SLFunction`
- `Stateflow.State`
- `Stateflow.TruthTable`

If the objects in the clipboard are all graphical (states, boxes, functions, annotations, transitions, or junctions), this object must be a chart or subchart.

Tips

When you paste graphical objects, the new parent must be a chart or a subchart. To convert a state, box, or graphical function to a subchart, set its `IsSubchart` property to `true`. After pasting, you can revert the parent by setting its `IsSubchart` and `IsGrouped` properties to `false`.

Version History

Introduced before R2006a

See Also

Functions

`copy` | `find` | `setdiff` | `sfclipboard`

Objects

`Stateflow.State` | `Stateflow.Clipboard`

Topics

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

renameReferences

Package: Stateflow

Rename symbol and update references to symbol name

Syntax

```
renameReferences(symbol, newName)
```

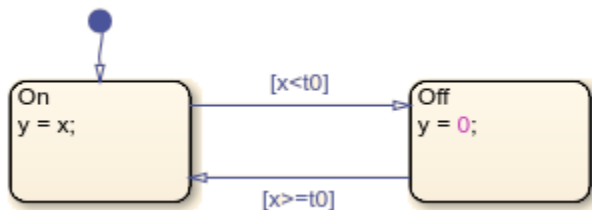
Description

`renameReferences(symbol, newName)` changes the name of `symbol` to `newName` and updates the references to the symbol name in the chart. Symbols include `Stateflow.Data`, `Stateflow.Event`, `Stateflow.Message`, and other Stateflow objects that have a `Name` property.

Examples

Rename Data and Update References in Chart

Rename and update the references to the chart output `y`.



Open the model and access the `Stateflow.Data` object for the chart output `y`.

```
open_system("sfRectifyAPIExample")
data = find(sfroot, "-isa", "Stateflow.Data", Scope="Output");
data.Name
```

```
ans =
'y'
```

Find the locations where the chart refers to the chart output.

```
references = getReferences(data)

references=2x1 object
 2x1 SymbolReference array with properties:

    Position
    WhereUsed
```

Display the names and entry actions of the states that refer to the chart output.

```
whereUsed = [references.WhereUsed];
classes = arrayfun(@class,whereUsed,UniformOutput=false);
idx = (classes=="Stateflow.State");
states = whereUsed(idx);
get(states,{"Name" "EntryAction"})

ans = 2x2 cell
    {'On' }    {'y = x;'}
    {'Off'}   {'y = 0;'}

```

Change the Name property of the chart output to "z" and update the references to the output in the chart.

```
renameReferences(data,"z")
data.Name
```

```
ans =
'z'
```

Display the names and modified entry actions of the states that refer to the chart output.

```
get(states,{"Name" "EntryAction"})
```

```
ans = 2x2 cell
    {'On' }    {'z = x;'}
    {'Off'}   {'z = 0;'}

```

Input Arguments

symbol — Symbol

Stateflow.Data object | Stateflow.Event object | Stateflow.Message object | ...

Symbol, specified as a Stateflow API object of one of these types:

- Stateflow.AtomicBox
- Stateflow.AtomicSubchart
- Stateflow.Box
- Stateflow.Data
- Stateflow.EMFunction
- Stateflow.Event
- Stateflow.Function
- Stateflow.Message
- Stateflow.SimulinkBasedState
- Stateflow.SLFunction
- Stateflow.State
- Stateflow.TruthTable

newName — New name

string scalar | character vector

New name for the symbol, specified as a string scalar or a character vector.

Limitations

- The functions `getReferences` and `renameReferences` do not find or update references to exported functions in other charts, atomic subcharts, or Function Caller blocks.

Version History

Introduced in R2023a

See Also

Functions

`find` | `getReferences`

Objects

`Stateflow.Data` | `Stateflow.Event` | `Stateflow.Message` | `Stateflow.State`

Topics

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

“Refactor Charts Programmatically” on page 1-26

“Summary of Stateflow API Objects and Properties” on page 1-36

“Modify Properties and Call Functions of Stateflow Objects” on page 1-11

setImage

Package: Stateflow

Insert image into annotation

Syntax

```
setImage(annotation, source)
```

Description

`setImage(annotation, source)` inserts an image from the clipboard or an image file into an annotation.

Examples

Add Image Annotation to Chart

Add an annotation in the chart `ch`. Use the file `myImageFile.png`, which is located in the folder `myfolder/annotation_images`, as the image for the annotation.

```
annotation = Stateflow.Annotation(ch);  
setImage(annotation, ...  
    fullfile("myfolder", "annotation_images", "myImageFile.png"));
```

Input Arguments

annotation — Annotation

`Stateflow.Annotation` object

Annotation, specified as a `Stateflow.Annotation` object.

source — Source of image

string scalar | character array | "clipboard" | ""

Source of the image, specified as a string scalar or character array that contains the full path and name of an image file. Alternatively, to insert an image from the clipboard, specify "clipboard".

To convert an image annotation to a text annotation, specify "".

Version History

Introduced in R2014a

See Also

Functions

`fullfile`

Objects

Stateflow.Annotation

Topics

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

setMappingForSymbol

Package: Stateflow

Set mapping for symbol in atomic subchart, atomic box, or Simulink based state

Syntax

```
setMappingForSymbol(subsystem, subsystemSymbol, chartSymbol)
```

Description

`setMappingForSymbol(subsystem, subsystemSymbol, chartSymbol)` maps the subsystem symbol `subsystemSymbol` to the main chart symbol `chartSymbol`, where `subsystem` is an atomic subchart, atomic box, or Simulink based state. For more information, see “Map Variables for Atomic Subcharts and Boxes” and “Map Variables for Simulink Based States”.

Examples

Map Variables in Atomic Subchart

In an atomic subchart called A, modify the mapping for the subchart input u1.

Open the model `sf_atomic_iodata_fixed.slx`.

```
open_system("sf_atomic_iodata_fixed")
```

Access the `Stateflow.AtomicSubchart` object for the atomic subchart A.

```
subsystem = find(sfroot, "-isa", "Stateflow.AtomicSubchart", ...  
    Name="A");
```

Use the `Subchart` property to access the `Stateflow.Data` object for subchart input u1.

```
subsystemSymbol = find(subsystem.Subchart, ...  
    "-isa", "Stateflow.Data", Name="u1");
```

Use the `Chart` property to access the `Stateflow.Data` object for chart input u2.

```
chartSymbol = find(subsystem.Chart, ...  
    "-isa", "Stateflow.Data", Name="u2");
```

Check the mapping for subchart input u1.

```
getMappingForSymbol(subsystem, subsystemSymbol).Name
```

```
ans =  
'u1'
```

Map subchart input u1 to chart input u2.

```
setMappingForSymbol(subsystem, subsystemSymbol, chartSymbol)  
getMappingForSymbol(subsystem, subsystemSymbol).Name
```

```
ans =
'u2'
```

Clear the mapping for subchart input u1.

```
clearMappingForSymbol(subsystem, subsystemSymbol)
getMappingForSymbol(subsystem, subsystemSymbol).Name
```

```
ans =
'u1'
```

Map Variables in Simulink Based State

In a Simulink based state called Locked, modify the mapping for the output we.

Open the model sf_clutch.slx.

```
open_system("sf_clutch.slx")
```

Access the Stateflow.SimulinkBasedState object for the Simulink based state Locked.

```
subsystem = find(sfroot, "-isa", "Stateflow.SimulinkBasedState", ...
    Name="Locked");
```

Check the mapping for Simulink based state output we.

```
getMappingForSymbol(subsystem, "we").Name
```

```
ans =
'we'
```

Map the Simulink based state output we to chart output wv.

```
setMappingForSymbol(subsystem, "we", "wv")
getMappingForSymbol(subsystem, "we").Name
```

```
ans =
'wv'
```

Clear the mapping for Simulink based state output we.

```
clearMappingForSymbol(subsystem, "we")
getMappingForSymbol(subsystem, "we").Name
```

```
ans =
'we'
```

Input Arguments

subsystem — Atomic subchart, atomic box, or Simulink based state

Stateflow.AtomicSubchart object | Stateflow.AtomicBox object |
Stateflow.SimulinkBasedState object

Atomic subchart, atomic box, or Simulink based state, specified as a Stateflow.AtomicSubchart, Stateflow.AtomicBox, or Stateflow.SimulinkBasedState object.

subsystemSymbol — Subsystem symbol

Stateflow.Data object | Stateflow.Event object | string scalar | character vector

Subsystem symbol, specified as a Stateflow.Data object, a Stateflow.Event object, a string scalar, or a character vector.

Note If the subsystem argument is a Stateflow.SimulinkBasedState object, this argument must be a string scalar or character vector.

chartSymbol — Main chart symbol

Stateflow.Data object | Stateflow.Event object | string scalar | character vector

Main chart symbol, specified as a Stateflow.Data object, a Stateflow.Event object, a string scalar, or a character vector.

Limitations

- The setMappingForSymbol function does not support mapping symbols in atomic subcharts and atomic boxes to expressions.

Version History

Introduced in R2022b

R2023a: Map variables for Simulink based states

Edit the mapping of symbols in Simulink based states by calling the object functions getMappingForSymbol, setMappingForSymbol, and clearMappingForSymbol on Stateflow.SimulinkBasedState objects.

See Also**Functions**

clearMappingForSymbol | disableMappingForSymbol | getMappingForSymbol

Objects

Stateflow.AtomicBox | Stateflow.AtomicSubchart | Stateflow.Data | Stateflow.Event | Stateflow.SimulinkBasedState

Topics

“Map Variables for Atomic Subcharts and Boxes”

“Map Variables for Simulink Based States”

sunkedTransitions

Package: Stateflow

Identify transitions with specified destination

Syntax

```
transitions = sunkedTransitions(destination)
```

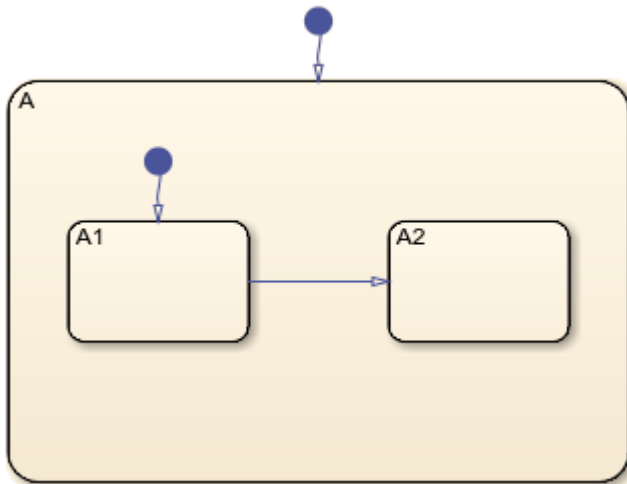
Description

`transitions = sunkedTransitions(destination)` returns an array of `Stateflow.Transition` objects with the specified destination.

Examples

Identify Transitions With Specified Destination

Find the transition that enters state A2.



Open the model and access the `Stateflow.Chart` object for the chart.

```
open_system("sfHierarchyAPIExample")
ch = find(sfroot, "-isa", "Stateflow.Chart");
```

Find the `Stateflow.State` object named A2.

```
sA2 = find(sfroot, "-isa", "Stateflow.State", Name="A2");
```

Identify the transition whose destination is state A2. Display the name of the source.

```
tr = sinkedTransitions(sA2);  
tr.Source.Name  
  
ans =  
'A1'
```

Input Arguments

destination – Destination object

Stateflow.Junction object | Stateflow.Port object | Stateflow.State object

Destination object, specified as a Stateflow API object of one of these types:

- Stateflow.Junction
- Stateflow.Port
- Stateflow.State

Version History

Introduced in R2012a

See Also

Functions

find | defaultTransitions | innerTransitions | outerTransitions |
sourcedTransitions

Objects

Stateflow.Junction | Stateflow.Port | Stateflow.State | Stateflow.Transition

Topics

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

“Access Objects in Your Stateflow Chart” on page 1-6

sourcedTransitions

Package: Stateflow

Identify transitions with specified source

Syntax

```
transitions = sourcedTransitions(source)
```

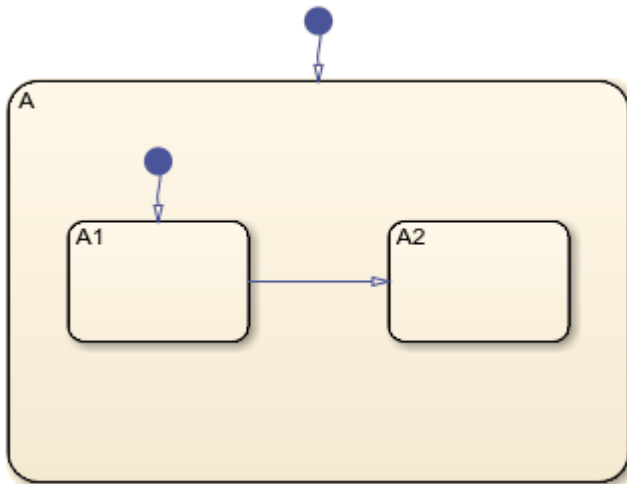
Description

`transitions = sourcedTransitions(source)` returns an array of `Stateflow.Transition` objects with the specified source.

Examples

Identify Transitions With Specified Source

Find the transition that exits state A1.



Open the model and access the `Stateflow.Chart` object for the chart.

```
open_system("sfHierarchyAPIExample")
ch = find(sfroot, "-isa", "Stateflow.Chart");
```

Find the `Stateflow.State` object named A1.

```
sA1 = find(sfroot, "-isa", "Stateflow.State", Name="A1");
```

Identify the transition whose source is state A1. Display the name of the destination.

```
tr = sourcedTransitions(sA1);  
tr.Destination.Name  
  
ans =  
'A2'
```

Input Arguments

source — Source object

Stateflow.Junction object | Stateflow.Port object | Stateflow.State object

Source object, specified as a Stateflow API object of one of these types:

- Stateflow.Junction
- Stateflow.Port
- Stateflow.State

Version History

Introduced before R2006a

See Also

Functions

find | defaultTransitions | innerTransitions | outerTransitions | sinkedTransitions

Objects

Stateflow.Junction | Stateflow.Port | Stateflow.State | Stateflow.Transition

Topics

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

“Access Objects in Your Stateflow Chart” on page 1-6

up

Package: Stateflow

(Not recommended) Identify parent of object

Note Using `up` is not recommended. Use `getParent` instead.

Syntax

```
parent = up(object)
```

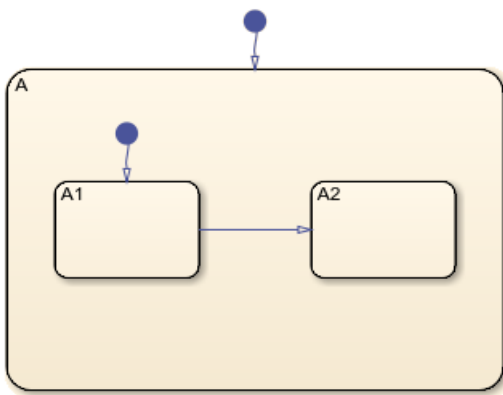
Description

`parent = up(object)` returns the parent of an object in a Stateflow chart, State Transition Table, Truth Table, or MATLAB Function block.

Examples

Identify Parent of State

Suppose that `ch` is the `Stateflow.Chart` object that corresponds to this chart. In this chart, the parent of state `A1` is state `A`. The parent of state `A` is the chart.



Find the `Stateflow.State` object named `A1`.

```
sA1 = find(ch, "-isa", "Stateflow.State", Name="A1");
```

Identify the parent of state `A1`. Display the name of the parent.

```
parent = up(sA1);
parent.Name
```

```
ans =  
    'A'
```

Identify the parent of state A. Display the name of the parent.

```
grandparent = up(parent);  
grandparent.Name  
  
ans =  
    'Chart'
```

Input Arguments

object – Object

`Stateflow.State object` | `Stateflow.Box object` | `Stateflow.Function object` | ...

Object in a Stateflow chart, State Transition Table, Truth Table, or MATLAB Function block, specified as a Stateflow API object of one of these types:

- `Stateflow.Annotation`
- `Stateflow.AtomicBox`
- `Stateflow.AtomicSubchart`
- `Stateflow.Box`
- `Stateflow.Data`
- `Stateflow.EMFunction`
- `Stateflow.Event`
- `Stateflow.Function`
- `Stateflow.Junction`
- `Stateflow.Message`
- `Stateflow.SimulinkBasedState`
- `Stateflow.SLFunction`
- `Stateflow.State`
- `Stateflow.Transition`
- `Stateflow.TruthTable`

Version History

Introduced before R2006a

See Also

Functions

`find` | `getChildren` | `getParent`

Objects

`Stateflow.State` | `Stateflow.Box` | `Stateflow.Function`

Topics

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

“Access Objects in Your Stateflow Chart” on page 1-6

view

Package: Stateflow

Display object in editing environment

Syntax

view(object)

Description

view(object) displays an object in its editing environment, such as the Stateflow, MATLAB, and Simulink Editors.

- The Stateflow Editor displays the contents of these objects:
 - Stateflow.AtomicBox
 - Stateflow.AtomicSubchart
 - Stateflow.Box with IsSubchart set to true
 - Stateflow.Chart
 - Stateflow.Function with IsSubchart set to true
 - Stateflow.State with IsSubchart set to true
- The Stateflow Editor shows these objects in their subviewer:
 - Stateflow.Annotation
 - Stateflow.Box with IsSubchart set to false
 - Stateflow.Function with IsSubchart set to false
 - Stateflow.Junction
 - Stateflow.Port
 - Stateflow.State with IsSubchart set to false
 - Stateflow.Transition
- The MATLAB Function Editor displays the code for Stateflow.EMChart and Stateflow.EMFunction objects.
- The Simulink Editor displays the block diagram for Stateflow.SimulinkBasedState and Stateflow.SLFunction objects.
- The Truth Table Editor displays the content of Stateflow.TruthTable and Stateflow.TruthTableChart objects.
- The State Transition Table Editor displays the content of Stateflow.StateTransitionTableChart objects.
- The Model Explorer displays the properties of these objects:
 - Stateflow.Data
 - Stateflow.Event

- Stateflow.Message

Examples

Display State in Chart

Open a Simulink model called `myModel`. Suppose that the model contains a Stateflow chart with a state named A.

```
open_system("myModel")
```

Find the state named A.

```
st = find(sfroot, "-isa", "Stateflow.State", Name="A");
```

Display the state in the Stateflow Editor.

```
view(st);
```

Input Arguments

object — Object to view

Stateflow.Chart object | Stateflow.State object | Stateflow.Box object | Stateflow.Function object | ...

Object to view, specified as a Stateflow API object of one of these types:

- Stateflow.Annotation
- Stateflow.AtomicBox
- Stateflow.AtomicSubchart
- Stateflow.Box
- Stateflow.Chart
- Stateflow.Data
- Stateflow.EMChart
- Stateflow.EMFunction
- Stateflow.Event
- Stateflow.Function
- Stateflow.Junction
- Stateflow.Message
- Stateflow.Port
- Stateflow.SimulinkBasedState
- Stateflow.SLFunction
- Stateflow.State
- Stateflow.StateTransitionTableChart
- Stateflow.Transition
- Stateflow.TruthTable

- `Stateflow.TruthTableChart`

Version History

Introduced before R2006a

See Also

`highlight` | `fitToView` | `zoomIn` | `zoomOut`

Topics

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

zoomIn

Package: Stateflow

Zoom in on Stateflow chart

Syntax

```
zoomIn(editor)
```

Description

`zoomIn(editor)` increases the magnification level of the `Stateflow.Editor` object `editor` for a chart.

Examples

Zoom in on Stateflow Chart

Increase the magnification level of a nonempty chart `ch`.

```
ed = ch.Editor;  
zoomIn(ed)
```

If the magnification level for the chart was initially 100%, this command increases it to 130%.

Input Arguments

editor — Editor for chart

`Stateflow.Editor` object

Editor for a chart, specified as a `Stateflow.Editor` object. The `Stateflow.Editor` object provides access to the graphical aspects of a chart. For example, to access the `Stateflow.Editor` object for a `Stateflow.Chart` object `ch`, enter:

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

Algorithms

The `zoomIn` function modifies the `ZoomFactor` property of the `Stateflow.Editor` object. The property is limited to a minimum of 0.5 and a maximum of 10. `zoomIn` multiplies `ZoomFactor` by a factor of 1.3 as long as the resulting value is in this range. Otherwise, `zoomIn` sets `ZoomFactor` to the maximum value of 10.

Version History

Introduced before R2006a

See Also

view | highlight | fitToView | zoomOut

Topics

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

zoomOut

Package: Stateflow

Zoom out on Stateflow chart

Syntax

```
zoomOut(editor)
```

Description

`zoomOut(editor)` reduces the magnification level of the `Stateflow.Editor` object `editor` for a chart.

Examples

Zoom out on Stateflow Chart

Decrease the magnification level of a nonempty chart `ch`.

```
ed = ch.Editor;  
zoomOut(ed)
```

If the magnification level for the chart was initially 100%, this command decreases it to 76.9%.

Input Arguments

editor — Editor for chart

`Stateflow.Editor` object

Editor for a chart, specified as a `Stateflow.Editor` object. The `Stateflow.Editor` object provides access to the graphical aspects of a chart. For example, to access the `Stateflow.Editor` object for a `Stateflow.Chart` object `ch`, enter:

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

Algorithms

The `zoomOut` function modifies the `ZoomFactor` property of the `Stateflow.Editor` object. The property is limited to a minimum of 0.5 and a maximum of 10. `zoomOut` divides `ZoomFactor` by a factor of 1.3 as long as the resulting value is in this range. Otherwise, `zoomOut` sets `ZoomFactor` to the minimum value of 0.5.

Version History

Introduced before R2006a

See Also

view | highlight | fitToView | zoomIn

Topics

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