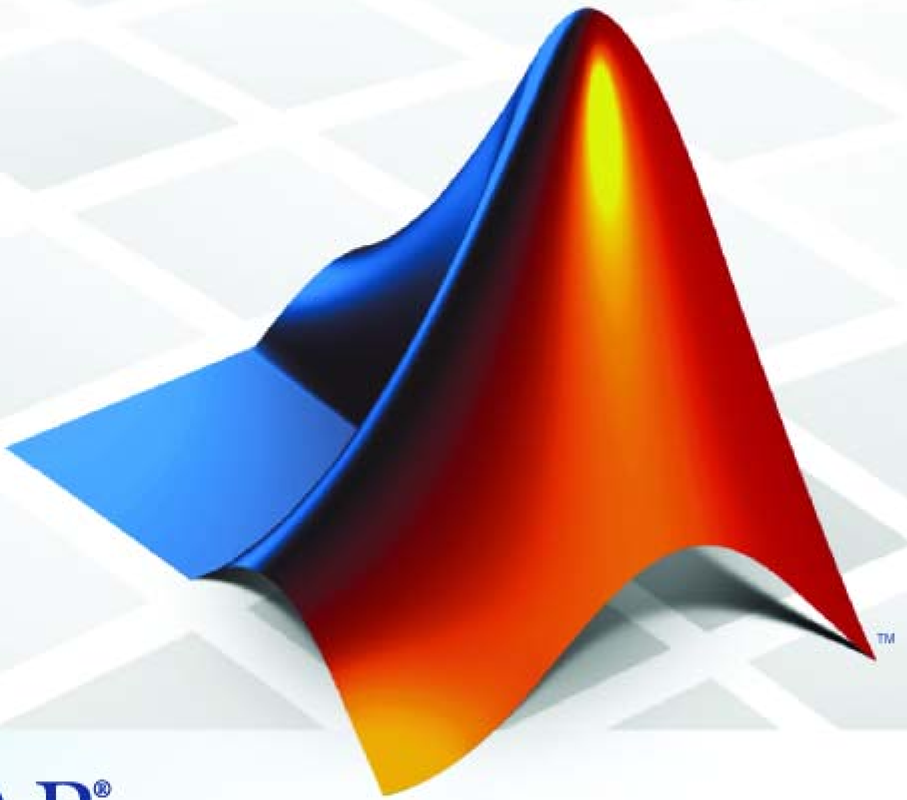
Stateflow® and Stateflow® Coder™ 7
Reference



MATLAB® & SIMULINK®

The MathWorks™
Accelerating the pace of engineering and science

How to Contact The MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support
`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

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

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

Stateflow® and Stateflow® Coder™ Reference

Revision History

March 2006	Online only	New for Version 6.4 (Release 2006a)
September 2006	Online only	Revised for Version 6.5 (Release R2006b)
September 2007	Online only	Rereleased for Version 7.0 (Release 2007b)
March 2008	Online only	Revised for Version 7.1 (Release 2008a)

Contents

Function Reference

Object Retrieval

`sfclipboard`	Get the Stateflow® clipboard object
`sfgco`	Get most recently selected objects in the Stateflow chart
`sfroot`	Get the Stateflow root object

Chart Creation

`sfnew`	Create a Simulink® model containing an empty Stateflow® block
`stateflow`	Create a Simulink model containing an empty Stateflow chart, and open the Stateflow library window

Chart Input/Output

`sfclose`	Close a Stateflow® chart
`sfopen`	Open the Stateflow machine
`sfprint`	Print graphical view of Stateflow charts
`sfsave`	Save the Stateflow machine in the current directory

Graphical User Interface

`sfdebugger`	Open the Stateflow® Debugger
`sfexplr`	Start the Model Explorer
`sflib`	Open the Stateflow library window

Help

`sfhelp`	Open the Stateflow® software online help

Functions — Alphabetical List

sfclipboard

Purpose Get the Stateflow® clipboard object

Syntax `object = sfclipboard`

Description `object = sfclipboard` returns a handle to the Stateflow clipboard object. Use the clipboard object to copy objects from one container object to another, as described in "Copying Objects" in the online Stateflow API Reference.

See Also `sfgco`, `sfnew`, `sfroot`, `stateflow`

Purpose

Close a Stateflow® chart

Syntax

```
sfclose
sfclose( 'Chart_Name' )
sfclose( Chart_Handle )
sfclose( 'All' )
```

Arguments

`'Chart_Name'`	Name of a Stateflow chart.
`Chart_Handle`	Handle to a Stateflow chart.
`'All'`	Literal string to close all open or minimized Stateflow charts.

Description

`sfclose` closes the current Stateflow chart.

`sfclose( 'Chart_Name' )` closes the Stateflow chart named **Chart_Name**.

`sfclose( Chart_Handle )` closes the Stateflow chart whose handle is `Chart_Handle`.

`sfclose( 'All' )` closes all open or minimized Stateflow charts.

See Also

`sfopen`, `sfnew`, `stateflow`

sfdebugger

Purpose

Open the Stateflow® Debugger

Syntax

```
sfdebugger
sfdebugger( 'Machine_Name' )
sfdebugger( Machine_Handle )
sfdebugger( Machine_Id )
```

Arguments

'Model_Name'	String name of a Stateflow machine.
Machine_Handle	Handle to a Stateflow machine.
Machine_Id	ID of a Stateflow machine.

Description

sfdebugger opens the Stateflow Debugger for the currently selected Stateflow machine.

sfdebugger(*'Machine_Name'*) opens the Stateflow Debugger for the Stateflow machine called **Machine_Name**.

sfdebugger(*Machine_Handle*) opens the Stateflow Debugger for the Stateflow machine whose handle is *Model_Handle*.

sfdebugger(*Machine_Id*) opens the Stateflow Debugger for the Stateflow machine whose `Id` is *Machine_Id*.

See Also

`sfexplr`, `sfhelp`, `sflib`

Purpose Start the Model Explorer

Syntax `sfexplr`

Description `sfexplr` starts the Model Explorer. For more information, see “The Model Explorer” in the online Simulink® software documentation.

See Also `sfdebugger`, `sfhelp`, `sflib`

sfgco

Purpose

Get most recently selected objects in the Stateflow® chart

Syntax

`object = sfgco`

Description

`object = sfgco` returns a handle or vector of handles to the most recently selected objects in a Stateflow chart, as follows:

If ...	sfgco returns ...
No Stateflow charts are open, or no open charts were edited or otherwise manipulated	Empty matrix
There is no selection list	Handle to the Stateflow chart most recently clicked
You select one object in a Stateflow chart	Handle to the selected object
You select multiple objects in a Stateflow chart	Vector of handles to the selected objects
You select multiple objects in multiple Stateflow charts	Vector of handles to the most recently selected objects in the most recently selected chart

See Also

`sfnew`, `stateflow`

Purpose Open the Stateflow® software online help

Syntax *`sfhelp`*

Description *`sfhelp`* opens the Stateflow software online help in the Help browser.

See Also `sfexplr`, `sfnew`, `sfprint`, `sfsave`, `stateflow`

Purpose

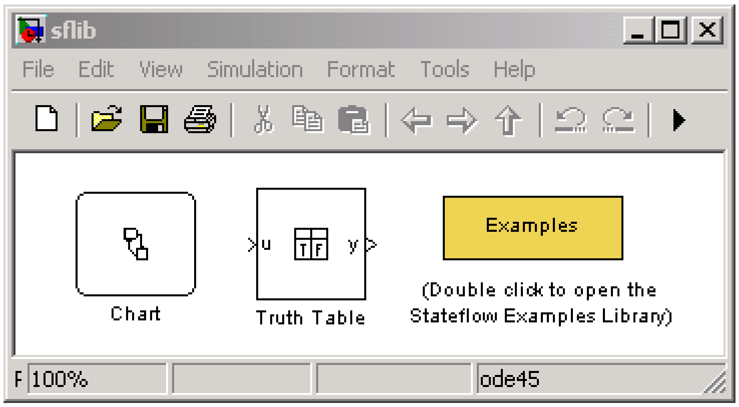
Open the Stateflow® library window

Syntax

`sflib`

Description

`sflib` opens the Stateflow library window:


sflib
File Edit View Simulation Format Tools Help
Chart
u
y
Truth Table
Examples
(Double click to open the Stateflow Examples Library)
100%
ode45

From this window, you can drag Stateflow charts and truth tables into Simulink® models, and access the Stateflow Examples Library.

See Also

`sfdebugger`, `sfexplr`, `sfhelp`

Purpose

Create a Simulink® model containing an empty Stateflow® block

Syntax

`Model_Handle = sfnew('-Chart_Type''Machine_Name')`

Arguments

`Model_Handle`	Handle to the new Simulink model that will contain the Stateflow block.
`Chart_Type`	Type of Stateflow block to add to the Simulink model. Enter • `'-Classic'` for a chart that implements full Stateflow chart semantics (default) • `'-Mealy'` for a chart that implements Mealy state machine semantics • `'-Moore'` for a chart that implements Moore state machine semantics • `'-TT'` for a truth table Optional.
`'Machine_Name'`	Name of the Stateflow machine (also becomes the model name). Optional.

Description

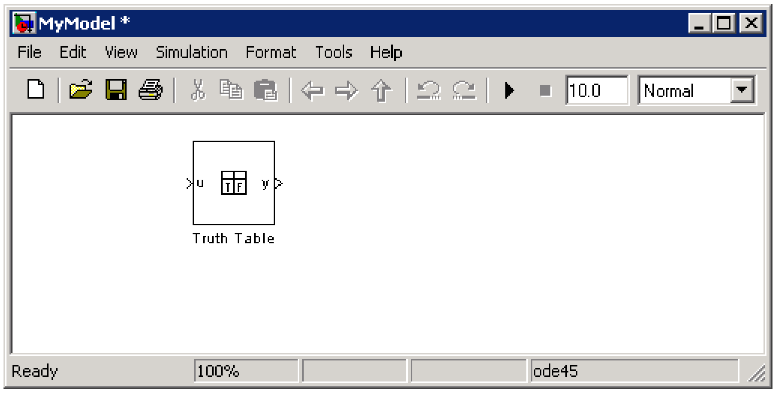
`Model_Handle = sfnew('-Chart_Type''Machine_Name')` returns the handle to a new model named **Machine_Name** that contains an empty Stateflow block of type *`Chart_Type`*, and opens the new model on your desktop. If *`Chart_Type`* is not specified, the default block is `Classic`. If *`Machine_Name`* is not specified, the default name is **untitled**.

Examples

Create a Simulink model called **MyModel** that contains an empty Stateflow truth table.

```
m = sfnew('-TT', 'MyModel')
```

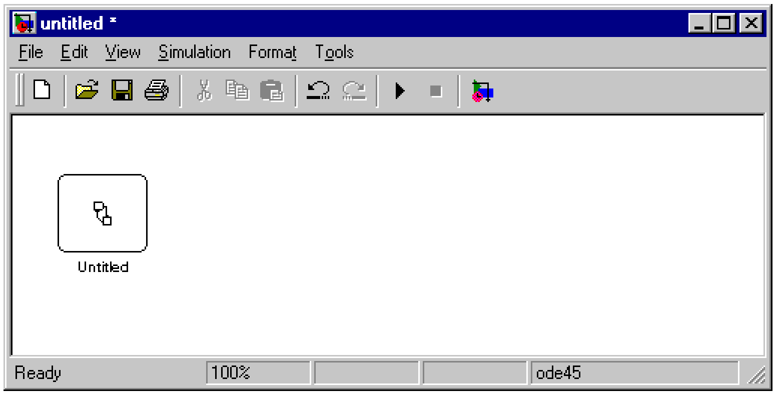
The new model looks like this:



Create an untitled Simulink model that contains an empty Stateflow chart.

```
m = sfnew
```

The new model looks like this:



See Also

`sfhelp`, `sfprint`, `sfroot`, `sfsave`, `stateflow`

Purpose Open the Stateflow® machine

Syntax `sfopen`

Description `sfopen` prompts you for an `.mdl` file and opens the one that you select from your file system.

See Also `sfclose`, `sfdebugger`, `sfexplr`, `sflib`, `sfnew`, `stateflow`

sfprint

Purpose Print graphical view of Stateflow® charts

Syntax

```
sfprint
sfprint( objects, format, outputOption, printEntireChart )
```

Arguments

`objects`

Any of the following object identifiers:

- String name of a Stateflow chart, or Simulink® model, system, or block
- Handle to a Stateflow chart, or Simulink model, system, or block
- Cell array of names of and/or handles to a Stateflow chart, or Simulink model, system, or block
- Vector of handles to a Stateflow chart, or Simulink model, system, or block
- Simulink model construction commands `gcb`, `gcbh`, or `gcs`

`format`

Optional literal string that specifies the print destination:

- `'default'` prints to a default printer
- `'ps'` generates a PostScript file
- `'psc'` generates a color PostScript file
- `'eps'` generates an Encapsulated PostScript file
- `'epsc'` generates a color Encapsulated PostScript file
- `'tif'` generates a TIFF file
- `'jpg'` generates a JPEG file
- `'png'` generates a PNG file
- `'meta'` saves the Stateflow chart image to the clipboard as a metafile (for Windows® operating systems only)
- `'bitmap'` saves the Stateflow chart image to the clipboard as a bitmap (for Windows operating systems only)

`outputOption`	Optional string that specifies an output file or printer: • String that specifies the name of a file to which to write (file will be overwritten if more than one chart is printed) • `'promptForFile'` prompts for file name interactively • `'printer'` sends output to default printer (use only with `'default'`, `'ps'`, or `'eps'` formats) • `'file'` sends output to a default file, specified as *<path to object>.<device extension>* • `'clipboard'` copies output to the clipboard
`printEntireChart`	Optional Boolean argument: • `1` (default) prints complete charts • `0` prints current view of charts

Description

`sfprint` prints the current Stateflow chart to a default printer.

`sfprint(` *`objects`*`,` *`format`*`,` *`outputOption`*`,` *`printEntireChart`* `)` prints all Stateflow charts identified in *`objects`* in the specified *`format`* to the file or printer specified in *`outputOption`*. Prints a complete or current view of charts as specified in *`printEntireChart`*. If the *`format`* argument is absent, the format defaults to `'ps'` and output is sent to the default printer. If the *`outputOption`* argument is absent, the name of the Stateflow chart in the current directory is used as the output file name.

Examples

Print the complete chart whose handle is *id* to a TIFF file called **myFilename**.

```
sfprint(id, 'tif', 'myFilename')
```

Print all Stateflow charts in the current system as a PostScript file to the default printer.

```
sfprint(gcs)
```

Print the current Stateflow block to a JPEG file whose name is specified by the user interactively.

```
sfprint(gcb, 'jpg', 'promptForFile')
```

Print the current view of all Stateflow charts in the current system in PNG format using default file names.

```
sfprint(gcs, 'png', 'file', 0)
```

Assume that you loaded a Simulink model named **myModel** that has two charts named **Chart1** and **Chart2**. Further, both **Chart1** and **Chart2** are represented by the Stateflow chart objects **ch1** and **ch2**, respectively.

Command	Result
`sfprint('myModel')`	Prints the graphical view of both **Chart1** and **Chart2** to the default printer.
`sfprint('myModel','ps')`	Prints the graphical view of both **Chart1** and **Chart2** to a PostScript file.
`sfprint(ch1.Id,'psc')`	Prints the graphical view of **Chart1** to a color PostScript file.
`sfprint([ch1.Id, ch2.Id])`	Prints the graphical views of both **Chart1** and **Chart2** to the default printer.

See Also `sfhelp`, `sfnew`, `sfsave`, `stateflow`

Purpose Get the Stateflow® root object

Syntax `object = sfroot`

Description `object = sfroot` returns the handle to the top-level object in the Stateflow machine hierarchy of objects. Use the root object to access all other objects in Stateflow charts, as described in “Accessing the Model Object” in the online Stateflow API Reference.

See Also `sfnew`, `sfgco`, `sfclipboard`, `stateflow`

Purpose

Save the Stateflow® machine in the current directory

Syntax

```
sfsave
sfsave( Model_Handle )
sfsave( Model_Handle, 'New_Model_Name' )
sfsave( Machine_Handle )
sfsave( 'Model_Name' )
sfsave( 'Defaults' )
```

Arguments

Model_Handle	Handle to a Simulink® model that contains a Stateflow block.
'New_Model_Name'	Name to assign to the model being saved.
Machine_Handle	Handle to a Stateflow machine.
'Model_Name'	Name of a Simulink model that contains a Stateflow block.
'Defaults'	Literal string used to save current settings as defaults.

Description

`sfsave` saves the current Stateflow machine in the current directory.

`sfsave(` *`Model_Handle`* `)` saves the Simulink model specified by *`Model_Handle`* in the current directory.

`sfsave(` *`Model_Handle`*`,` *`'New_Model_Name'`* `)` saves the Simulink model specified by *`Model_Handle`* as **New_Model_Name** in the current directory.

`sfsave(` *`Machine_Handle`* `)` saves the Simulink model that contains the Stateflow machine specified by `Machine_Handle` in the current directory.

`sfsave(` *`'Model_Name'`* `)` saves the Simulink model called **Model_Name** in the current directory.

`sfsave(` *`'Defaults'`* `)` saves the settings of the current Stateflow machine as defaults.

Examples

Save the model whose handle is `m` as **MyModel** in the current directory.

```
sfsave(m, 'MyModel')
```

Save the model that contains a Stateflow machine whose handle is `sf` in the current directory.

```
sfsave(sf)
```

See Also

`sfclose`, `sfnew`, `sfopen`, `sfprint`

stateflow

Purpose

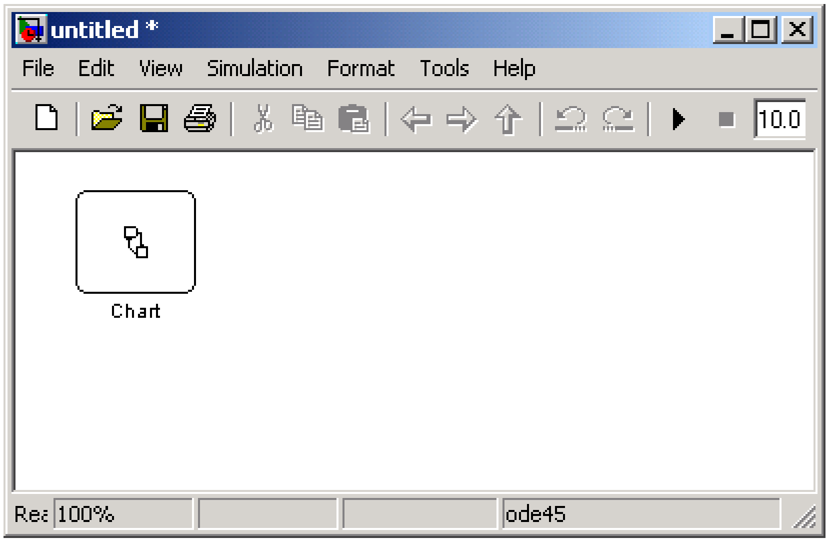
Create a Simulink® model containing an empty Stateflow® chart, and open the Stateflow library window

Syntax

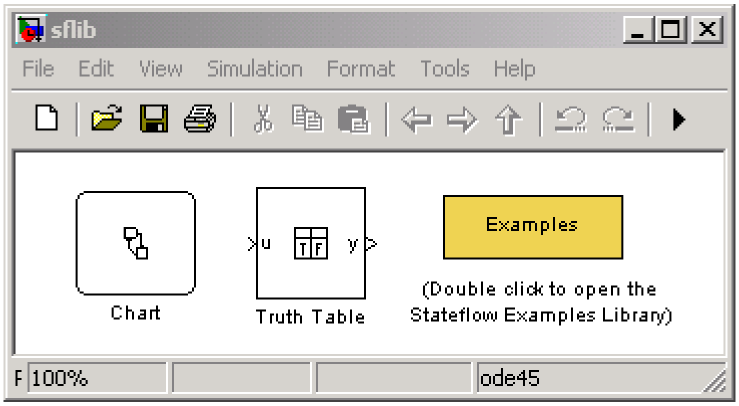
`stateflow`

Description

`stateflow` creates a new Simulink model that is preconfigured with an empty Stateflow chart:



The function also opens the Stateflow library window:


sflib
File Edit View Simulation Format Tools Help
Chart
Truth Table
u
y
Examples
(Double click to open the Stateflow Examples Library)
100%
ode45

From this window, you can drag other Stateflow blocks into your Simulink model and access the Stateflow Examples Library.

See Also

`sflib`, `sfnew`, `sfroot`

3

Block Reference

Stateflow Chart

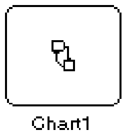
Purpose

A version of a finite state machine for controlling a physical plant

Library

Stateflow®

Description

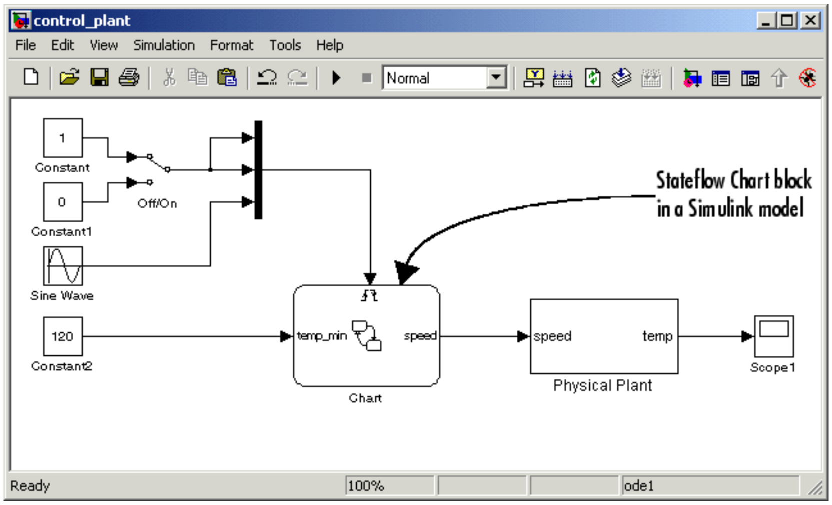


A *finite state machine* is a representation of an event-driven (reactive) system. In an event-driven system, the system responds by making a transition from one state (mode) to another prescribed state in response to an event, provided that the condition defining the change is true.

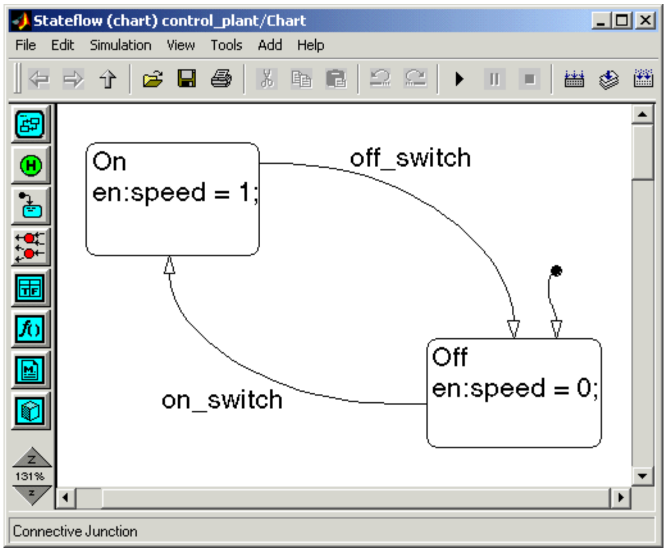
A Stateflow chart is a graphical representation of a finite state machine, where *states* and *transitions* form the basic building blocks of the system. You can also represent stateless flow graphs. To add your control logic to a Simulink® model, use a Stateflow block.

You can use Stateflow charts to control a physical plant in response to events such as a temperature or pressure sensor, or clock or user-driven events. For example, you can use a state machine to represent the automatic transmission of a car. The transmission has a number of operating states: park, reverse, neutral, drive, and low. As the driver shifts from one position to another the system makes a transition from one state to another, for example, from park to reverse.

The following diagram shows a simple Simulink model that has a Stateflow block named Chart (default) that responds to input from a manual switch:



If you double-click the Stateflow block in the Simulink model, the Stateflow chart that programs the Stateflow block appears in the Stateflow Editor.


Stateflow (chart) control_plant/Chart
File Edit Simulation View Tools Add Help
On
en:speed = 1;
off_switch
Off
en:speed = 0;
on_switch
131%
Connective Junction

During simulation of the Simulink model, you can interactively debug Stateflow charts in animation mode. Stateflow charts generate efficient C code for simulation targets, and also for embedded targets.

For an introduction to using Stateflow charts in Simulink models, see the Stateflow Getting Started Guide.

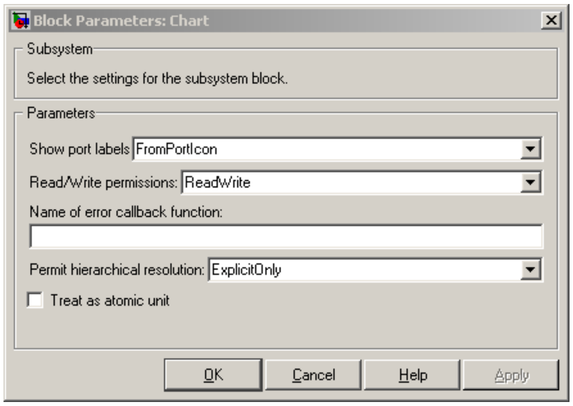
Data Type Support

The Stateflow block accepts inputs of any type including two-dimensional matrices and fixed-point data. Floating-point inputs pass through the block unchanged. `Boolean` inputs are treated as `uint8` signals.

For a discussion on the variable types supported by Embedded MATLAB™ functions in Simulink models, refer to the Simulink software documentation.

You can declare local data of any type or size.

Parameters and Dialog Box


Block Parameters: Chart

Subsystem

Select the settings for the subsystem block.

Parameters

Show port labels: FromPortIcon

Read/Write permissions: ReadWrite

Name of error callback function:

Permit hierarchical resolution: ExplicitOnly

Treat as atomic unit [ ]

Note It is highly recommended that the default settings for the block parameters of an Embedded MATLAB Function block not be changed.

Characteristics

Direct Feedthrough	Yes
Sample Time	Specified in the **Sample time** parameter
Scalar Expansion	N/A
Dimensionalized	Yes
Zero Crossing	No

Truth Table

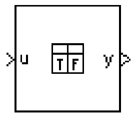
Purpose

Represents logical decision-making behavior with conditions, decisions, and actions.

Library

Stateflow®

Description



Truth Table

The Truth Table block is an Embedded MATLAB™ truth table function that you can add to a Simulink® model directly. The Truth Table block requires a Stateflow software license.

There are several advantages to adding a Truth Table block directly to a Simulink model instead of calling truth table functions from a Stateflow chart:

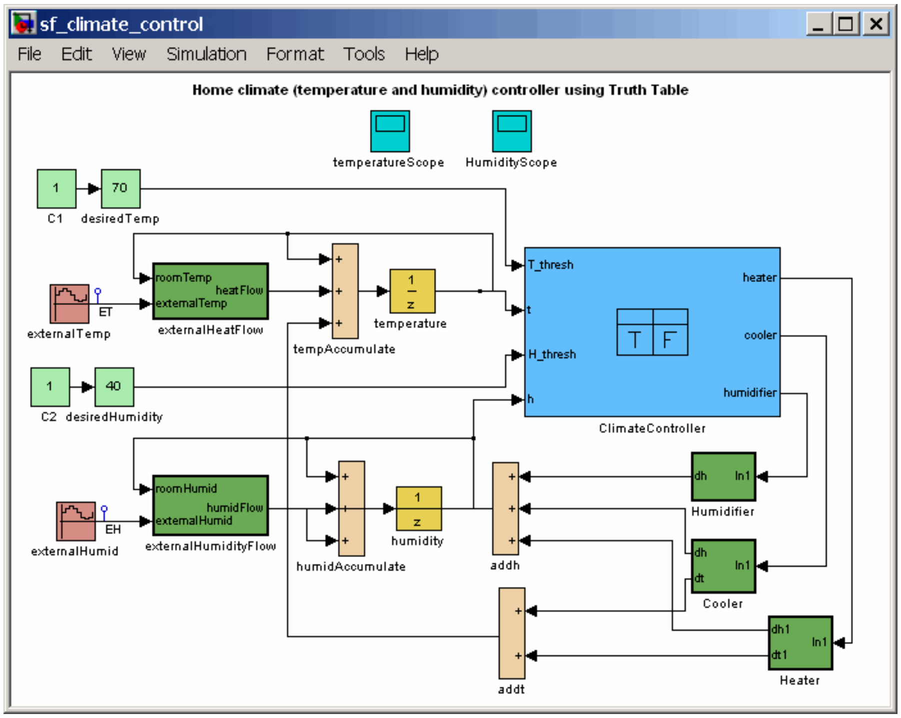
- It is a more direct approach, especially if your model requires only a single truth table.
- You can define truth table inputs and outputs to have inherited types and sizes.

The Truth Table block supports the Embedded MATLAB language subset for programming conditions and actions, and generates content as Embedded MATLAB code. Embedded MATLAB functions work with a subset of the MATLAB® language that is optimized for generating embeddable C code.

As a result, you can take advantage of Embedded MATLAB tools to debug your Truth Table block during simulation. For more information, see "Debugging an Embedded MATLAB Function".

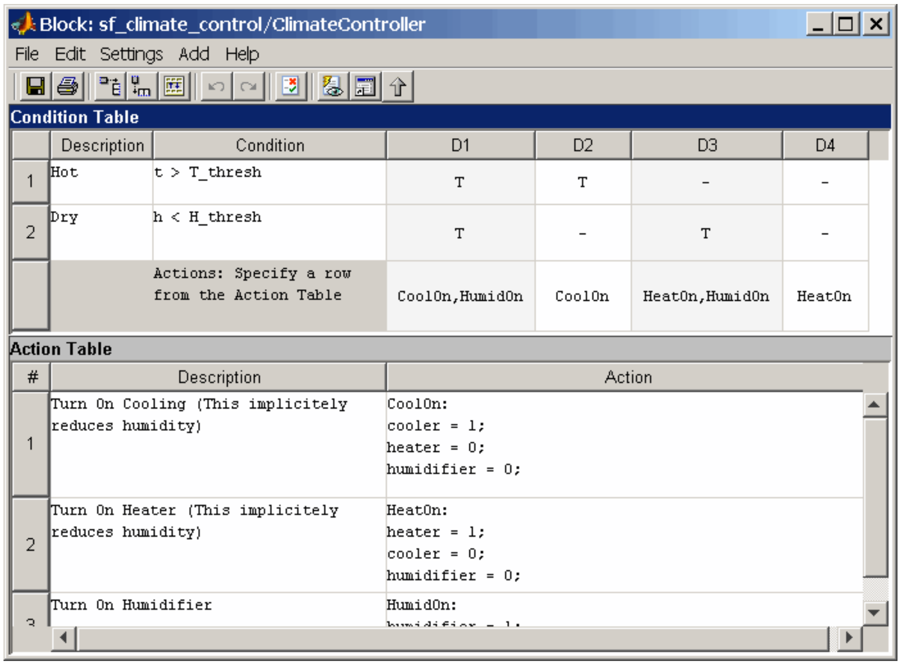
For purely logical behavior, truth tables are easier to program and maintain than graphical functions. Truth tables also provide diagnostics that indicate whether you have too few (underspecified) or too many (overspecified) decisions for the conditions you specify. For an introduction to truth tables, see “Truth Table Functions”.

This figure shows a Simulink model (`sf_climate_control.mdl`) of a home environment controller that attempts to maintain a selected temperature and humidity. The model has a Truth Table block (`ClimateController`) that responds to changes in room temperature (input `t`) and humidity (input `h`).



Truth Table Editor

If you double-click the Truth Table block in the Simulink model, the Truth Table Editor opens to display its conditions, actions, and decisions. Here is the display for the Truth Table block named `ClimateController`.


Block: sf_climate_control/ClimateController
File Edit Settings Add Help

Condition Table

	Description	Condition	D1	D2	D3	D4
1	Hot	t > T_thresh	T	T	-	-
2	Dry	h < H_thresh	T	-	T	-
		Actions: Specify a row from the Action Table	CoolOn,HumidOn	CoolOn	HeatOn,HumidOn	HeatOn

Action Table

#	Description	Action

1	Turn On Cooling (This implicitely reduces humidity)	CoolOn:
		cooler = 1;
		heater = 0;
		humidifier = 0;

2	Turn On Heater (This implicitely reduces humidity)	HeatOn:
		heater = 1;
		cooler = 0;
		humidifier = 0;

3	Turn On Humidifier	HumidOn: humidifier = 1;

Note how the inputs `t` and `h` are used to define the conditions, and the outputs `heater`, `cooler`, and `humidifier` are used to define the actions for this Truth Table block. For more details, refer to the demo for this model.

Using the Truth Table Editor, you can:

- Enter and edit conditions, actions, and decisions
- Add or modify Stateflow data and ports using the Ports and Data Manager
- Run diagnostics to detect parser errors
- View generated content after simulation

For more information about the Truth Table Editor, see “Truth Table Editor Operations”.

Ports and Data Manager

If you want to add or edit data in a Truth Table block, open the Ports and Data Manager by clicking the **Edit Data/Ports** button in the Truth Table Editor toolbar:

The Ports and Data Manager lets you add the following elements to a Truth Table block:

Element	Tool	Description
Data		You can add the following types of data: • Local • Constant • Parameter • Data store memory

Element	Tool	Description
Input trigger		An *input trigger* causes a Truth Table block to execute when a Simulink control signal changes or through a Simulink block that outputs function-call events. You can add the following types of input triggers: • Rising edge • Falling edge • Either rising or falling edge • Function call For more information, see "Defining Input Events".
Function call output	f()	A *function call output* triggers a function call to a subsystem. For more information, see "Function-Call Subsystems" in the Simulink software documentation.

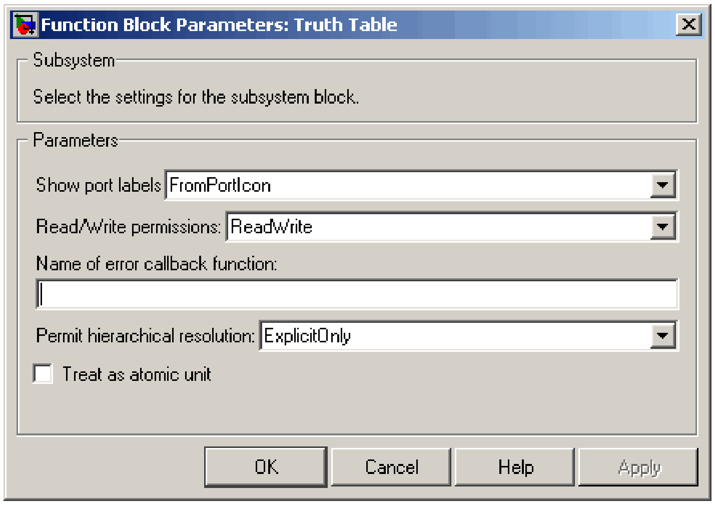
Data Type Support

The Truth Table block accepts signals of any data type supported by Simulink models, including fixed-point data types and frame-based signals. Truth Table blocks work with frame-based signals in the same way as Embedded MATLAB Function blocks (see "Working with Frame-Based Signals" in the Simulink software documentation).

For a discussion of data types supported by Simulink models, refer to the Simulink software documentation.

Parameters and Dialog Box

Right-click over a Truth Table block, and from the submenu, select **Subsystem Parameters**.


Function Block Parameters: Truth Table

Subsystem

Select the settings for the subsystem block.

Parameters

Show port labels: FromPortIcon

Read/Write permissions: ReadWrite

Name of error callback function:

Permit hierarchical resolution: ExplicitOnly

Treat as atomic unit: ☐

Characteristics

Direct Feedthrough	Yes
Sample Time	Specified in the **Sample time** parameter
Scalar Expansion	N/A
Dimensionalized	Yes
Zero Crossing	No

Index