VisSim/State Charts™

Design and Simulation Environment for Event-Driven Systems

Key Highlights

- Graphical state chart editor
- OMG UML 2.1 compliant
- Incorporate and control state charts in VisSim
- Represent hierarchical and parallel states with transitions
- State actions and transitions defined using standard C syntax
- Trigger state actions and transitions
- Integrated debugger
  - Breakpoints
  - Highlight active state
  - Single step simulation
  - Data logging
  - Quick Watch window
- ANSI C code generation

System Requirements

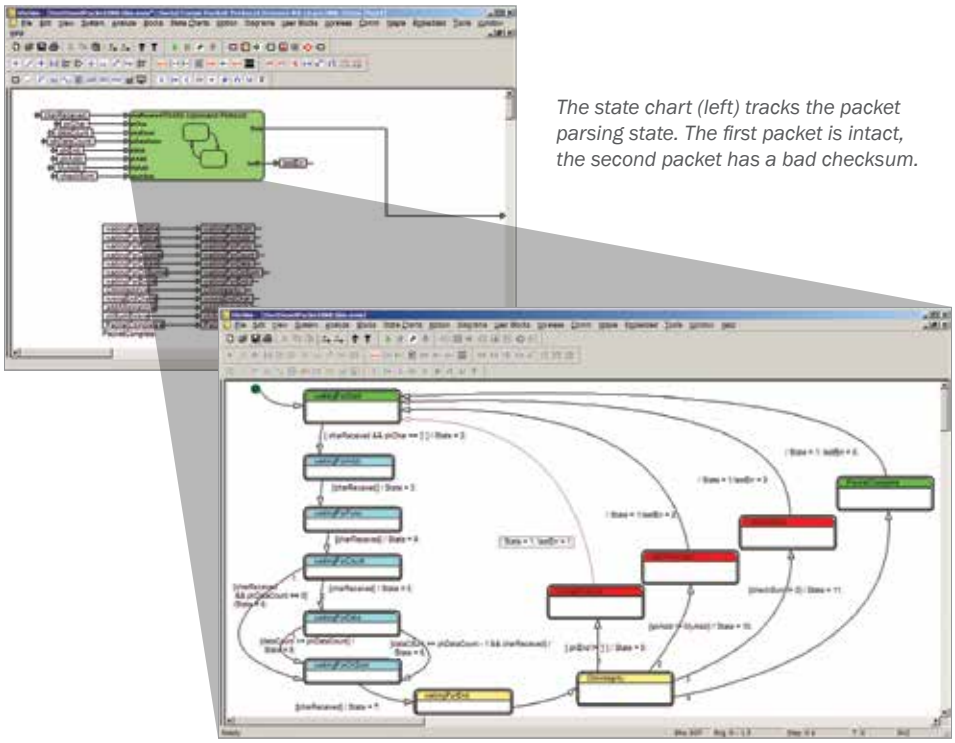
- Professional VisSim v9.0
- Windows XP, Vista, 7 or 8
- 128 MB RAM
- 125 MB hard disk space

Introduction

VisSim/State Charts lets you graphically create, edit, and simulate finite-state machines within the VisSim design environment. Finite state machines are event-driven systems that transition from one state to another based on a set of rules.

VisSim/State Charts is based on the industry-standard OMG UML 2.1. It uses common graphical elements – such as states, pseudo-states, and transition arcs – to represent simple, hierarchical, and parallel states, and the transitions among them. You can attach C code actions to state activity and transitions to further define state chart behavior.

State chart execution runs in synch with VisSim operation. Communication between a state chart and the VisSim model uses standard VisSim variables, triggers, and I/O pins on the state chart.



The state chart (left) tracks the packet parsing state. The first packet is intact, the second packet has a bad checksum.

VisSim continuous model (top) simulates two serial packets in a character stream. Packets consist of a start char, address char, command char, data payload count, variable length data, checksum byte, and end char. The VisSim model monitors the currently active state in the state chart to see if a remote command is active and responds appropriately.

VisSim State Charts provides an intuitive method for developing system control. The graphical editor along with standard C syntax combine to offer a familiar design entry experience. Behavioral operation of the control can easily be simulated before targeting the embedded hardware.

Randall Pipho, Senior Design Engineer, TECO Westinghouse Motor Company

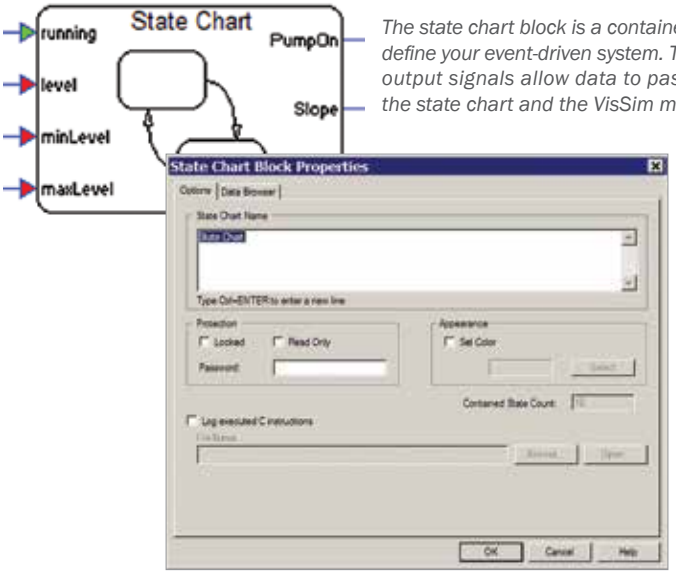
Defining the VisSim Model – State Chart Interface

In VisSim, a state chart appears as a standard block with input and output pins. It is through these connections that the state chart and VisSim model share data.

You can have multiple state chart blocks in a single VisSim diagram. Each state chart block corresponds to a single state chart.

A state chart has its own hierarchy. The highest level of hierarchy in the state chart is the state chart block.

To edit or view the structure of the state chart, you "drill" into it, similar to the way that you drill into standard block hierarchy.



The state chart block is a container in which you define your event-driven system. The input and output signals allow data to pass between the state chart and the VisSim model.

CTRL+right-click over a state chart block to access customizable parameters and options.

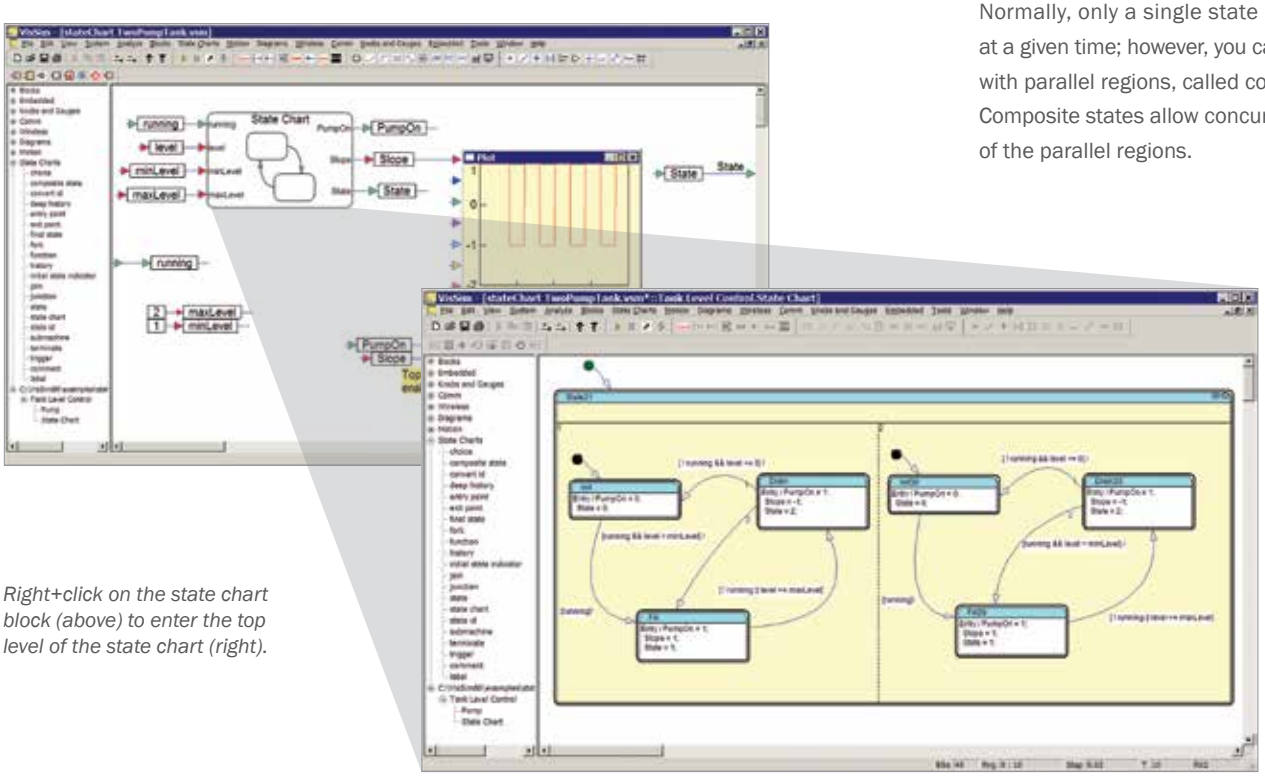
Defining States and Pseudo-States

A state describes a period of time during which a specific mode of operation is executing. The number of states in state chart depends on the number of operating modes. For example, a simple three-state pump would have these operating modes: idle, filling, and draining. Pseudo-states are used to connect transitions into more complex state transition paths, such as dynamic and static conditional branches, forks, and joiners. Unlike states, psuedo-states are not occupied for any period of time.

State Hierarchy

Organizing state chart elements into logical subsystems lets you build charts with hierarchy. States that contain other states are called submachine states.

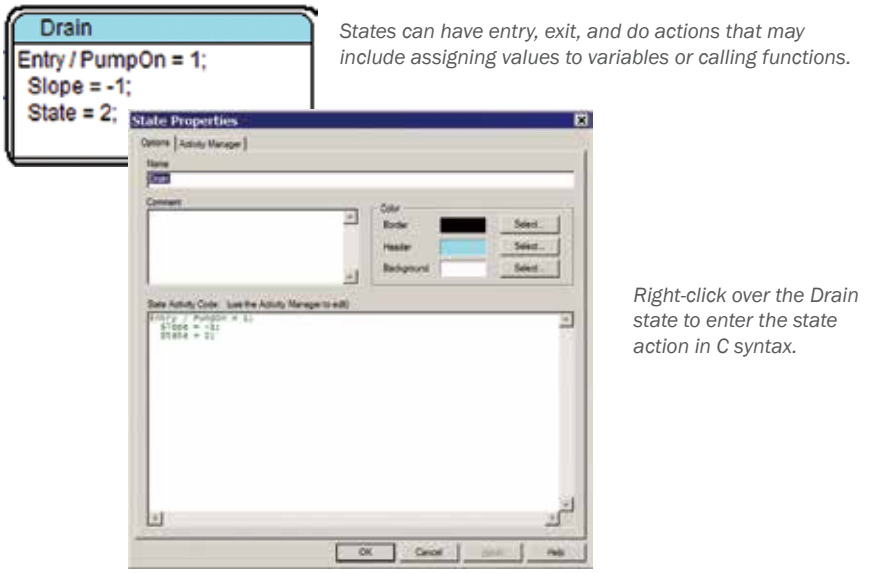
Normally, only a single state can be active at a given time; however, you can create states with parallel regions, called composite states. Composite states allow concurrent execution of the parallel regions.



Right+click on the state chart block (above) to enter the top level of the state chart (right).

Defining State Actions and Variables

States can perform actions while they are active. If a state has actions, the actions are executed upon entering the state, exiting the state, or while in residence. State actions are defined using standard ANSI C operations.



States can have entry, exit, and do actions that may include assigning values to variables or calling functions.

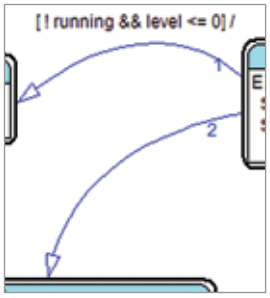
Right-click over the Drain state to enter the state action in C syntax.

Defining Transitions

Transitions are paths from one state or psuedo-state to another on which the logic of the system flows. Transitions are represented as lines with an arrowhead at one end, indicating the direction of signal flow.

You can apply conditional guards to transitions that are evaluated dynamically. When the condition evaluates to TRUE, the transition occurs.

Like state actions, conditional guards are defined in standard C syntax.



Transition 1 fires when the guard expression within brackets is TRUE.

Using Triggers to Control State Chart Execution

Triggers allow you to control the execution of state charts. Triggers can be part of a transition specification. Transitions with triggers automatically have higher priority than those without triggers.

Debugging a State Chart

VisSim provides debugging tools for examining, locating and correcting inconsistent state chart behavior. You can step through the simulation, one time step at a time. You can also set breakpoints that halt the simulation when a selected state action or transition is activated. The Quick Watch window displays the current values of the selected variables and expressions.

Generating Production-Quality Code

If you have installed the VisSim/C-Code module (available separately), you can generate efficient, in-line code from your state chart. This code communicates seamlessly with the continuous portion of the VisSim diagram, and it allows you to incorporate your state chart into your embedded controller.

ANSI C code generated from a state chart.