Visual Solutions, Inc.

VisSim/State Charts User's Guide Version 8.0

Copyright © 2010 Visual Solutions, Inc. 
All rights reserved.

Visual Solutions, Inc.
487 Groton Road
Westford, MA 01886

Trademarks
VisSim, VisSim/Analyze, VisSim/CAN, VisSim/C-Code, VisSim/C-Code Support Library source, VisSim/Comm, VisSim/Comm C-Code, VisSim/Comm Red Rapids, VisSim/Comm Turbo Codes, VisSim/Comm Wireless LAN, VisSim/Fixed-Point, VisSim/Knobs & Gauges, VisSim/Model-Wizard, VisSim/Motion, VisSim/Neural-Net, VisSim/OPC, VisSim/OptimizePRO, VisSim/Real-TimePRO, VisSim/State Charts, VisSim/Serial, VisSim/UDP, VisSim Viewer, and flexWires are trademarks of Visual Solutions. All other products mentioned in this manual are trademarks or registered trademarks of their respective manufacturers.

Copyright and use restrictions
The information in this manual is subject to change without notice and does not represent a commitment by Visual Solutions. Visual Solutions does not assume any responsibility for errors that may appear in this document.

No part of this manual may be reprinted or reproduced or utilized in any form or by any electronic, mechanical, or other means without permission in writing from Visual Solutions. The Software may not be copied or reproduced in any form, except as stated in the terms of the Software License Agreement.

Acknowledgements
The following engineers contributed significantly to preparation of this manual: Mike Borrello, Allan Corbeil, and Richard Kolk.
# Contents

## Introduction
- What is a state chart ................................................................. 1
- Basic state chart elements .................................................. 2
- Creating a simple state chart ........................................ 2
  - Inserting states ........................................................................ 3
  - Creating transitions .......................................................... 4
  - Defining state chart variables ....................................... 5
  - Configuring states ................................................................ 6
  - Denoting transition specifications .................................... 10
- Setting up the VisSim diagram to interact with the state chart .................. 12
- Simulating the state chart .................................................. 13

## Working with State Charts
- Creating state charts ................................................................. 15
  - Naming a state chart .......................................................... 15
  - Applying protection to state chart ................................. 16
  - Coloring state chart blocks ........................................... 16
  - Setting up state chart variables .................................... 16
  - Setting up the State Id data type .................................. 18
  - Annotating state charts .................................................. 21
- Working with states and pseudo-states ........................................ 22
  - Inserting a state or pseudo-state ..................................... 23
  - Moving and resizing states ............................................... 23
  - Navigating through submachine state hierarchy .......... 23
  - Specifying state names .................................................... 24
  - Coloring states and pseudo-states .................................. 24
  - Associating behaviors with states ................................. 25
  - Working with regions in composite states .................. 27
- Working with transitions ....................................................... 28
  - Drawing a transition .......................................................... 28
  - Bending and moving transitions ..................................... 28
  - Deleting transitions .......................................................... 29
  - Defining transition specifications ................................... 29
  - Determining transition evaluation order ......................... 30
  - Setting breakpoints for transitions ............................... 31
  - Logging messages for transitions .................................. 31
- Working with triggers ............................................................. 31
  - Using triggers to control a state chart from VisSim .......... 31
  - Using the SendT () function to activate triggers .......... 32
  - Referencing triggers .......................................................... 33
  - Using deferred triggers .................................................... 34

## Evaluating State Charts

---

Version 8.0 VisSim/State Charts User’s Guide

Contents • iii
State chart evaluation ................................................................. 37
Event processing: run-to-completion step ........................................ 37
Evaluating states .............................................................................. 38
  Active and inactive states ......................................................... 38
  Entering and exiting states ....................................................... 38
  Entering and exiting composite states ........................................ 38
Evaluating transitions ................................................................. 40
  Transition firing priority ......................................................... 40
  Transition selection algorithm .................................................. 40
  High-level transitions .............................................................. 40
  Compound transitions ............................................................. 40
  Completion transitions ............................................................ 41
  Enabled transitions ................................................................. 41
  Guards ...................................................................................... 41
  Transition execution sequence ................................................ 42
  Transition kinds ...................................................................... 42

State Chart Elements ..................................................................... 43
States ............................................................................................. 43
  simple state ............................................................................. 43
  composite state ................................................................. 46
  submachine state .................................................................. 49
  final state ............................................................................ 53
Pseudo-states ............................................................................... 54
  choice .................................................................................. 54
  deep history ......................................................................... 56
  entry points and exit points ................................................... 57
  fork ...................................................................................... 58
  history ................................................................................ 58
  initial state indicator ............................................................. 59
  join ...................................................................................... 60
  junction ............................................................................... 60
  terminate ............................................................................. 61
Transition ..................................................................................... 62
Blocks ......................................................................................... 64
  comment ............................................................................. 64
  convert ID .......................................................................... 64
  function .............................................................................. 65
  label .................................................................................. 66
  state chart ................................................................. 66
  state ID ................................................................. 69
  trigger .............................................................................. 70
Commands ................................................................................... 70
  Check Integrity ................................................................. 70
  State Id Lookup .................................................................. 71
  Breakpoints ..................................................................... 71
  Preferences ..................................................................... 71

Debugging State Charts .............................................................. 73
Visual debugging ........................................................................... 73
Functional debugging ............................................................... 73
  Using breakpoints ............................................................... 73
  Logging messages ............................................................... 75
Chart-wide debugging ............................................................... 77
Error Navigation panel ................................................................. 78
    Jumping to the error location .................................................. 78
    Error and warning messages .................................................. 78

Index 81
Introduction

This section contains…

What is a state chart

A state chart is a graphical representation of a finite-state machine, that is, a system with a defined number of states, where transition from one state to another occurs when certain conditions are met. An example of a simple state chart is shown below:

You can create state charts anywhere within a VisSim diagram, and you can easily exchange data between the state chart and the continuous portion of a diagram. In addition, you can interactively simulate the state chart diagram or generate code to be run on an embedded target.

The continuous portion of the VisSim diagram can control state chart execution with triggers.
Basic state chart elements

A state chart diagram consists of five main elements:

- **States** that define behavior and may produce actions
- **Transitions** that are movement from one state to another
- **Pseudo-states**, like the initial state indicator, that are used to connect transitions to more complex state transition paths
- Behaviors that are associated with **transitions** and **state actions**
- **Triggers** that are either externally or internally generated

Received input events act as triggers that cause an evaluation of the rules that govern the transition from the current state to other states.

State behavior and transition specifications are specified as C expressions. If you are unfamiliar with C, we recommend *C: A Software Engineering Approach*.

Creating a simple state chart

To introduce VisSim/State Charts, you will build a simple three-state pump within a VisSim model. The three states are: control OFF; control ON to pump water into the tank; and control ON to pump water out of the tank.

During simulation, the pump controls the water level in a tank by keeping the water within a specified minimum and maximum levels. An interactive ON/OFF button controls the pump. The tank drains completely if control is OFF, but it will never overflow.

**To create a container for the 3-state pump**

A **state chart** block is a container block inside which you build a state chart.

1. Open the VisSim model
   C:\VisSim8\Examples\StateCharts\stateChartTank2.vsm.

2. Right click the mouse over the **Tank Level Control** compound block.
3. Create a container for the state chart by choosing **State Charts > state chart**.
4. Click the mouse anywhere in the VisSim work area to place the state chart block. You will see the following:

![State Chart Block](image)

5. Click the right mouse button over the state chart block to enter the state chart design environment.

**Inserting states**

For this example, you will use an initial state indicator and three simple states to represent the three states of the pump.

**To create a three-state system**
1. Choose **Start Charts > initial state indicator**.
2. Click the mouse anywhere in the work area to create the following:

![Initial State](image)

3. Choose **State Charts > state**.
4. Click the mouse anywhere in the work area to create the following:

![State Block](image)

5. To create a three-state system, insert two more state blocks into the work area to create the following:

![Three States](image)
Creating transitions

A transition is a relationship between two states that indicates when an object can move the focus of control on to another state once certain conditions are met.

A transition is represented as line between two states. An arrowhead at one end of the transition indicates the direction of the transition.

When a state has multiple transitions exiting it, the transitions are numbered to indicate evaluation order.

After you create a transition, you can define a transition specification for it.

To create transitions

1. Position the cursor over the edge of a source state. The cursor shape changes to a pencil ( ).
   
   **Note:** If the crosshair has arrow heads, moving the mouse moves the state rather than draws the arc.

2. Depress the mouse and drag into the target state body.

3. Release the mouse button.

4. The transition will appear as a straight line from the source state to the target state.

5. Repeat this exercise to create the following:
Notice that when multiple transitions are coming from a given state, VisSim labels them according to evaluation order. Transitions with lower numbers have higher priorities.

**To bend and move transitions**

1. Click the mouse over a transition that you want to bend or move. The transition line turns purple, indicating that you can edit it.
   - To bend the transition arc, point to the transition line and drag the mouse.
   - To move where the transition line connects to a state, position the cursor over the end of the transition line you want to move. When the cursor changes to a 4-arrow crosshair, depress and drag the mouse.

2. Repeat step 1 for each transition arc until you have the following:

![Diagram of state chart transitions](image)

**Defining state chart variables**

To exchange data between the VisSim model and the state chart, you use variables. State chart variables are declared in the State Chart Block Properties dialog box. In this example, you will use four input variables and three output variables.

**To define input and output variables**

1. Add four input pins and three output pins to the state chart block using the **Edit > Add Connector** command (or toolbar button). Your state chart block will look like this:

![Diagram of state chart block with input and output pins](image)

**Note:** Activate **View > Connector Labels** to see the pin names.
2. To edit the attributes for each variable, hold down the **CTRL** key and **right click** the mouse over the **state chart** block.

3. Click on the **Activity Manager** tab.

![State Chart Block Properties](image)

4. To edit a variable, double click over each **attribute** (name, type, and scope) to make the changes shown below:

![State Chart Block Properties](image)

5. Click on the **OK** button.

**Configuring states**

Configuring a state includes naming it and optionally assigning a behavior (C code) to selected actions. A state has three pre-defined actions (entry, exit, and do) and any number of inner transitions that are fired by triggers.

**To configure a state**

1. Right click the mouse in the **title bar** of **State1**.

The State Properties dialog box appears.
2. Do the following:
   - Under **Name**, enter **Init**.
   - Under **Color**, select a color for the border and a color for the header.

3. Click on the **Activity Manager** tab to select an action and enter behaviors.
4. Under **Actions**, select **Entry** and click on the **Add Action** button.

The following updates to the dialog box are made:
5. Under **Edit Behavior**, replace /* TODO: Insert your code here */ with the C code to indicate the pump is OFF, as shown below.

![State Properties window](image)

**Note:** If you are unfamiliar with the C language, refer to *C: A Software Engineering Approach*.

6. Click on the **OK** button.

Your state chart will look like this:
7. Repeat steps 1 – 6 for State2 and State3 so that your diagram looks like this:

![Diagram](image)

**Denoting transition specifications**

A transition has the following basic format:

\[ \text{trigger(s) [guard]} / \text{behavior} \]

Together, the triggers and guard represent a logical expression that evaluates to TRUE or FALSE. When the logical expression is TRUE, the transition is taken to the next state. When it is FALSE, another transition is tested; if there are no other transitions, the state of origin remains active.

For more information, see “Defining transition specifications.

For this simple example, there are no triggers or behaviors in the transition specification, only guards.

**To define transition specifications**

1. Point to the transition between Init state and Fill state and click the mouse.

   The transition turns purple.

2. Click the right mouse button over the transition.

   The following Pop-Up menu appears:

   ![Pop-Up menu](image)

3. Click on Properties.
The Transition Properties dialog box appears.

4. Under **Edit Behavior**, enter the **guard** using the C language. Enclose the code in square brackets and terminate with a forward slash.

5. Click on the **OK** button.

6. Repeat these steps to add the following guards to the remaining transitions.

<table>
<thead>
<tr>
<th>Transition</th>
<th>Guard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill to Drain</td>
<td>![running</td>
</tr>
<tr>
<td>Drain to Init</td>
<td>![running &amp;&amp; level &lt;=0]/</td>
</tr>
<tr>
<td>Drain to Fill</td>
<td>![running &amp;&amp; level &lt; minLevel]/</td>
</tr>
</tbody>
</table>
Your state chart will look like this:

---

**Setting up the VisSim diagram to interact with the state chart**

For the state chart to exchange data with the VisSim model, you must create the pump dynamics, and link the dynamics via variables to the state chart, as shown below:

The Pump compound block contains an integrator block to define the pump logic. The round input pin enables the logic.
Simulating the state chart

Before simulating the state chart, set your simulation parameters using the Simulate > Simulation Properties command.

To simulate the state chart

- Choose Simulate > Go, press the toolbar button.

In the state chart, the active state is highlighted to show it is executing. In the top level VisSim model, the two plots State Chart and Tank Level monitor the pump and the tank level, respectively.

Note: At the top level of the diagram, the button block wired into the Tank Level Control block must be turned ON.
Working with State Charts

This section contains…

Creating state charts

You create a state chart by inserting a state chart block into a VisSim diagram and then populating the state chart with state chart elements, including states, pseudo-states, transitions, and a subset of VisSim blocks.

A VisSim diagram can contain multiple state chart blocks.

To create a state chart

1. Choose State Charts > state chart.
2. Click the mouse in the work area.

<table>
<thead>
<tr>
<th>To</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open the state chart</td>
<td>Double-click on the state chart block</td>
</tr>
<tr>
<td>Access state chart properties</td>
<td>CTRL+right-click on the state chart block</td>
</tr>
</tbody>
</table>

Naming a state chart

You specify state chart names in the Options tab for the State Chart Properties dialog box. A state chart name appears on the state chart block.

A state chart name can consist of any number of alphanumeric characters. For long names, press CTRL+ENTER to continue the name on a new line. By default, state chart blocks are named State ChartN, where N is a unique number.
Applying protection to state chart

You apply protection to a state chart in the Options tab for the State Chart Properties dialog box.

Restricting access to a state chart is similar to restricting access to a block diagram. You have the choice of password protection, read-only password protection, and read-only requested protection.

To restrict or change the access of a state chart diagram

1. Do one of the following:
   - To lock the state chart, activate the Locked parameter and enter a password in the Password box.
   - To make the state chart read-only, activate the Read Only parameter and enter a password in the Password box.
   A password can contain up to 10 characters and can include any combination of letters and numbers. VisSim echoes an asterisk (*) for each character you type. Passwords are case sensitive.

2. Click on the OK button, or press ENTER.

Coloring state chart blocks

You apply color to a state chart in the Options tab for the State Chart Properties dialog box.

You can change the color of a state chart block to make it more visible in your block diagram.

To color state chart blocks

- Activate the Set Color option and click on Select to choose a color.

Setting up state chart variables

The Data Browser tab for the State Chart Properties dialog box lists the current variables and variable attributes for the state chart diagram. You can add and delete variables, and update their attributes through this window.

To add and delete variables

There are several ways to add and delete variables.

- In the Data Browser window, use the following buttons:
• To add input and output variables, use the toolbar button.

![State Chart Diagram]

• To delete input and output variables, use the toolbar button.

To edit variables

• In the Data Browser window, double-click over the attribute to be changed.
  • For **Type**, **Scope**, and **Pin**, a drop-down menu appears in which to choose an attribute value.
  • For **Name**, **Default Value**, and **Comment**, a box surrounds the variable attribute indicating Edit mode. Enter the new attribute value and press ENTER.

**Defining variable data types**

You can choose from these built-in data types.

<table>
<thead>
<tr>
<th>Date type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>8-bit character</td>
</tr>
<tr>
<td>unsigned char</td>
<td>8-bit unsigned character</td>
</tr>
<tr>
<td>short</td>
<td>16-bit signed integer</td>
</tr>
<tr>
<td>unsigned short</td>
<td>16-bit unsigned integer</td>
</tr>
<tr>
<td>int</td>
<td>32-bit signed integer</td>
</tr>
<tr>
<td>long</td>
<td>32-bit signed integer</td>
</tr>
<tr>
<td>unsigned long</td>
<td>32-bit unsigned integer</td>
</tr>
<tr>
<td>float</td>
<td>Single-precision floating point</td>
</tr>
<tr>
<td>double</td>
<td>Double-precision floating point</td>
</tr>
<tr>
<td>trigger</td>
<td>Special data type for state chart triggers. See Using SentT() function to activate triggers.</td>
</tr>
<tr>
<td>state ID</td>
<td>Integer whose value is equal to VisSim-internal identifier of the active state. For more information, see Setting up state ID data type.</td>
</tr>
</tbody>
</table>
Defining variable scope

The variable scope determines how variables are referenced. There are five scope attributes.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>Constant value used inside the state chart diagram. You can only assign it a default value.</td>
</tr>
<tr>
<td>External</td>
<td>For code generation.</td>
</tr>
<tr>
<td>Input</td>
<td>Passes information to the state chart diagram. You can assign it a type, pin number, and default value.</td>
</tr>
<tr>
<td>Local</td>
<td>Value used inside the state chart diagram. TBA: What are restrictions (if any) for data type of local variables.</td>
</tr>
<tr>
<td>Output</td>
<td>Passes information from the state chart to the VisSim diagram. You can assign it a type, pin number, and default value.</td>
</tr>
</tbody>
</table>

Setting up the State Id data type

If you select the special data type State Id, you can produce a unique numeric identifier of the currently active state. This can be plotted during the course of a simulation to trace system behavior.

If there are nested regions in the state chart, the state chart can have multiple active states: one in each active region.

To configure a state Id data type in multiple active regions

1. In the Data Browser window for the State Chart Properties dialog box, select the variable state ID. For example:

2. Click on the (Configure State ID) button.

3. The Select State or Region window appears. For example:
4. Click on the **plus (+)** signs to expand the state chart tree. For example:

![State Chart Expanded](image1)

5. Select a state or region. For example:

![State Chart Region](image2)
6. Click on the **OK** button.

The selected state or region will be displayed under Default Value in the Data Browser. For example:

**Looking up a state with the state ID**

You use the state ID Lookup command to display the state name that corresponds to a given state ID numeric value. You can also use the **convert ID** block to perform this same function.

**To convert the state ID numeric number**

1. Choose **State Charts > state ID Lookup**.
The State Lookup dialog box appears.

2. In the State Id box, enter the numeric value and click on Lookup.
   If there is more than one state chart in the diagram, the State Lookup dialog box expands to include a drop down list of state charts from which you select the state chart containing the state.

3. The corresponding state name appears in the State Name window.

**Jumping to the state corresponding to the state ID**

You can also use the state ID command to jump to the state corresponding to a specified state ID numeric value.

To jump to a state
1. Choose State Charts > state ID Lookup.
   The State Lookup dialog box appears.

2. In the State Id box, enter the numeric value and click on Jump.
   If there is more than one state chart in the diagram, the State Lookup dialog box expands to include a drop down list of state charts from which you select the state chart containing the state.

**Annotating state charts**

You use the comment and label blocks to annotate your state chart. Both these blocks are located under the State Charts menu.

**Adding comments**

The comment block lets you add a multi-line explanatory text to your state chart.

To add text
1. Right click the mouse over the comment block.
2. The pointer changes into a vertical I-beam, indicating that you’re in text-entry mode.

As you insert text, VisSim automatically scrolls the text if it runs out of room in the viewable region of the block.

**To correct or remove text**
- Use the **DEL** and **BACKSPACE** keys.

**To exit text-entry mode**
- Click the right mouse button on the comment block a second time.

**To resize a comment block:**
- Drag on its edges.

**Adding labels**

The label block lets you insert floating labels in a state chart. You can choose the text attributes for the label, as well as a colored background.

To display a label, View > Block Labels must be activated.

**To create a label**
1. Right click the mouse over the **label** block.
2. In the Label Properties dialog box, enter the label. You can specify background color and font color for the label.

---

**Working with states and pseudo-states**

A state represents a distinct context for the behavior of an object. A state can be active or inactive during execution. A state becomes active when it is entered as a result of a transition, and becomes inactive if it is exited as a result of a transition. A state can be exited and entered as a result of the same transition (referred to as self transition).

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>composite state</td>
<td>A composite state is a state with one or more regions.</td>
</tr>
<tr>
<td>final state</td>
<td>Completion of a region.</td>
</tr>
<tr>
<td>state</td>
<td>A state that does not have regions or hierarchy. It is sometimes referred to as a simple state.</td>
</tr>
<tr>
<td>submachine state</td>
<td>A submachine state contains nested states. The submachine state lets you divide a complex start chart into subsystems.</td>
</tr>
</tbody>
</table>

A pseudo-state refers to any element in the state chart that will not be occupied for any duration. Pseudo-states are typically used to connect transitions into more complex state transition paths.

<table>
<thead>
<tr>
<th>Pseudo-state</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>choice</td>
<td>Splits transition into multiple paths</td>
</tr>
<tr>
<td>deep history</td>
<td>Represents the most recent active configuration of the composite state</td>
</tr>
<tr>
<td>entry point/exit point</td>
<td>Allow connections from a state to a submachine state</td>
</tr>
<tr>
<td>fork</td>
<td>Allows entry into multiple regions of a composite state</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>history</td>
<td>Represents the most recent active substate of its</td>
</tr>
<tr>
<td></td>
<td>containing state</td>
</tr>
<tr>
<td>initial state indicator</td>
<td>Represents the source for a single transition</td>
</tr>
<tr>
<td>join</td>
<td>Exits multiple regions of a composite state</td>
</tr>
<tr>
<td>junction</td>
<td>Chains together multiple transitions</td>
</tr>
<tr>
<td>terminate</td>
<td>Causes the state chart to terminate all execution</td>
</tr>
</tbody>
</table>

**Inserting a state or pseudo-state**

**To insert a state or pseudo-state**
1. Click on **State Charts**.
2. Choose the appropriate state or pseudo-state.
3. Click the mouse over the work area.

**Moving and resizing states**

**To move a state**
1. Position the cursor over the **Header** bar of the state. The Header bar is the upper portion of the state displaying the state name.
   The cursor changes to ✈.
2. Hold down the mouse button and drag the mouse.

**To move a group of states and pseudo-states**
1. Hold down the mouse button and drag the mouse so that it lassos the elements to be grouped.
2. As you drag the mouse, a marquee appears around the grouped elements.
3. Release the mouse button.
4. Position the cursor over a pseudo-state or over the **Header** bar of a state in the group.
   The cursor changes to ✈.
5. Hold down the mouse button and drag the mouse.

**To change the size of a state**
1. Position the cursor over one of the corners of the state.
   The cursor changes to ✈.
2. Hold down the mouse button and drag the mouse.

**Navigating through submachine state hierarchy**

**To move down**
1. Position the cursor over the **Header** bar of the submachine state. The Header portion is the upper portion of the state displaying the state name.
   The cursor changes to ✈.
2. Double-click the mouse.

To move up
1. Position in the work area over empty screen space.
2. Right-click the mouse.

Specifying state names
You can apply names to simple, composite, and submachine states. The state name appears in the Header portion of the state. The default name is StateN.

To specify a state name
1. Position the cursor over the Header bar of the state.
   The cursor changes to ▼.
2. Right-click the mouse to access the Properties dialog box.
3. Click on the Options tab.
4. In the Name box, enter a new name and press the OK button.

Coloring states and pseudo-states
You can edit the color of individual states and pseudo-states. The colors you specify override the colors specified in State Charts > Preferences.

To change state colors
1. Position the cursor over the Header bar of the state.
   The cursor changes to ▼.
2. Right-click the mouse to access the Properties dialog box.
3. Click on the Options tab.
4. Under Color, make the color selections for the Header, Border, and Background, and press the OK button.

To change pseudo-state colors
1. Position the cursor over the pseudo-state.
   The cursor changes to ▼.
2. Right-click the mouse to access the Properties dialog box.
3. Under Color, make the color selections for the Foreground and Background, and press the OK button.

To set default colors and fonts for state chart elements
You can set default colors and fonts for elements in your state charts using the State Charts > Preferences command. The colors and fonts you specify will not be applied to elements already in the state chart.
1. Choose State Charts > Preferences.
   The following dialog box appears.
The Category window lists the types of objects for which you can set color and
font characteristics.

2. In the **Category** window, select the **category**.
   The Appearance window displays your selection.

3. Do the following:

<table>
<thead>
<tr>
<th>To change</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Font</strong></td>
<td>Under <strong>Font</strong>, click the ... button.</td>
</tr>
<tr>
<td><strong>Color</strong></td>
<td>Under <strong>Colors</strong>, select the area to be colored (if there are choices), then click the <strong>Select</strong> button.</td>
</tr>
</tbody>
</table>

As you make changes color and font changes, those changes appear in the
Appearance window.

4. Click the **OK** button, or press **ENTER**.

**Associating behaviors with states**

A state can have a number of behaviors associated with it. A behavior, which is
defined using C statements, is executed upon entry, exit, on trigger, or presence in a
given state.

**To associate a behavior with a state**

1. Position the cursor over the **Header** portion of the state.
   The cursor changes to .

2. Right-click the mouse to access the Properties dialog box.

3. Click on the **Activity Manager** tab.

4. Under **Actions**, select the action for the behavior. That is, specify when the code
   that defines the behavior is executed. Your choices are:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>Code is executed on state entry</td>
</tr>
<tr>
<td>Exit</td>
<td>Code is executed on state exit</td>
</tr>
<tr>
<td>Do</td>
<td>Code is executed for each execution cycle that the state remains active</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>On Trigger</td>
<td>Code is executed when the state is active and when a specified trigger is TRUE; you must also select the trigger from Available Triggers</td>
</tr>
</tbody>
</table>

5. Press the **Add Action** button.
6. The selected action is displayed in the **Selected Actions** and **Edit Actions** windows.
7. In the **Edit Actions** window, using **C statements**, enter the state behavior for the corresponding action.
8. Click on the **OK** button, or press **ENTER**.

**Editing a state behavior**

State behavior is defined using C statements. If you are unfamiliar with the C language, refer to *C: A Software Engineering Approach*.

**To edit a behavior**

1. Position the cursor over the **Header** portion of the state.
   The cursor changes to 🖼.
2. Right-click the mouse to access the Properties dialog box.
3. Click on the **Activity Manager** tab.
4. Under **Selected Actions**, select the action corresponding to the behavior you want to edit.
5. The code to be edited is displayed in the **Edit Action** window.
6. Edit the code.
7. Click on the **OK** button, or press **ENTER**.

**Changing the order of internal transitions**

When a state has two or more internal transitions, you may need to change their execution order.

**To change the execution order**

1. Position the cursor over the **Header** portion of the state.
   The cursor changes to 🖼.
2. Right-click the mouse to access the Properties dialog box.
3. Click on the **Activity Manager** tab.
4. Under **Selected Actions**, select the trigger whose execution order is to be changed.
5. Use the 🖼 arrows to change the order of the triggers.
**Setting breakpoints for states**

You can attach a breakpoint to an action that causes VisSim to stop executing when the selected action occurs. If a breakpoint is active on a state, a red ball appears to the left of the state name in the state title bar. For more information, see [Using breakpoints](#).

**Logging messages for states**

You can log user-specified messages when a state behavior is executed and use the eventDisplay block to browse the logged messages. When logging is active on a state, a red exclamation point appears next to the state name in the Header. For more information, see [Logging messages](#).

**Working with regions in composite states**

A composite state has the following areas:

![Composite State Diagram](image1)

When you create regions, those regions are in the decomposition compartment of the composite state. VisSim provides each region with a number. The number indicates the evaluation order.

![Composite State Diagram with Regions](image2)

The above composite state has two regions. The top region is labeled 1, indicating it will be evaluated first. For more information on how VisSim evaluates composite states, see “Evaluating an orthogonal composite state.”

You can add regions horizontally or vertically; you can also resize regions.

**To add a region**

1. Click right the mouse over in **Decomposition compartment** of the composite state.
2. In the pop-up menu, choose **Insert Horizontal Region** or **Insert Vertical Region**.

**Note:** Horizontal regions work best when you are using join or fork pseudo-states.

**To resize a region**
1. Point to the dashed line representing the region border.
   The cursor changes to a double-headed arrow.
2. Hold down mouse key and drag the border.

To remove a region
1. Point to the dashed line representing the region border.
   The cursor changes to a double-headed arrow.
2. Hold down mouse key and drag to the closest border.

When you release the mouse, the remaining regions are renumbered appropriately.

To change the evaluation order
1. Right click the mouse over any region.
2. In the pop-up menu, choose **Reorder Region**.
   The regions are outlined in blue.
3. Point to the region to be designated 1 and click the mouse.
4. Repeat step 3 for all remaining regions that are to be re-ordered.
5. Right click the mouse to exit re-order mode.

---

**Working with transitions**

Transitions are paths from one block to another on which the logic of the system flows. Transition from one state to another occurs when certain conditions are met.

**Drawing a transition**

1. Position the cursor over the edge of a source state.
2. The cursor changes to a \[ \text{transition} \].
3. Depress the mouse and drag into the target state body.
4. Release the mouse button.
5. The transition line is drawn from the source state to the target state.

**Bending and moving transitions**

To improve the appearance of transitions, you can bend them or move where they attach to the target state.

<table>
<thead>
<tr>
<th>To</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bend</td>
<td>Click the mouse over the transition. The transition is highlighted in purple. Drag the mouse to create an arc.</td>
</tr>
<tr>
<td>Move</td>
<td>Click the mouse over the transition. The transition is highlighted in purple. Position the cursor over the transition arrowhead. The cursor changes to [ \text{move} ]. Drag the mouse to move the transition.</td>
</tr>
</tbody>
</table>
Deleting transitions

1. Click the mouse over the transition to be deleted.
   The transition is displayed in purple.

2. Do one of the following:
   - Press the DELETE key.
   - Right click the mouse and select Delete from the pop-up menu.
   - Position the mouse over the transition arrowhead, drag the mouse away from the target state, and release the mouse button. The distance you must drag the arrowhead is dictated by the Auto Radius Connect setting in the Edit > Preferences menu.

Defining transition specifications

A transition specification can contain triggers, a guard, and a behavior expression in the following general format:

```
(trigger(s) [guard]) / behavior
```

- **trigger**: one or more optional events that cause the transition to occur, provided that the guard, if specified, is TRUE. Separate multiple triggers with commas. For multiple triggers, as long as at least one trigger is active the transition, and the guard, if specified, is TRUE, the transition occurs. For more information on triggers, see Working with triggers.

- **[guard]**: an optional Boolean condition that, when TRUE, allows the transition to occur for the specified triggers. Enclose the guard in square brackets[].

- **/ behavior**: one or more optional C statements that are executed after the source state is exited and before the target state is entered. The forward slash (/) is mandatory, even if there are no triggers or guards.

The full BNF notation for the transition specification is:

```
[<trigger> [',' <trigger>]* ['[' <guard>']] ['/' <behavior>]]
```

In the following example, a pump switches from a draining mode to idle mode based on the transition specification.
The trigger `not-running` causes the transition of the pump from ON to IDLE as long as the condition `[level <= 0]` is TRUE.

**To create a transition specification**

1. Do one of the following:
   - Double-click the mouse over the transition.
   - Click the mouse over the transition to highlight it in purple; then right-click the mouse and select **Properties** from the pop-up menu.
2. Right-click the mouse to access the Properties dialog box.
3. Under **Edit Action**, enter the **transition specification**, as described above.

You specify a C expression guard in the Edit Action area of the Transition Properties dialog box. You must surround the expression with square brackets. Click the OK button to complete the transition behavior.

After you specify a behavior, it will appear over the transition in the state chart. You can move the rule by clicking on it and dragging to a new location. If you move the state or transition, the rule will maintain its relative location on the transition.

**Determining transition evaluation order**

If you have more than one transition coming from a given state, the transitions will be numbered according to the evaluation order. Transitions with lower numbers have higher priorities.
To change the evaluation order

1. Right click the mouse on one of the transitions.
2. In the pop-up menu, select Reorder Transitions.
3. Click on the transitions in the order you want them evaluated.

Setting breakpoints for transitions

You can attach a breakpoint to transition that causes VisSim to stop executing when the selected transition occurs. If a breakpoint is active on a transition, a red ball appears in the arrowhead of the transition. For more information, see Using breakpoints.

Logging messages for transitions

You can log user-specified messages when a transition is executed and use the eventDisplay block to browse the logged messages. When logging is active on a transition, a red ball appears on the transition arrowhead. For more information, see Logging messages.

Working with triggers

Two ways to use triggers:

- In the VisSim continuous modeling space with the trigger block
- As a local variable to a given state chart

Using triggers to control a state chart from VisSim

If you want to control your state chart from VisSim, use the trigger block in the continuous space and connect it to a Boolean expression. When input to the trigger block is 1, the trigger activates; when input is 0, it de-activates.

In the diagram below, a pulseTrain block is used to control the trigger. When the simulation starts, the initial state transitions to State1, and the variable $a$ equals 2. When the trigger goes high, State1 transitions to the final state. On State1 exit, $a$ equals 3. The variable $b$ shows that the transition to final state occurred.
Using the SendT() function to activate triggers

To activate a local state chart variable trigger, use the pre-defined SendT() function in places where you can insert behavioral (C) code. Typically, behavioral code can be inserted in state and transition behaviors using the Do action.

The format of the SendT() function is:

\[ \text{SendT}( \text{trigger-name} ) \]

Trigger name is a user-specified name. The following rules apply to trigger-name:

- It must be defined in the Data Browser tab for the State Chart Block Properties dialog box
- It has Local scope
- Its data type is Trigger

The following example uses a local variable named trigger and the SendT() function to control trigger.
When the simulation starts, the initial state transitions to State 1, and the variable $a = 2$. The trigger is executed if the state is active and remains active for the current simulation step. When trigger goes high, State 1 transitions to the final state. On State1 exit, $a = 3$. The variable $b$ shows that the transition to final state occurred.

Referencing triggers

You can reference triggers on transitions or in states.

To reference the trigger on the transition

1. Do one of the following:
   - Double-click the mouse over the transition.
   - Click the mouse over the transition to highlight it in purple; then right-click the mouse and select Properties from the pop-up menu.
2. Right-click the mouse to access the Properties dialog box.
3. Under **Available Triggers**, select the trigger and click on **Add**.

4. The trigger is added to the **Selected Trigger** window and the **Edit Action** window.

5. Click on the **OK** button, or press **ENTER**.

**To reference the trigger in the state**

1. Position the cursor over the **Header** portion of the **state**.

   The cursor changes to $\bullet$.

2. Right-click the mouse to access the Properties dialog box.

3. Click on the **Activity Manager** tab.

4. Under **Actions**, select **On Trigger**.

5. Under **Available Triggers**, select the trigger and click on the **Add Action** button.

6. In the **Selected Actions** window, the **On Trigger** action is displayed.

7. In the **Edit Action**, the trigger is displayed.

8. Edit the code.

9. Click on the **OK** button, or press **ENTER**.

**Using deferred triggers**

Only states can defer triggers.

A state may specify a set of triggers that may be deferred in that state. A trigger that does not fire any transitions in the current state will remain active if it is declared in the deferred trigger set of that state. That is, it remains active on consecutive simulation steps. This situation persists until a state is reached where either the trigger is no longer deferred or the trigger fires a transition.

If the deferred trigger fires while in the state in which it is deferred, its firing will be remembered.

**Using deferred triggers in composite states**

Composite states introduce potential trigger deferral conflicts. Each of the substates may defer or consume a trigger, potentially conflicting with the composite state (for example, a substate defers a trigger while the composite state consumes it, or vice versa).

Substates of parallel regions may also introduce deferral conflicts. The conflict resolution follows the triggering priorities, where nested states override enclosing states. In case of a conflict between states in different regions, a consumer state overrides a deferring state.

**To specify a deferred trigger**

1. Position the cursor over the **Header** portion of the **state**.

   The cursor changes to $\bullet$.

2. Right-click the mouse to access the Properties dialog box.
3. Click on the Activity Manager tab.
5. Under Available Triggers, select the trigger and click on the > button.
   
   The trigger is moved to the Deferred Triggers window.
6. Click on the OK button, or press ENTER.
Evaluating State Charts

This section contains…

State chart evaluation

A state chart is evaluated at each time step of the VisSim simulation clock, or at each sample interval of the embedded system. At most, a single transition is taken within a simple state chart, or single transitions within each region of a composite state.

Event processing: run-to-completion step

Event occurrences are detected, dispatched, and then processed by the state chart one at a time. The semantics of event processing is based on run-to-completion processing. Run-to-completion processing means that an event is only processed when the processing of the previous event is fully completed. The processing of a single event occurrence by a state chart is known as a run-to-completion step.

Before commencing a run-to-completion step, a state chart is in a stable state configuration with all entry, exit, and internal activities completed. The same conditions apply after the run-to-completion step is completed. Thus, an event will never be processed while the state chart is in an intermediate and inconsistent situation.

The run-to-completion step is the passage between two state configurations of the state chart. When an event is detected and dispatched, it may result in one or more transitions being enabled for firing. If no transition is enabled and the event is not in the deferred event list of the current state, the event is discarded and the run-to-completion step is completed.

In the presence of parallel regions, it is possible to fire multiple transitions as a result of the same event. If one or more transitions are enabled, the state chart selects a subset and fires them. The enabled transition that actually fires is determined by the transition selection algorithm. Each parallel region in the active state chart diagram can fire at most one transition during a run step. When all regions have finished executing the transition, the current run-to-completion step is complete.
Evaluating states

Active and inactive states

A simple, composite, and submachine state can be active or inactive during execution. A state becomes active when it is entered as a result of a transition, and becomes inactive if it is exited as a result of a transition. A state can be exited and entered as a result of the same transition (referred to as self transition).

Active states in a hierarchical state chart

In a hierarchical state chart, more than one state can be active at the same time. If the state chart is in a simple state that is contained in a composite state, then all the composite states that either directly or transitively contain the simple state are also active. Furthermore, since the state chart and some of the composite states in the hierarchy may be orthogonal (that is, contain multiple regions), the current active state is actually represented by a set of trees of states starting with the top-most states of the root regions down to the innermost active substate. These state trees are referred to as a state configuration. Except during transition execution, the following invariants always apply to state configurations:

- If a composite state is active and not orthogonal, at most one of its substates is active.
- If the composite state is active and orthogonal, all of its regions are active, with at most one substate in each region.

Visually displaying the active state

1. Choose State Charts > Preferences.
2. Activate Highlight Active States During Execution.
3. Click on the OK button, or press ENTER.

Entering and exiting states

Whenever a state is entered, it executes its entry action before any other action is executed. Whenever a state is exited, it executes its exit action as the final step prior to leaving the state.

Action while in a state (do action)

While the state chart remains in a state, the do action is performed.

Entering and exiting composite states

In ideal world, regions of a composite state should be evaluated in parallel. In VisSim/State Charts execution model, regions are evaluated sequentially. Each region within an orthogonal composite state has a number. A region with smaller number will be entered and executed before a region with higher number. The regions are exited in opposite order.

Entering a single-region composite state

There are five ways to enter a single region composite state: default entry, explicit entry, shallow history entry, deep history entry, and entry point entry.
Default entry

Graphically, default entry is indicated by an incoming transition that terminates on the outside edge of the composite state. In this case, the default entry rule is applied. If there is a guard on the trigger of the transition, it must be enabled. A disabled initial transition is an error. The entry action of the composite state is executed before the action associated with the initial transition.

Explicit entry

If the transition goes to a substate of the composite state, then that substate becomes active and its entry code is executed after the execution of the entry code of the composite state. This rule applies recursively if the transition terminates on a transitive nested substate.

Shallow history entry

If the transition terminates on a shallow history pseudo-state, the active substate becomes the most recently active substate prior to this entry, unless the most recently active substate is the final state or if this is the first entry into this state. In the latter two cases, the default history state is entered. This is the substate that is target of the transition originating from the history pseudo-state. (If no such transition is specified, the initial transition is executed.) If the active substate determined by history is a composite state, then it proceeds with its default entry.

Deep history entry

The shallow history rule applies with this exception the rule is applied recursively to all levels in the active state configuration below this one.

Entry point entry

If a transition enters a submachine state through an entry point pseudo-state, then the entry behavior is executed before the action associated with the internal transition emanating from the entry point.

Entering an orthogonal (multi-region) composite state

Whenever an orthogonal composite state is entered, each one of its regions is also entered, either by default or explicitly. If a transition terminates on the outer edge of the composite state, then all the regions are entered using default entry. If the transition explicitly enters one or more regions (for example, in case of a fork), these regions are entered explicitly and the others by default.

Exiting a single-region composite state

When exiting from a single-region composite state, the active substate is exited recursively. This means that the exit activities are executed in sequence starting with the innermost active state in the current state configuration. If, in a submachine state, the exit occurs through an exit point pseudostate, the exit behavior of the state is executed after the behavior associated with the transition incoming to the exit point.

Exiting an orthogonal (multi-region) composite state

When exiting from an orthogonal composite state, each of its regions is exited. After that, the exit activities of the containing state are executed.
Evaluating transitions

Transition firing priority

It is possible for more than one transition to be enabled within a state chart. For example, there can be two transitions originating from the same state, triggered by the same event, but with different guards. In this case, only one of them will fire in a given step: the lower order transition (marked by a number near its exit from the state) will fire.

Transitions that occur in parallel regions may be fired simultaneously.

A transition originating from a substate has higher priority than a transition originating from any of its containing states. The priority of a transition is defined based on its source state. The priority of joined transitions is based on the priority of the transition with the most transitively nested source state. In general, if t1 is a transition whose source state is s1, and t2 has source s2, then:

- If s1 is a direct or transitively nested substate of s2, then t1 has higher priority than t2.
- If s1 and s2 are not in the same state configuration, then there is no priority difference between t1 and t2.

Transition selection algorithm

For every state, there is a set of outgoing transitions, of which no more than one will fire. The order of evaluation of transitions is determined by the following:

- Outgoing transitions belonging to the given region and with triggers.
- The transitions must retain their order in vertex's outgoing transitions list.
- External compound transitions with triggers outgoing from enclosed states.
- Outgoing transitions belonging to the given region and without triggers.
- The transitions must retain their order in vertex's outgoing transitions list.
- External compound transitions without triggers outgoing from enclosed states (for non-simple states).

High-level transitions

Transitions originating from composite states are called high-level or group transitions. If triggered, they result in exiting of all the substates of the composite state executing their exit activities starting with the innermost states in the active state configuration. A high-level transition with a target outside the composite state causes execution of the exit action of the composite state, while a high-level transition with a target inside the composite state will not cause execution of the exit action of the composite state.

Compound transitions

A compound transition represents a “semantically complete” path made up of one or more transitions, originating from a set of states (as opposed to pseudostate) and targeting a set of states. The transition execution semantics described below refer to compound transitions. In general, a compound transition is an acyclical unbroken
chain of transitions joined via join, junction, choice, or fork pseudostates that define a path from a set of source states (possibly a singleton) to a set of destination states (possibly a singleton).

For self-transitions, the same state acts as both the source and the destination set. A simple transition connecting two states is therefore a special common case of a compound transition. The tail of a compound transition may have multiple transitions originating from a set of mutually orthogonal regions that are joined by a join point. The head of a compound transition may have multiple transitions originating from a fork pseudostate targeted to a set of mutually orthogonal regions.

In a compound transition, multiple outgoing transitions may emanate from a common junction point. In that case, only one of the outgoing transitions whose guard is true is taken. If multiple transitions have guards that are true, a transition from this set is chosen. Note that in this case, the guards are evaluated before the compound transition is taken.

In a compound transition where multiple outgoing transitions emanate from a common choice point, the outgoing transition whose guard is true at the time the choice point is reached, will be taken. If multiple transitions have guards that are true, one transition from this set is chosen. If no guards are true after the choice point has been reached, the model is ill formed.

**Completion transitions**

A completion transition is a transition originating from a state or an exit point, and does not have an explicit trigger, although it may have a guard defined. For a simple state, if it is active at the beginning of simulation step, then its completion transition may be executed.

If the state is a composite state or a submachine state, a completion transition may be executed if either the submachine or the composite state is active at the beginning of simulation step, and its contained regions have reached final states. If multiple completion transitions are defined for a state, then they should have mutually exclusive guard conditions.

**Enabled transitions**

A transition is enabled if:

- Its source state is active.
- One of the transition triggers is true.
- If there exists at least one full path from the source state to either a target state or to a choice point in which all guard conditions are true (transitions without guards are treated as if their guards are always true).

**Guards**

In a simple transition with a guard, the guard expression must evaluate to true before the transition is triggered. In compound transitions involving multiple guards, all guards are evaluated before a transition is triggered, unless there are choice points along one or more of the paths. Guards downstream of a choice point are only evaluated if the choice point is reached. Guards should not include expressions causing side effects.
Transition execution sequence

Every transition, except for internal and local transitions, causes exiting of a source state, and entering of the target state.

Transition kinds

- Internal implies that the transition occurs without exiting or entering the source state. Thus, it does not cause a state change. This means that the entry or exit condition of the source state will not be invoked. An internal transition can be taken even if the state chart is in one or more regions nested within this state.

- Local implies that the transition, if triggered, will not exit the composite (source) state, but it will apply to any state within the composite state, and these will be exited and entered.

- External implies that the transition, if triggered, will exit the composite (source) state.
State Chart Elements

This section describes all the elements you use to build a state chart.

States

A state describes a period of time during which some condition holds. The state may represent a static situation such as waiting for some event to occur or a dynamic condition such as the process of performing some behavior (for example, the filling of a tank).

The state is entered when the condition starts and is exited as when the condition is complete. For complete information, see Evaluating states.

simple state

A simple state is a state with no regions or substates. For information on editing states, see Working with states and pseudo-states.

Icon

![State Icon](image)
**Border:** Specifies the color for the state borders.

**Header:** Specifies the color of the header region. To change the font color, use the View > Fonts command.

**Background:** Specifies the color of the decomposition region of the state.

**Indicates information or notes about the state. The comments only appear in the dialog box.**

**Specifies a name for the state. The name can be alphanumeric characters. It must be unique with respect to other state names in the state chart.**

**Specifies one or more actions and associated State Activity Code**
behaviors. Enter and edit state activity code in the Activity Manager window.

Entry: Code is executed on state entry.
Exit: Code is executed on state exit.
Do: Code is executed for each execution cycle that the state remains active.
Inner Trigger: Code is executed when the state is active and when the specified trigger is TRUE.

Adds or deletes the action to the Selected Actions window.

Specifies the state behavior for the corresponding action. State behavior is specified as C statements. For more information, see Associating behaviors with

Add/Delete Action

Edit Behavior
States:

Lists the available triggers.

Lists the deferred triggers.

Sets a breakpoint on the state that causes the simulation to stop after the corresponding VisSim time step is complete. For more information, see Using breakpoints.

Logs the specified message when a state behavior is executed. For more information, see Logging messages.

Available Triggers

Deferred Triggers

Attach Breakpoint

Log Message

**composite state**

A composite state is a state with at least one region. The icon, in the upper left corner of the composite state, distinguishes it from a simple state.

**Icon**

Regions are created in the Decomposition Compartment. If a composite state contains at least two regions, it is referred to as an orthogonal composite state. Each region, in an orthogonal composite state, has a set of states and transitions that can execute in parallel, but cannot transition between regions. Each region can support an active state and take transitions in parallel with other regions of an orthogonal composite state.

Any state within a region of a composite state is called a substate of that composite state. If the substate is not contained by any other state, it is called a direct substate; otherwise, it is referred to as an indirect substate.

Each region of a composite state may have an initial state indicator and a final state. A transition to the enclosing state represents a transition to the initial state indicator in each region. A transition to a final state represents the completion of the enclosing region. Completion in all orthogonal regions represents completion of the enclosing state. Completion of the topmost regions of a composite state corresponds to its termination.
If a transition terminates on an enclosing state and the enclosed regions do not have an initial pseudo-state, it is considered an ill-formed model. That is, the initial pseudo-state is mandatory.

For information on editing composite states, see Working with states and pseudo-states.

**Constraints**

- A region can have at most one initial state indicator
- A region can have at most one deep history pseudo-state
- A region can have at most one shallow history pseudo-state

---

**Border**: Specifies the color for the state borders.

**Header**: Specifies the color of the header region. To change the font color, use the View > Fonts command.

**Background**: Specifies the color of the decomposition region of the state.

**Comment**: Indicates information or notes about the state. The comments only appear in
the dialog box.

Specifies a name for the state. The name can be alphanumeric characters. It must be unique with respect to other state names in the state chart.

Specifies one or more actions and associated behaviors. Enter and edit state activity code in the Activity Manager window.

Entry: Code is executed on state entry.
Exit: Code is executed on state exit.
Do: Code is executed for each execution cycle that the state remains active.
Inner Trigger: Code is executed when the state is active and when the specified trigger is TRUE.
<table>
<thead>
<tr>
<th>Feature Description</th>
<th>Button Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adds or deletes the action to the Selected Actions window.</td>
<td>Add/Delete Action</td>
</tr>
<tr>
<td>Specifies the state behavior for the corresponding action. State behavior is specified as C statements. For more information, see Associating behaviors with states.</td>
<td>Edit Behavior</td>
</tr>
<tr>
<td>Lists the available triggers.</td>
<td>Available Triggers</td>
</tr>
<tr>
<td>Lists the deferred triggers.</td>
<td>Deferred Triggers</td>
</tr>
<tr>
<td>Sets a breakpoint that causes the simulation to stop after the corresponding VisSim time step is complete. For more information, see Using breakpoints.</td>
<td>Attach Breakpoint</td>
</tr>
<tr>
<td>Logs the specified message when a state behavior is executed. For more information, see Logging messages.</td>
<td>Log Message</td>
</tr>
</tbody>
</table>

**submachine state**

A submachine state encapsulates one or more states. The \( \bigcirc \) icon, in the upper right corner of the state, distinguishes it from a simple state.

To dive into the next lower level of a submachine state, double-click the Header bar.

A state chart that contains a submachine state is called the **containing state chart**. You can have multiple levels of containing state charts. Transitions in the containing state chart can have entry and exit points of the inserted state chart as targets and sources.

A submachine state is semantically equivalent to a composite state with a single region. The entry, exit, and behavior actions and internal transitions are defined as part of the state.

You can enter a submachine state in the same way as you enter a non-orthogonal composite state. You can also enter and leave a submachine state, in contrast to an ordinary composite state, via entry and exit points.

For information on editing submachine states, see Working with states and pseudo-states.
Connection Point References

When you exit a submachine state, the entry/exit point references appear on the edge of the submachine state.

Constraints

- Only submachine states can have connection point references.
- The connection point references used as destinations/sources of transitions associated with a submachine state must be defined as entry/exit points in the submachine state chart.

Border: Specifies the color for the state borders.
Header: Specifies the color of the header region. To change the font color, use the View > Fonts command.
Background: Specifies the color of the decomposition region of the state.

Color
Indicates information or notes about the state. The comments only appear in the dialog box. Comment

Reserved. Embed Existing State Machine

Specifies a name for the state. The name can be alphanumeric characters. It must be unique with respect to other state names in the state chart. Name

Protects the submachine state. To lock the submachine state, activate Locked and enter a password in the Password box. To make the submachine read-only, activate Read-Only and enter a password in the Password box. Protection

Specifies one or more actions and associated behaviors. Enter and edit state activity code in the Activity Manager window. State Activity Code
Indicates variable names. See Setting up state chart variables.

Indicates the data type of the variable. See Defining variable data types.

Indicates how the variable is scoped. See Defining variable scope.

Indicates the input or output pin number on the state chart block. For State ID, it indicates the active state from the given region.

Indicates the initial value for the variable.

Indicates the current value for the variable.
of the variable.

*Indicates user-specified comments.*

**Comment**

**Example**

The submachine state State2 has one entry point and one exit point. The next lower level of State2 has two exit points,

**final state**

The final state is used in a region to indicate completion of the region. If all other regions in the composite state are also completed, then the first active exit transition from the composite state is taken. For more information, see Completion transitions.

When the final state is entered, its containing region is completed. The containing state for this region is considered completed when all contained regions are completed. If the region is contained in a state machine and all other regions in the state machine also are completed, the entire state machine terminates, implying the termination of the context object of the state machine.

**Icon**

**Constraints**
• A final state cannot have any outgoing transitions.
• A final state has no actions.

**Final State Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td><img src="image" alt="Color Selection" /></td>
</tr>
<tr>
<td>Foreground</td>
<td><img src="image" alt="Foreground" /></td>
</tr>
<tr>
<td>Background</td>
<td><img src="image" alt="Background" /></td>
</tr>
<tr>
<td>OK</td>
<td>Cancel</td>
</tr>
</tbody>
</table>

*Foreground:* Specifies the color of the center and outer ring.

*Background:* Specifies the color of the inner ring.

**Name**

Specifies a name for the state.

---

**Pseudo-states**

A pseudo-state refers to any node in the state chart diagram that will not be occupied for any duration. Pseudo-states are typically used to connect transitions into more complex state transition paths.

**choice**

The choice pseudo-state evaluates the guards of the outgoing transitions. This forms a *dynamic conditional branch*. It allows splitting of transitions into multiple outgoing paths such that the decision of which path to take may be a function of the results of prior actions.

The dynamic conditional branch can be expressed as follows:

```plaintext
if (A) {
    Exit State One;
    B;
```
if (C){
    D;
    Enter State Two:
}
else if (E){
    F;
    Enter State Three:
}
}

If more than one the guard evaluates to TRUE, the transition will be executed according to its evaluation order. If none of the guards evaluates to TRUE, then the state chart is in error, and a message is produced indicating the offending element. To avoid this, it is recommended that every choice pseudo-state have one outgoing transition with the predefined “else” guard.

**Icon**

[Diagram of a diamond shape]

**Constraints**

- A choice pseudo-state must have at least one incoming and one outgoing transition.

**Example**

*Foreground:* Specifies the color of the center.
*Background:* Specifies the color of the border.
The result of the choice pseudo-state controls whether State Two or State Three is executed based on the value of state chart variables second and third. In this example, the top and bottom buttons are ON, resulting in the execution of state Three.

**deep history**

The deep history pseudo-state represents the most recent active configuration of the composite state that directly contains the deep history. That is, the state configuration that was active when the composite state was last exited.

**Icon**

A composite state can have at most one deep history pseudo-state. At most one transition may originate from the deep history pseudo-state to the default history pseudo-state. This transition is taken when the composite state has never been active before and therefore the history is empty. Entry actions of states entered on the path to the configuration represented by a deep history are performed.

**Foreground:** Specifies the color of the outer ring.

**Background:** Specifies the color of the center.
entry points and exit points

Entry and exit points are pseudo-states used within a submachine state. They allow connections from a state in the same chart into the entry or exit point of a submachine.

Icon

An entry point is the point of entry to a submachine. It has a single transition to a state within the same region. An exit point is a point of exit for a submachine. Transitioning to an exit point implies the exit of the containing submachine state.

To add entry and exit points

1. Double click over submachine title bar to enter the submachine state.
2. Choose State Charts > entry point to add entry points.
3. Choose State Charts > exit point to add exit points.

To change the appearance of entry and exit points:

1. Right click the mouse over an entry or exit point.
2. The Pseudostate Properties dialog box appears.
3. Make the necessary changes to the name and color of the point.
4. Click on the OK button, or press ENTER.

Foreground: Specifies the color of the outer ring.
Background: Specifies the color of the center.
Specifies a name for the state.

**Name**

**fork**

The fork pseudo-state is used to enter multiple regions of a composite state.

The transitions exiting a fork cannot have guards or triggers. The fork can have one or more exiting arrows to states.

**Icon**

```
[icon image]
```

**Constraints**

- A fork pseudo-state must have at least two outgoing transitions and exactly one incoming transition.
- All transitions outgoing a fork pseudo-state must target states in different regions of an orthogonal state.

**foreground**: Specifies the color of the icon.

**Color**

```
[foreground color selection]
```

**history**

The history pseudo-state, also referred to as shallow history, represents the most recent active substate of its containing state. Unlike deep history, history does not represent the history of any substates of that substate.

A composite state can have at most one history pseudo-state. A transition into history is equivalent to a transition coming into the most recent active substate of a state. At most, one transition may originate from the history pseudo-state to the default history state. This transition is taken in case the composite state has never been active before and therefore the history is empty. Entry actions of states entered on the path to the state represented by a history are performed.

**Icon**

```
[icon image]
```
Constraints

- History pseudo-states can have at most one outgoing transition.

**History Pseudostate Prop...**

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
</tr>
</tbody>
</table>

**Foreground:** Specifies the color of the outer ring.

**Background:** Specifies the color of the center.

---

**Initial state indicator**

The initial state indicator pseudo-state is the source for a single transition to the default state of a composite state. There can be no or one initial state indicator in a region. The outgoing transition from the initial state indicator may have an associated behavior, but not a trigger or guard.

**Icon**

- **Foreground:** Specifies the color of the outer ring.

**Background:** Specifies the color of the center.

---

**Constraints**

- An initial state indicator can have at most one outgoing transition.
- The outgoing transition from an initial state indicator may have a behavior, but not a trigger or guard.
**join**

The join pseudo-state is used to exit multiple regions of a composite state. The transitions entering a join cannot have guards or triggers. The join pseudo-state may have one or more arrows from source states.

**Icon**

```
[ ]
```

**Constraints**

- A join pseudo-state must have at least two incoming transitions and exactly one outgoing transition.
- All transitions incoming a join pseudo-state must originate in different regions of an orthogonal state.

```
Join Pseudostate Properties

Name

Color

Foreground: Specifies the color of the icon.

Color

OK  Cancel  Help
```

**junction**

The junction pseudo-state is used to chain together multiple transitions. For example, a junction can be used to converge multiple incoming transitions into a single outgoing transition representing a shared transition path, called a merge. Conversely, junctions can be used to split an incoming transition into multiple outgoing transition segments with different guard conditions. This realizes a *static conditional branch*. Outgoing transitions whose guard conditions evaluate to FALSE are disabled. A predefined guard named “else” may be used for one outgoing transition. This transition is enabled if the guards labeling all the other transitions are FALSE.
The static conditional branch can be expressed as follows:

```c
if (A && C) {
    Exit State One;
    B;
    D;
    Enter State Two;
}
else if (A && E) {
    Exit State One;
    B;
    F;
    Enter State Three;
}
```

**Icon**

Static conditional branches are distinct from dynamic conditional branches that are realized by choice pseudo-states.

**Constraints**

- A junction pseudo-state must have at least one incoming and one outgoing transition.

**foreground:** Specifies the color of the icon.

**terminate**

A terminate pseudo-state causes the state chart to terminate all execution.

Entering a terminate pseudo-state implies that the execution of this state machine is terminated. The state machine does not exit any states nor does it perform any exit actions other than those associated with the transition leading to the terminate.

**Icon**
Foreground: Specifies the color of the icon.

Transition

A transition is a directed relationship between a source state and a target state. It may be part of a compound transition, which takes the state chart from one state configuration to another, representing the complete response of the state chart to an event.

For more information on transitions, see Working with Transitions.

Constraints

- A fork segment must not have guards or triggers.
- A join segment must not have guards or triggers.
- A fork segment must always target a state.
- A join segment must always originate from a state.
- Transitions outgoing pseudo-states may not have a trigger.
Lists all triggers within the scope of the transition.

Available Triggers

Lists the triggers associated with the transition.

Selected Triggers

Defines the transition specification. It can contain triggers, a BOOLEAN guard, and a behavioral expression. For more information, see Defining transition specifications.

Edit Behavior

Sets a breakpoint the transition that causes the simulation to stop after the corresponding VisSim time step is complete. For more information, see Using breakpoints.

Attach Breakpoint

Logs the specified message when a transition behavior is executed. For more information, see Logging messages.

Log Message
Repositions the transition specification label to its default location in the middle of the transition arc.

Blocks

A subset of blocks can be used with state charts.

**comment**

**Block Category:** State Charts, Blocks > Annotation

The comment block lets you add comments to a state chart. For more information, see Adding comments.

**convert ID**

**Block Category:** State Charts

The convert ID provides a method to identify the currently active state. It converts a numeric state ID to the corresponding state name.

The convert ID block is not only available for use inside the state chart; you can only insert convert ID blocks in the continuous portion of the diagram.

Indicates that the complete path to the currently active state is used. This is useful when your state charts contain multiple submachine states or composite states.

Indicates that only the name of the currently active state is used.

**Example**
In this example, output from the state chart block is the numeric ID of the currently active state. The convert ID block converts the numeric ID of the currently active state to its state name. When the simulation begins, convert ID converts the numeric ID to Initial. When trigger Turn ON evaluates to TRUE, the transition is taken from Initial state to Running state. The convert ID block converts the numeric ID to Running.

**function**

**Block Category:** State Charts

The function block lets you define one or more C functions that are called from any state or transition within the state chart.

You use the function block in the levels below the state chart container block.

*Background:* Specifies the color of the function window.
Specifies a name for the function. Function names must be unique within state chart.

Example

This state chart cycles through three states. The Function1 block increments an integer counter. When the counter reaches 11, the control is transitioned to the next state. The output variable for the exiting state is plotted.

label

**Block category:** State Charts, Blocks > Annotation

The label block lets you insert floating labels in a state chart. For more information, see Adding labels.

state chart

**Block Category:** State Charts

The state chart block is a container block in which you build your state chart. There is no restriction on the number of state chart blocks in a given diagram; however, you cannot nest state chart blocks.

To enter into the state chart block to build and edit your state chart, double-click the mouse over the state chart block.

To access the State Chart Block Properties dialog box to set the characteristics of the state chart, hold down the CTRL key while you right click the mouse.
**State Chart Block Properties**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appearance</td>
<td>Specifies the color of the state chart block. Activate <strong>Set Color</strong>, then click on <strong>Select</strong> to choose a color.</td>
</tr>
<tr>
<td>Contained State Count</td>
<td>The read-only parameter displays the number of states in the state chart.</td>
</tr>
<tr>
<td>State Chart Name</td>
<td>Specifies a name for the state chart. The name can be alphanumeric characters. It must be unique with respect to other state charts in the diagram. For long names, press <strong>CTRL+ENTER</strong> to wrap the name to a new line.</td>
</tr>
<tr>
<td>Log Executed C Instructions</td>
<td>Writes the C instructions in the specified .LOG file. Use the Browse button to select a .LOG file; use the Open button to open the .LOG file listed under File Name.</td>
</tr>
<tr>
<td>Protection</td>
<td>Restricts access to the state</td>
</tr>
</tbody>
</table>
To lock the state chart, activate **Locked** and enter a **password** in the **Password** box. To make the state chart read-only, activate **Read-Only** and enter a **password** in the **Password** box.

<table>
<thead>
<tr>
<th>Indicate</th>
<th>Name</th>
<th>Type</th>
<th>Scope</th>
<th>Pin</th>
<th>Default Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indicates variable names.</td>
<td>See <a href="#">Setting up state chart variables</a>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indicates the data type of the variable.</td>
<td>See <a href="#">Defining variable data types</a>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indicates how the variable is scoped.</td>
<td>See <a href="#">Defining variable scope</a>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indicates the input or output pin number on the state chart block.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indicates the initial value for the variable.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Indicates the current value of the variable.

Indicates user-specified comments.

Value

Comment

**state ID**

**Block Category:** State Charts

The state ID block outputs the following information about the selected state:

- Whether the state is active
- The numeric ID for the state

This block is useful for determining if the specified state is active. Thus, the state chart can control the model.

The state ID block is not only available for use inside the state chart; you can only insert state ID blocks in the continuous portion of the diagram.

Selects the state for which you want to display information. Select the state name in the window that shows the state chart composition. If the state chart has hierarchy or regions, use the + and – signs to navigate to the state to be selected.

State Name
Indicates the information to be displayed.

**Is State Active:** When activated, outputs 0 if the state is inactive, or 1 if the state is active.

**State ID:** When activated, outputs the numeric ID for the state.

Controls how the state name appears on the state ID block.

**Qualified Name:** Includes the complete path to the state.

**Short Name:** Displays on the state name.

Output Pin

Show Block Name As

 trigger

**Block Category:** State Charts

The trigger block lets you control your state chart from the continuous portion of the VisSim diagram. For more information, see Working with triggers.

Specifies a name for the trigger.

Trigger Name

Commands

The following commands are available for state charts.

**Check Integrity**

You use the Check Integrity command to see if your state chart contains errors. If there are one or more errors or warnings, they are listed in the ensuing Error Navigation panel.
State Id Lookup

You use the State Id Lookup command to display the state name that corresponds to a given state ID numeric value.

Breakpoints

You can set breakpoints on states and transitions. You use the Breakpoints command to list all the breakpoints that have been set in the state chart, delete breakpoints, and jump to breakpoints. For more information, see Using breakpoints.

Preferences

The Preferences command lets you control the appearance of your state chart.

Controls the color and font choices for the state and code windows, states and pseudo-states. Under Category, select the item to be customized; then make your font and color selections. A sample is shown in the Appearance window.

When activated, the active state is bolded.

Highlight Active State During Execution
If you are generating C code from your state chart, you can choose **Boolean array** or **Bit-field**. In C, the data type BOOL is defined as int. To minimize memory used to store Boolean values, you may want to select **Bit-field**. However, the code generated by the compiler to handle bit fields is usually slower and longer.

Generate Code For Triggers As
Debugging State Charts

Both common VisSim debugging capabilities and specific state chart debugging tools can be used to debug VisSim models containing state charts.

Visual debugging

To easily see where state chart execution is at any given moment, the following visual cues make it easy to see the active state and transition:

- Active states have a thicker border
- Active transitions are highlighted

If the active state changes frequently, you will want to run the simulation step by step to observe the currently active state or transition.

Visual debugging can be turned ON or OFF using the Highlight Active States During Execution parameter in the State Chart > Preferences dialog box.

Functional debugging

You can set up breakpoints and log messages when an action on either a state or transition is executed.

You may need to examine the state chart variable values when debugging a state chart. State chart variables are defined by the state chart block on the VisSim model level or the submachine state inside the state chart. The current value of a state chart variable is listed in the Data Browser tab for the State Chart Properties dialog box and the Submachine Properties dialog box.

Using breakpoints

State chart execution is atomic, that is, a simulation cannot be stopped in the middle of a simulation step. When a state chart breakpoint is hit, the simulation breaks after the corresponding VisSim time step is finished.

You can attach a breakpoint to an action that causes VisSim to stop executing when the selected action occurs.

You can set breakpoints on states and transitions. If a breakpoint is active on a state, a red ball appears to the left of the state name in the state title bar. If a breakpoint is active on a transition, a red ball appears on the arrowhead of the transition.
You can use the Breakpoints command to list all the breakpoints that have been set in the state chart, delete breakpoints, and jump to breakpoints. Breakpoints are prefaced with a red ball.

**To attach a breakpoint to a state action**
1. Position the cursor over the **Header** portion of the **state**.
   The cursor changes to ⬤.
2. Right-click the mouse to access the Properties dialog box.
3. Click on the **Activity Manager** tab.
4. Under **Selected Actions**, select the action that is to have a breakpoint.
5. Activate the **Attach Breakpoint** parameter.
6. Click on the **OK** button, or press **ENTER**.

**To attach a breakpoint to a transition**
1. Position the cursor over the transition.
2. Right-click the mouse and select Properties to access the Transition Properties dialog box.
3. Activate the **Attach Breakpoint** parameter.
4. Click on the **OK** button, or press **ENTER**.

**To list all breakpoints**
- Choose **State Charts > Breakpoints**.

**To jump to a breakpoint**
1. Choose **State Charts > Breakpoints**.
2. In the dialog box, select the breakpoint and click on the **Jump** button.

**To delete a breakpoint**
- Do one of the following:
  1. Choose **State Charts > Breakpoints**.
  2. In the dialog box, select the breakpoint to be deleted and click on the **Remove** button.
     - Or —
   1. Position the cursor over the **Header** portion of the **state**.
      The cursor changes to ⬨.
   2. Right-click the mouse to access the Properties dialog box.
   3. Click on the **Activity Manager** tab.
   4. Under **Selected Actions**, select the action that has a breakpoint.
   5. De-activate the **Attach Breakpoint** parameter.
   6. Click on the **OK** button, or press **ENTER**.

**To delete all breakpoints**
1. Choose **State Charts > Breakpoints**.
2. In the dialog box, click on the **Remove All** button.

**To resume execution**
You resume execution by pressing the toolbar button.

Logging messages

You can log user-specified messages when a state behavior is executed and use the eventDisplay block or Breakpoints command to browse the logged messages. When logging is active on a state, a red exclamation point appears next to the state name in the Header.

When logging is active on a transition, a red ball appears on the arrowhead for the transition.

Logging does not break chart execution.

Logging messages can be used not only for state chart debugging, but also as part of state chart interface for reporting events of interest.

To log state messages

When you simulate the diagram, the messages are stored in a temporary buffer. The contents of the buffer can be displayed in an eventDisplay block.

1. Position the cursor over the Header portion of the state.
   The cursor changes to •.
2. Right-click the mouse to access the Properties dialog box.
3. Click on the Activity Manager tab.
4. Activate the Log Message parameter and enter the message into the corresponding text box.
5. Click on the OK button, or press ENTER.

To attach a message to a transition

When you simulate the diagram, the messages are stored in a temporary buffer. The contents of the buffer can be displayed in an eventDisplay block.

1. Position the cursor over the transition.
2. Right-click the mouse and select Properties to access the Transition Properties dialog box.
3. Activate the Attach Log Messages parameter and enter a message in the corresponding text box.
4. Click on the OK button, or press ENTER.

To display logged messages with the eventDisplay block

The eventDisplay block displays the messages stored in a VisSim temporary buffer. You can insert an eventDisplay block at any time and if there are stored messages, the eventDisplay block will automatically list them.

Each time the diagram is executed, the temporary buffer overwrites the existing messages with new messages from the simulation.
1. In the continuous time portion of the diagram, insert an eventDisplay block, located under Blocks > Signal Consumer, into the work area.

2. If messages have already been logged, they will appear in eventDisplay window.

**To display logged messages with the Breakpoints command**

The Breakpoints automatically displays all logged messages stored in a VisSim temporary buffer. Logged messages are prefaced with a red exclamation point.

Each time the diagram is executed, the temporary buffer overwrites the existing messages with new messages from the simulation.

- Choose State Charts > Breakpoints.

**To jump to a logged message with the Breakpoints command**

1. Choose State Charts > Breakpoints.
2. In the dialog box, select the breakpoint and click on the Jump button.

**To delete a logged message**

- Do one of the following:
  1. Choose State Charts > Breakpoints.
  2. In the dialog box, select the logged message to be deleted and click on the Remove button.

    - Or -

    1. Position the cursor over the Header portion of the state.

        The cursor changes to $\Rightarrow$.

    2. Right-click the mouse to access the Properties dialog box.

    3. Click on the Activity Manager tab.

    4. Under Selected Actions, select the action that has a logged message.

    5. De-activate the Log Message parameter.

    6. Click on the OK button, or press ENTER.

**Example**

In this example, the state Init has message logging enabled. When the simulation runs, the eventDisplay block logs the message.
Chart-wide debugging

When you simulate a state chart, you can instruct VisSim to enter every instruction executed by the C interpreter into a corresponding Log file. If your state chart is not executing correctly, reviewing the Log file can help you erroneous or unwanted code execution.

By default, VisSim uses the name of the corresponding block diagram as the Log file name, and stores the Log file in the same directory as the corresponding block diagram.

To enter instructions in the Log file
1. Hold down the CTRL key and right click the mouse over the state chart block.
2. In the State Chart Block Properties dialog box, activate the Log Executed C Instructions parameter.

To change the name or location of the Log file
1. Hold down the CTRL key and right click the mouse over the state chart block.
2. In the State Chart Block Properties dialog box, click on the Browse button to enter a new name and location for the Log file.

To examine the Log file
1. Hold down the CTRL key and right click the mouse over the state chart block.
2. In the State Chart Block Properties dialog box, click on the Open button.

The Log file is opened in Notepad.
Error Navigation panel

If your state chart has errors or warnings, messages will be logged in the Error Navigation Panel.

The Error Navigation Panel appears if:

- You run a simulation that contains one or more state chart errors.
- You choose the State Charts > Check Integrity command and there are one or more errors or warnings in the state chart.

The Error Navigation Panel stores the error messages in a temporary buffer. When you close the Panel, the messages are discarded.

For the complete list of error and warning messages, see Error and warning messages.

Jumping to the error location

You use the arrow and hand keys on the left side of the panel to jump to error locations in the state chart. When you jump to the error location, the state that caused the error is highlighted. If the source of error is a transition, the source state for the offending transition is highlighted.

The Error Navigation Panel remains visible when you jump to the location of the error.

To move to an error location

2. Click on the hand.
3. To move to another error location, click on the appropriate arrow key

Error and warning messages

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A composite state can have at most one (deep) history vertex</td>
<td>A composite state, even orthogonal, may have no more than one history pseudo-state. A history pseudo-state stores the history for the whole composite state, not just for the region where the pseudo-state resides.</td>
</tr>
<tr>
<td>A (deep) history vertex can have at most one outgoing transition</td>
<td>A (shallow or deep) history pseudo-state must have just a single outgoing transition targeting a state in the same region. This transition is called a default history transition.</td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>A fork transition segment must always target a state</td>
<td>The transitions outgoing a fork pseudo-state must target states (but not pseudo-states) in different regions of the same composite (orthogonal) state.</td>
</tr>
<tr>
<td>A join transition segment must always originate from a state</td>
<td>The transitions incoming a join pseudo-state must originate from states, not pseudo-states, in different regions of the same composite (orthogonal) state.</td>
</tr>
<tr>
<td>All transitions incoming a join vertex must originate in different regions of an orthogonal state</td>
<td>The transitions incoming a join pseudo-state must originate from states (but not pseudo-states) in different regions of the same composite (orthogonal) state.</td>
</tr>
<tr>
<td>All transitions outgoing a fork vertex must target different regions of an orthogonal state</td>
<td>The transitions outgoing a fork pseudo-state must target states (but not pseudo-states) in different regions of the same composite (orthogonal) state.</td>
</tr>
<tr>
<td>An entry point must have a single transition to a vertex within the same region</td>
<td>An entry point pseudo-state must have just a single outgoing transition targeting a vertex in the same region.</td>
</tr>
<tr>
<td>An initial vertex can have at most one outgoing transition</td>
<td>An Initial vertex can have just a single outgoing transition targeting a state in the same region.</td>
</tr>
<tr>
<td>An initial vertex must have a single transition within the same region</td>
<td>An Initial vertex can have just a single outgoing transition targeting a state in the same region.</td>
</tr>
<tr>
<td>Argument for InState must be a quoted string constant</td>
<td>The InState function must have an argument that is a non-empty string constant.</td>
</tr>
<tr>
<td>A submachine state can have at most one (deep) history vertex</td>
<td>A submachine state may have no more than one history pseudo-state.</td>
</tr>
<tr>
<td>A transition outgoing a (deep) history vertex must target a vertex within the same region</td>
<td>A (shallow or deep) history pseudo-state must have just a single outgoing transition targeting a state in the same region.</td>
</tr>
<tr>
<td>Default history transition may result in an endless loop</td>
<td>The transition, outgoing the history state, targets directly or indirectly the same history state, what may result in an endless execution loop.</td>
</tr>
<tr>
<td>Empty string cannot be used as argument for InState</td>
<td>The InState function must have an argument that is a non-empty string constant.</td>
</tr>
<tr>
<td>Incomplete statemachine: a fork vertex must have at least two outgoing transitions and exactly one incoming transition</td>
<td>A fork pseudo-state may have just a single incoming transition and at least two outgoing transitions. The transitions outgoing a fork pseudo-state must target states (but not pseudo-states) in different regions of the same composite (orthogonal) state.</td>
</tr>
<tr>
<td>Incomplete statemachine: a join vertex must have at least two incoming transitions and exactly one outgoing</td>
<td>A join pseudo-state may have just a single outgoing transition and at least two incoming transitions. The transitions</td>
</tr>
<tr>
<td>transition</td>
<td>incoming a join pseudo-state must originate from states (but not pseudo-states) in different regions of the same composite (orthogonal) state.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Missing ')'</td>
<td>The number of ( and ) does not match.</td>
</tr>
<tr>
<td>Missing argument for InState</td>
<td>The InState function must have an argument that is a non-empty string constant.</td>
</tr>
<tr>
<td>Missing argument for SendT</td>
<td>The SendT function must have an argument that can be evaluated to a trigger.</td>
</tr>
<tr>
<td>No initial transition found in the enclosed region(s)</td>
<td>One or several regions of a composite state or a submachine state have no initial transition. A region without an initial transition can exist but in this case the region can become active only through an incoming transition to one of its enclosed states.</td>
</tr>
<tr>
<td>No initial transitions found on the top level</td>
<td>The state chart has no initial transition.</td>
</tr>
<tr>
<td>The outgoing transition from an initial vertex may have a behavior but not a trigger or guard</td>
<td>An initial transition may not have any triggers or guards. In general, any transition outgoing a pseudo-state may not have a trigger.</td>
</tr>
<tr>
<td>There can be at most one history vertex in a region</td>
<td>Only one history pseudo-state is allowed in a region.</td>
</tr>
<tr>
<td>There can be at most one initial vertex in a region</td>
<td>Only one initial transition is allowed in a region.</td>
</tr>
<tr>
<td>The transition incoming a join vertex may not have triggers or guards</td>
<td>Transitions incoming a join pseudo-state, may not have any triggers or guards despite the fact that they originate from a states and not pseudo-states.</td>
</tr>
<tr>
<td>The transition outgoing a fork vertex may not have triggers or guards</td>
<td>Transitions outgoing a fork pseudo-state may not have any triggers or guards. In general, any transition outgoing a pseudo-state may not have a trigger.</td>
</tr>
<tr>
<td>Transitions outgoing choice vertices must have guards</td>
<td>Every transition outgoing a choice pseudo-state must have a guard. It is recommended that one outgoing transition has the predefined else guard.</td>
</tr>
<tr>
<td>Transitions outgoing pseudostates may not have a trigger</td>
<td>Any transition outgoing a pseudo-state may not have a trigger.</td>
</tr>
</tbody>
</table>

**Warning**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial vertex has no outgoing transitions</td>
</tr>
</tbody>
</table>
## Index

### A
- Action while in a state (do action) 38
- Active and inactive states 38
- Active states in a hierarchical state chart 38
- Adding comments 21
- Adding labels 22
- Annotating state charts 21
- Applying protection to state chart 16
- Associating behaviors with states 25

### B
- Basic state chart elements 2
- Bending and moving transitions 28
- Blocks 64
- Breakpoints 71

### C
- Changing the order of internal transitions 26
- Chart-wide debugging 77
- Check Integrity 70
- choice 54
- Coloring state chart blocks 16
- Coloring states and pseudo-states 24
- Commands 70
- comment 64
- Completion transitions 41
- composite state 46
- Compound transitions 40
- Configuring states 6
- convert ID 64
- Creating a simple state chart 2
- Creating state charts 15
- Creating transitions 4

### D
- Debugging State Charts 73
- deep history 56
- Defining state chart variables 5
- Defining transition specifications 29
- Defining variable data types 17
- Defining variable scope 18
- Deleting transitions 29
- Denoting transition specifications 10
- Determining transition evaluation order 30
- Drawing a transition 28

### E
- Editing a state behavior 26
- Enabled transitions 41
- Entering a single-region composite state 38
- Entering an orthogonal (multi-region) composite state 39
- Entering and exiting composite states 38
- Entering and exiting states 38
- entry points and exit points 57
- Error and warning messages 78
- Error Navigation panel 78
- Evaluating State Charts 37
- Evaluating states 38
- Evaluating transitions 40
- Event processing: run-to-completion step 37
- Exiting a single-region composite state 39
- Exiting an orthogonal (multi-region) composite state 39

### F
- final state 53
- fork 58
- function 65
- Functional debugging 73

### G
- Guards 41

### H
- High-level transitions 40
- history 58

### I
- initial state indicator 59
- Inserting a state or pseudo-state 23
- Inserting states 3
- Introduction 1

### J
- join 60
- Jumping to the error location 78
- Jumping to the state corresponding to the state ID 21
junction 60

L
label 66
Logging messages 75
Logging messages for states 27
Logging messages for transitions 31
Looking up a state with the state ID 20

M
Moving and resizing states 23

N
Naming a state chart 15
Navigating through submachine state hierarchy 23

P
Preferences 71
Pseudo-states 54

R
Referencing triggers 33

S
Setting breakpoints for states 27
Setting breakpoints for transitions 31
Setting up state chart variables 16
Setting up the State Id data type 18
Setting up the VisSim diagram to interact with the state chart 12
simple state 43
Simulating the state chart 13
Specifying state names 24
state chart 66
State Chart Elements 43
State chart evaluation 37
state ID 69
State Id Lookup 71
States 43
submachine state 49

T
terminate 61
Transition 62
Transition execution sequence 42
Transition firing priority 40
Transition kinds 42
Transition selection algorithm 40
trigger 70

U
Using breakpoints 73
Using deferred triggers 34
Using the SendT() function to activate triggers 32
Using triggers to control a state chart from VisSim 31

V
Visual debugging 73
Visually displaying the active state 38

W
What is a state chart 1
Working with regions in composite states 27
Working with State Charts 15
Working with states and pseudo-states 22
Working with transitions 28
Working with triggers 31