

Telelogic
StateMate®

Continuous Modeling



IBM®

Statemate[®]

Continuous Modeling



Before using the information in this manual, be sure to read the “Notices” section of the Help or the PDF file available from **Help > List of Books**.

This edition applies to Telelogic Statemate 4.5 and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 1997, 2008.

US Government Users Restricted Rights—Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Continuous Modeling With VisSim	1
Hybrid Systems	1
Pure Discrete Logic System	2
Discrete Logic System Interacting with a Physical Environment (Hybrid System Type A)	2
Discrete Logic System Containing Some Dynamics and Reacting to a Physical Environment (Hybrid System Type B)	2
Implementation Overview	3
Licensing	3
Windows Installation	3
Hybrid Modeling	4
Graphical Representation	5
Top Down or Bottom Up Workflow	6
Simulation and Code Generation	7
Check Model	10
Workarea and Databank	10
Printing Continuous Diagrams	10
Documentor	11
New Dataport Queries	13
Reports Tool	13
Session Status	13
DOORS/RTM Interface	14
Properties	14
A Simple Example	15
Set Up the Project	16
Create the Continuous Diagram	17
Check the Model	20
Bind Formal Parameters	22
Run the Simulation	24

Execution Semantics	29
Key Concepts	29
Basics of Execution	30
Continuous Activity Behavior	30

Continuous Modeling With VisSim

StateMate has traditionally dealt only with discrete, state-based system designs. These designs, however, do not necessarily reflect the real world for all possible applications. To address the need for a continuous time modeling capability, StateMate uses Visual Solutions' VisSim.

Note

According to Visual Solutions, "VisSim is a Windows-based program for the modeling and simulation of complex dynamic systems. VisSim combines an intuitive drag & drop block diagram interface with a powerful simulation engine. The visual block diagram interface offers a simple method for constructing, modifying and maintaining complex system models. The simulation engine provides fast and accurate solutions for linear, nonlinear, continuous time, discrete time, time varying and hybrid system designs."

The term *continuous*, in this context, is used to mean those segments of a system whose behavior is described mathematically, i.e., by means of differential/integral equations, feedback loops, control law theory elements, S space and Z space transformation functions, etc.

Hybrid Systems

Most embedded systems are made up of discrete logic behavior that can be described by StateMate:

- ◆ State-based behavior (statecharts)
- ◆ Sequential logic algorithms (graphical subroutines)
- ◆ Combinational logic (Truth Tables)

However, many embedded systems also require a continuous time notation to describe the physical/mechanical elements they interact with or to represent the control equation. A hybrid system is one that contains both continuous time behavior and discrete logic behavior. In this context, embedded systems can be divided into three conceptual groups:

Pure Discrete Logic System

This type of system contains no control law algorithms and interacts with no continuous environmental elements. Its behavior is best described using a finite-state machine notation (Statecharts), decoding and encoding logic (Truth-Tables), decision making algorithms (Graphical Procedures) and basic equations (mini-specs).

The environment is best described in this type of system using a panel to represent the operator interface and using the above notations to represent the discrete elements that it interacts with (i.e. bus protocols etc.). Examples of this type of system would be an exterior lighting system in a car or an electronic display unit.

Discrete Logic System Interacting with a Physical Environment (Hybrid System Type A)

This type of system contains no control law algorithms but interacts with continuous environmental elements (often called a plant). The behavior is best describe using a finite-state machine notation (Statecharts), decoding and encoding logic (Truth-Tables), decision making algorithms (Graphical Procedures) and basic equations (mini-specs).

In this case the environment may have an operator interface best described using a panel and may interact with discrete elements. However, it may also contain some continuous elements (often called plants) best described using the continuous time simulation block diagram notation. An example of this type of system would be a garage door controller, monitoring a moving door position.

Discrete Logic System Containing Some Dynamics and Reacting to a Physical Environment (Hybrid System Type B)

This type of system is similar to type A, except that it will also include some continuous controllers. An example of this might be an automatic transmission controller in which the system would be functionally decomposed using Statemate. The high level modes (1st, 2nd, 3rd, reverse) of the system would be described using Statecharts, the user inputs would be described using a panel and would be decoded using truth-tables, the clutch control algorithm however would be described using the continuous time block diagram, and the external power train would also be described using a continuous time block diagram.

Implementation Overview

StateMate's Continuous Modeling module uses a version of VisSim that has been engineered specifically for this purpose and that only can be invoked from within StateMate. You cannot use stand-alone instances of VisSim for this purpose.

As the ability to do hybrid simulations (and to generate code from these systems) depends on the ability to compile code generated by VisSim, make sure you are able to use the usual "external code" feature in StateMate simulation. For example, if you do not have an available C compiler for the StateMate external code, you cannot compile and run the VisSim generated code.

Licensing

The StateMate/VisSim interface is activated when you purchase and use a license. If you do not have a license, the user interface elements (such as menu items, dialog boxes etc.) that are related to this module are hidden or disabled.

Windows Installation

The StateMate installation asks you if you want to install VisSim. If you click yes, it installs VisSim in your `STM_ROOT/vsm` directory. Once the VisSim installation and the StateMate installation are finished, you are ready to run.

Changing the VisSim Location

Modification of the default install location for VisSim is not recommended. If you choose to do so, manually modify your `run_stmm.bat` file, so that `STM_VSM_ROOT` is pointing to the installation location.

Client/Server Installation

You can install the Continuous Modeling module in a client/server environment. The VisSim software files will be installed on the client system, as well as on the server.

Hybrid Modeling

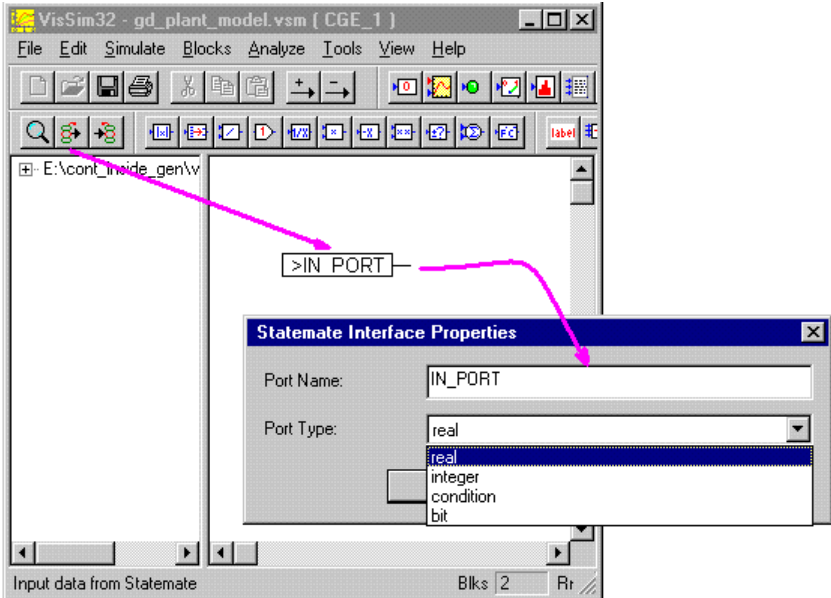
Continuous diagrams are referenced from activity-charts. The reference is done in a similar way to an instantiation of a generic activity-chart (i.e., by using generic instance activities).

Note

In StateMate 2.0.1, you cannot create a component that includes instances of a continuous diagram.

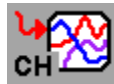
The interface between a continuous diagram and the rest of the model is done via the parameter mechanism of generic instances and charts, using formal to actual binding. This binding supports signals of type real, integer, bit and condition. The supported port directions are in and out.

The formal parameters list of a continuous diagram (as seen in the properties) is read-only. The list is actually a reflection of the “statemate port” blocks of the continuous diagram. To define “statemate ports” in VisSim, choose **Blocks > StateMate** from the menu or use the toolbar buttons as shown in the following figure.

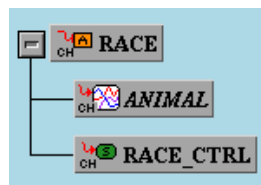


Graphical Representation

A continuous diagram is graphically represented by unique bitmap (shown in the following diagram) in all StateMate tools that display a tree view of a scope (Workarea Browser, profiles, and so forth).



In the Workarea Browser, continuous diagrams are much like generics. You will see a continuous diagram as an instance. If it does not exist in your work area, it is “unresolved” as in the following diagram:



In fact, a continuous diagram is always a generic style chart and each new continuous diagram you create or import is by definition a chart, which can be instantiated.

All the operations used on charts from the Workarea Browser are available for continuous diagrams: view abilities as editing, printing, manipulations such as rename, copy, delete and version control operations.

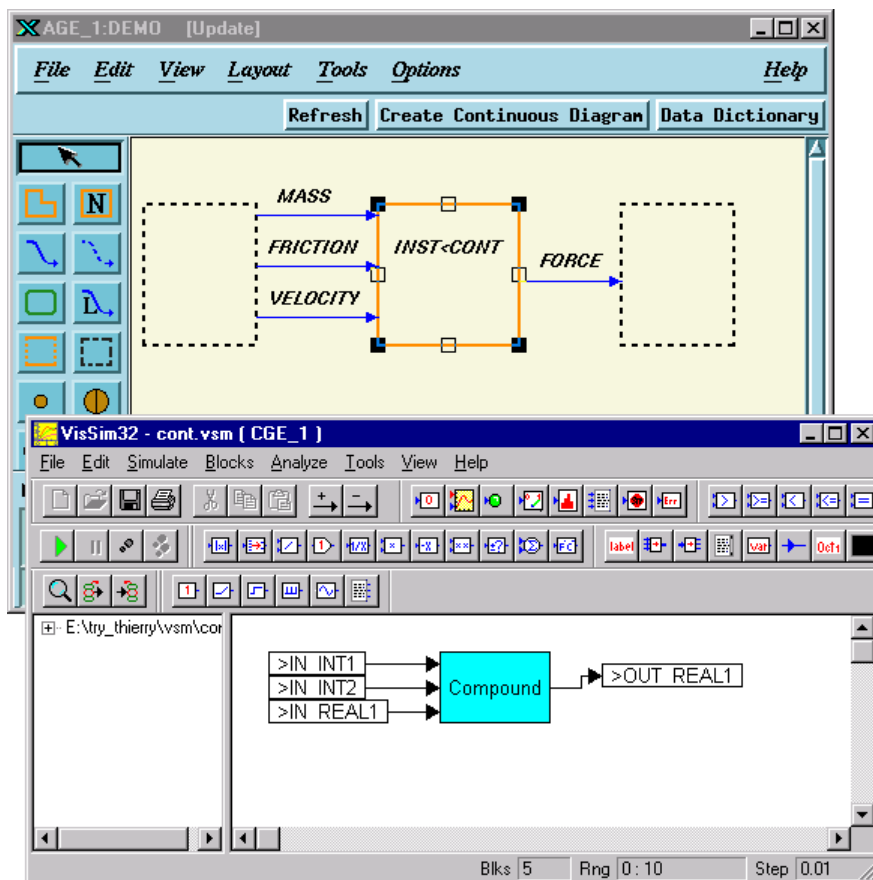
Top Down or Bottom Up Workflow

If you choose to describe your system in a top down workflow, you can create a continuous diagram as demonstrated in the following figure.

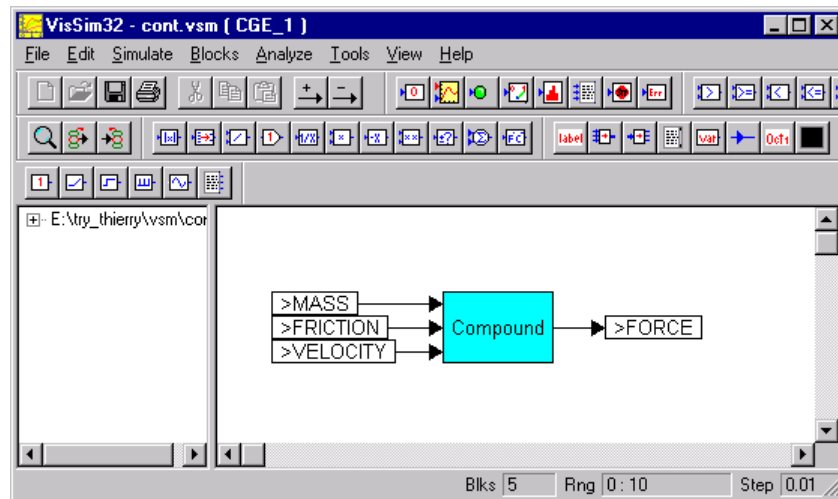
If you connect the instance with flow lines, you will automatically get a continuous diagram with the matching set of ports, which are given generic names. This port list is only a starting point and evolves later in the design cycle.

To modify the feature so that the “Statestate port” names in the continuous diagram are identical (or sometimes similar) to the actual parameters, simply set (in your `run_stmm.bat`) `VSM_NONGENERIC_PORT_NAMES` to ON.

For example, the activity chart below contains an instance of continuous activity (`inst<cont`), and indicates by flow lines that the inputs are mass, fractionated velocity, and the output is force (these are the actual parameters). Selecting the instance and choosing “create continuous sub chart” creates a new continuous diagram that looks like this:



However, if `VSM_NONGENERIC_PORT_NAMES` is set, it will look like this:



The type of the “stateport” blocks in VisSim reflect the Stateport data type.

If you already have a continuous diagram with its ports (for example, you imported it), and you wish to create an instance of it (bottom up workflow), you can simply create an instance of it, and in the instance properties, use the “fill formals” button to begin the binding phase.

Simulation and Code Generation

As you add charts containing continuous diagrams to your simulation scope, these elements are included automatically in the Simulation/Codegen. The continuous diagrams are shown graphically in the tree area of the Profile Editor.

The simulation is actually using code generated by VisSim, as external code to the simulation. Thus, the Stateport simulation does not use the VisSim editor to simulate the continuous diagram, but instead runs a code segment.

Note

For the vissim code generator to work correctly, the contents of the continuous diagram should be of the structure:

`stm_inputs` → compound block → `stm_outputs`

In other words, the uppermost level of your continuous diagram must be a single compound block with an optional set of input and output ports.

All formal/actual bindings (signal mappings) are taken automatically from the binding information provided by the user. If there is some kind of binding problem, the instance is ignored.

If a continuous diagram was modified but not saved, Statemate automatically updates the simulation and code generation.

Note

Check Model reflects binding errors, but does not check for unsaved port modifications done inside VisSim.

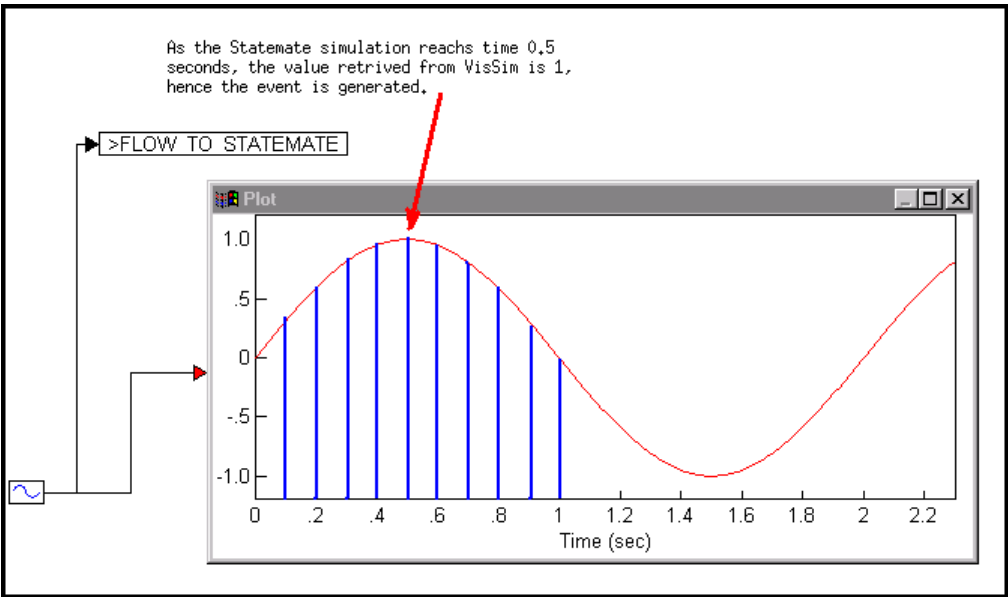
Time Settings

In the time setting dialog box (asynchronous), a new field is used: “sampling rate”. The sampling rate defines the times of data exchange between Statemate and VisSim. For example, consider a sine wave generated by VisSim, with period time of 2 seconds, and its output value is monitored by Statemate. The Statemate controller generates an event each time the sine wave exceeds a threshold of 0.99.

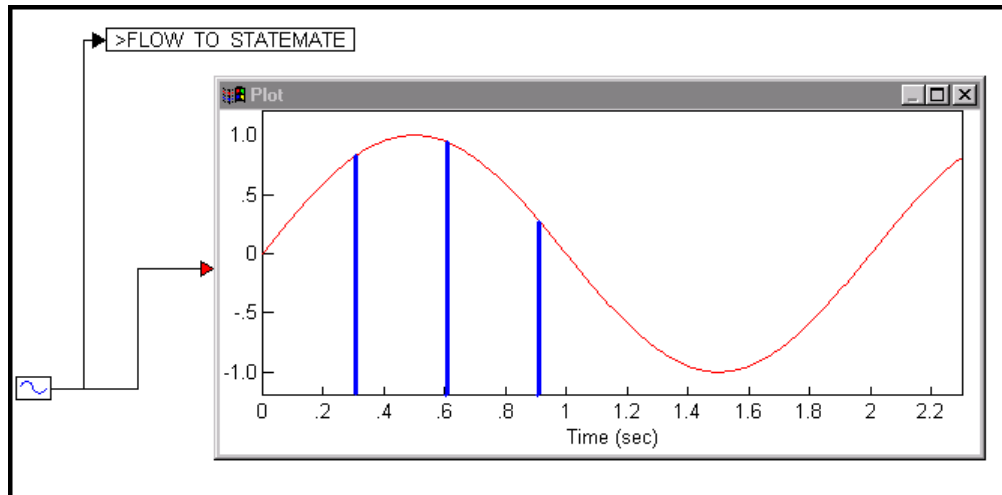
Note

In synchronous mode, the “step size” field is used for the sampling rate. Synchronous simulation and code generation is recommended for HDL-oriented designs only. It is recommended that you do not use synchronous mode simulations or code generation for continuous modeling.

The following two pictures explore the first period of the sine wave. Using a sampling rate of 0.1 second, a threshold cross is detected, and the event is generated after 0.5 seconds



However, using a larger sampling rate (say 0.3 second) the threshold cross is missed and the event is not generated; the output value sensed in Statemate is too small on both time=0.3 seconds and time=0.6 seconds



Setting a sampling rate that fits best to your system properties and simulation needs is an important task in hybrid system design.

Any other necessary VisSim simulation settings that are not given in the Statemate user interface (like VisSim internal integration step size) must be set via the VisSim user interface.

Debugging Tools

All the peripheral tools that accompany Statemate simulation and code generation can be used to debug the combined model. This includes monitors, charts animation, GBA, panels etc. These tools cannot display VisSim internal signals; they can only display the ports.

When using the waveform, use the TIME STAMPS feature to see the time advance while the continuous diagram code is running.

Limitations

- ◆ Simulation: The “go back” command is not supported by VisSim. This is also true for save status and restore status.
- ◆ Simulation and code generator: monitors, panels etc. cannot reflect internal VisSim parameter values. They can only reflect the “ports.”
- ◆ Code generator: the “real time” option is not supported.

Check Model

A Check Model profile activated on a scope containing a continuous diagram reports binding problems (similar to generics), and internal continuous diagram problems, detected and reported by VisSim.

Note

Check Model does not check for unsaved port modifications done inside VisSim.

Workarea and Databank

Continuous diagrams are Statemate configuration items. As such, they can be loaded from (and stored to) your Statemate databank. All configuration management operations (such as version control, check-in, check-out with or without lock, import, export etc.) are applicable for these charts.

A continuous diagram (a VisSim model) can refer to external files, such as .dat files, .mat files, .m files, .wav files, and so forth. (For more information on these files and their functionality, refer to the VisSim documentation.) These files are also Statemate configuration items. They are referenced by their filename and are loaded automatically into your workarea whenever a model is loaded.

Printing Continuous Diagrams

Printing from the Workarea Browser is done in the usual way, i.e., select a continuous diagram and choose **File > Print Selected...** However, only the following fields of the print option dialog are supported:

- ◆ With Elements' Properties
- ◆ Device Name: Postscript / Word
- ◆ Send to printer (for Postscript)
- ◆ Number of copies
- ◆ Write to file
- ◆ Header
- ◆ Orientation

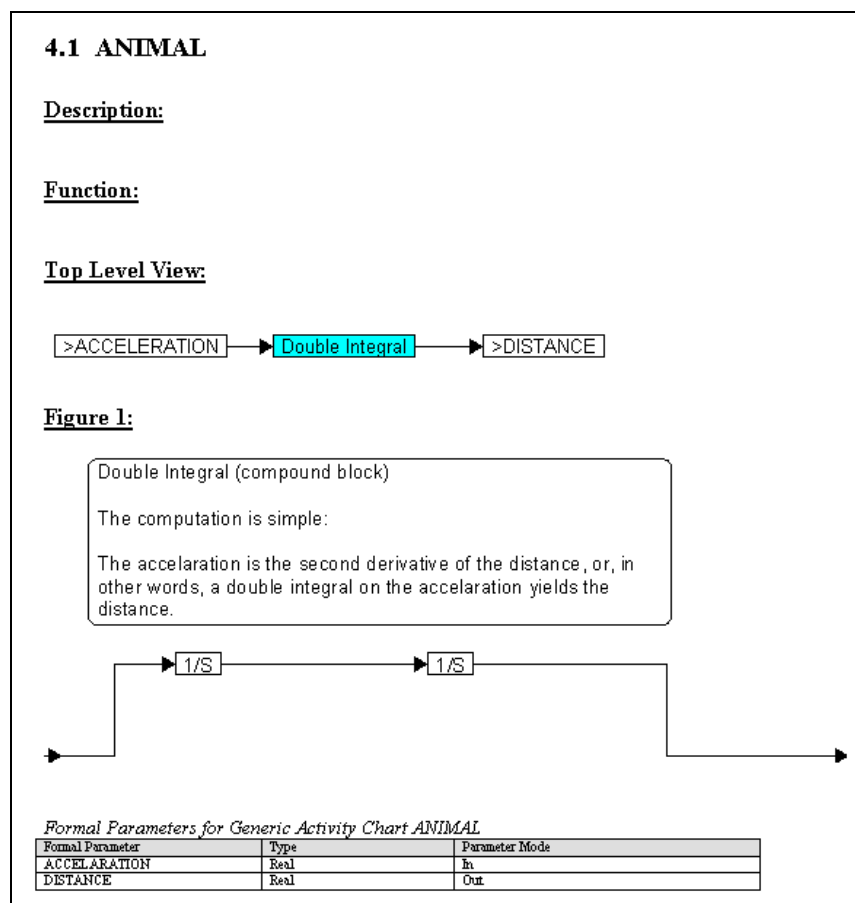
The internal VisSim print options support a larger collection of options and settings.

On Windows, print commands sent from Statemate to VisSim work only if a printer is installed on the PC. The printers defined in the “output device” list of Statemate do not necessarily work for continuous diagrams.

Documentor

The standard RTF/Word documentor template reflects continuous diagrams. A new chapter named “Continuous Diagrams” has been added to the generated document. Instances of Continuous Diagrams are described in the same manner as Generic Activities (as shown below) and the chapter is organized in the same manner as “Generic Activity Charts.”

You can choose to include only the top level of the continuous diagrams or all the levels of the compound blocks (as shown below). In the file `run_stmm.bat`, set the environment variable `VSM_BLOCKS_PLOT` to ON.



Segments containing Postscript plots of the VisSim diagram can be also implemented by the user.

Output Document Portability

The output of the StateMate Documentor is an RTF document that contains links to VisSim plot files, rather than the plot files themselves. VisSim generated plots are WMF (Windows Metafile Format) files located in the workarea/doc directory. Each time you run the Documentor, these files are overwritten.

To make a document portable, it is recommended that you:

- ◆ Embed the VisSim plot files in the document.
- ◆ Save the document in DOC (rather than RTF) format.

To embed VisSim plot files in a document, you must “break the links.” This is the procedure (in Word97):

1. Click **Edit > Links...**
2. Select all links in the Links dialog box.
3. Click **Break Link** (and confirm).

New Documentor DGL Queries

Three new DGL functions were added to support Continuous Diagrams:

- ◆ **STM_R_AC_CONTINUOUS_INSTANCE_AC**
Returns those activities in the input list that are continuous chart instances.
- ◆ **STM_R_CH_CONTINUOUS_CH**
Returns the continuous charts in the input list.
- ◆ **STM_R_CH_DEFINING_CD_INST_AC**
Returns the chart that defines the Continuous Diagram (CD) instance activities in the input list.

New Dataport Queries

Three new Dataport functions were added to support Continuous Diagrams:

- ◆ `stm_r_ac_continuous_instance_ac`
Returns those activities in the input list that are continuous chart instances.
- ◆ `stm_r_ch_continuous_ch`
Returns the continuous charts in the input list.
- ◆ `stm_r_ch_defining_cd_inst_ac`
Returns the continuous chart that defines the Continuous Diagram (CD) instance activities in the input list.

These queries are available from dataport programs, not from the Databank Browser.

Reports Tool

StateMate reports reflect relevant information on a continuous diagram in the report scope. In some reports, continuous diagrams are referred to as “generics.”

Session Status

A continuous diagram can be monitored and aborted in the Session Status window, and is referred to as CGE (Continuous Graphic Editor).

DOORS/RTM Interface

In general, exporting a continuous diagram to DOORS or RTM, through the DOORS or RTM interfaces in Statemate 2.0.1 is similar to exporting any other chart. However, there are a few differences:

- ♦ **Plot File** - No plot file is created for the exported continuous diagram. After the export the attribute “Chart Plot” in DOORS remains empty.
- ♦ **RT MODULE Attribute** - The module or class to which the chart is exported is determined by the name defined in the attribute RT MODULE in the chart's Properties Editor form. Since attributes cannot be added manually to continuous diagrams, the chart is exported to a module or class matching its name.
 - Note:** In RTM, a class with the chart name must be created before any export attempt.
- ♦ **Chart's Elements** - After the export, a matching object is created in DOORS or RTM that represents the continuous diagram. There are no objects of that kind for the chart's elements.
- ♦ **RT Interface Preferences** - All of the RT Interface preferences are irrelevant for continuous diagrams.

Properties

You can view a continuous diagram in the Properties. The Statemate element stores the formal ports defined in VisSim. However, this chart is read only for Statemate tools. The only way to modify the formal ports on a continuous diagram is by using VisSim. The ports stored in Statemate are automatically update when you save the continuous diagram. The ports are actually data items defined in the chart.

The chart also holds a placeholder subroutine automatically implemented by Statemate. This is also a read only element and is part of the interface implementation.

The “long description,” “short description,” and attributes are also read only and are not accessible to the user.

A Simple Example

This section presents a very simple example that demonstrates the fundamentals of using continuous notation. It is strongly recommended that you treat this example as a tutorial and follow along on your own workstation.

The example simulates a race between a cat and mouse. Before the race begins, both animals are held in cages, some distance (in meters) away from a wall, where the mouse's hole is.

When the mouse's cage is opened, it runs towards its hole. A few seconds later, the cat is released. The cat (who accelerates faster) tries to catch the mouse before it reaches the hole in the wall.

The goal is to explore the system's behavior (who won? when? how far from the beginning?) using input values for parameters such as race distance, time gap size, accelerations of both the cat and mouse, and so forth.

For convenience, the required activity chart, statechart, and panel are provided in the Statemate online reference library.

Note

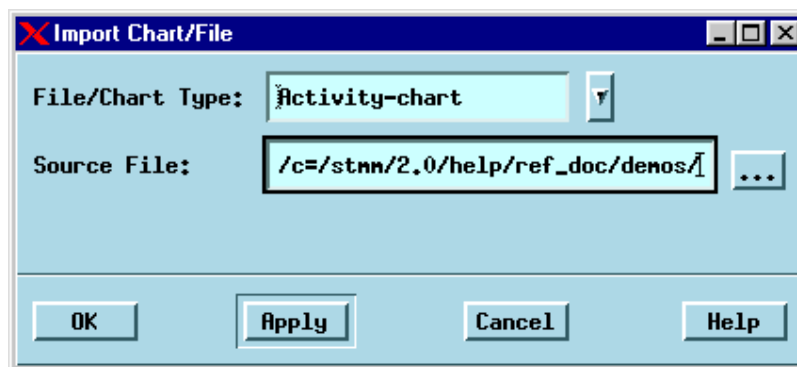
For simplicity, the statechart ignores the case where the mouse reaches the wall before the cat is released and the case where they both reach the wall at the same time (a tie).

Set Up the Project

Complete the following steps to set up the project:



1. Create a new project to hold the example (**File > New Project...**).
2. Open a Workarea Browser
3. Click **File > Import > Import from File...** The Import Chart/File dialog box opens.



4. Import the following three files from:

```
%STM_HELP%\ref_doc\demos\vissim\race\
```

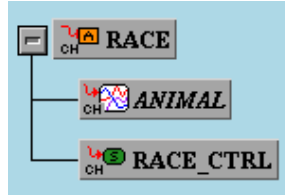
Note: Four files are provided. Do not import `animal.vsm` at this point.

- ◆ Activity chart: `race.ach`
- ◆ Statechart: `race_ctrl.sch`
- ◆ Panel: `race.pnl`

Note:

- ◆ `%STM_HELP%` is the directory containing the Statemate on-line help and reference library. It is usually located directly under `%STM_ROOT%` (the directory in which Statemate is installed). For example: `c:\stmm\2.0\help\`
- ◆ Make sure to set the **File/Chart Type:** correctly for each file. In this example, the panel is given the name **RACE**.

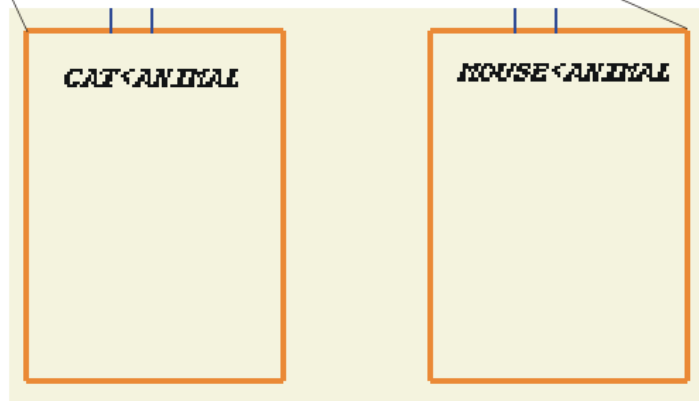
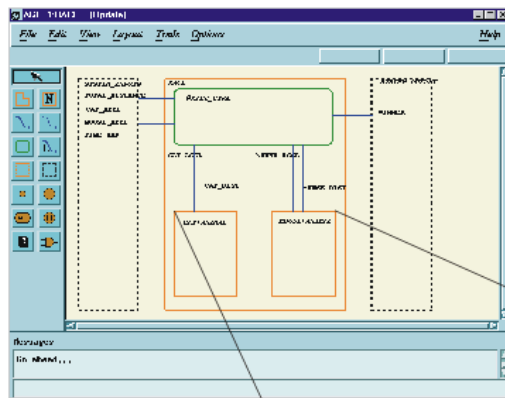
- When all three files have been imported, the workarea tree looks like this:



Create the Continuous Diagram

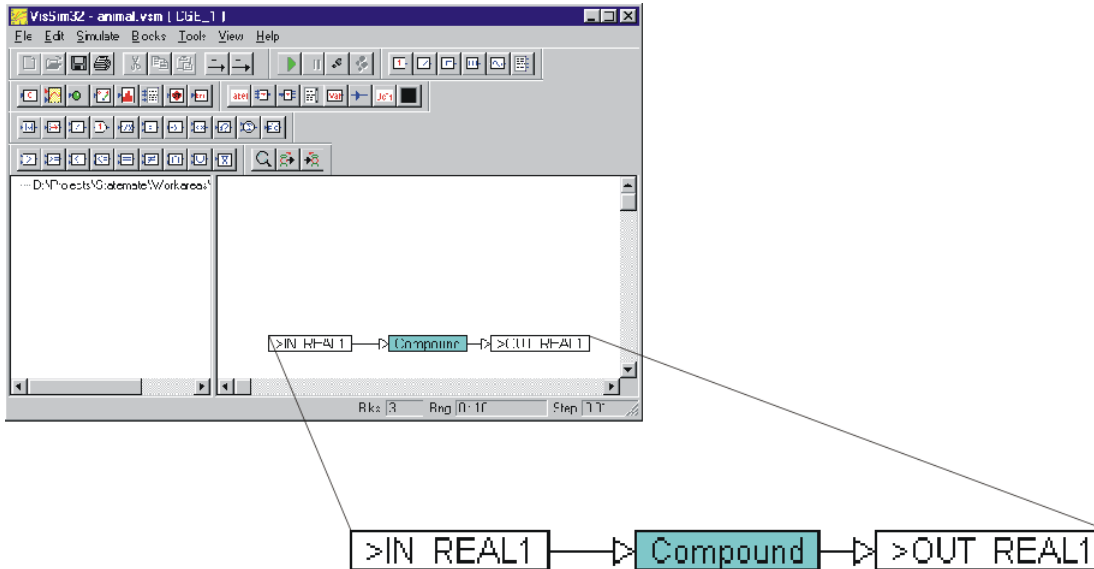
- Double-click the activity chart (RACE) to open the Activity Chart Graphic Editor.

Note that both CAT and MOUSE are instances of ANIMAL, which is an unresolved continuous diagram as can be seen in the Workarea Browser.



A Simple Example

2. In the activity chart, select either the CAT or the MOUSE and select **File > Create Continuous Diagram**. The Continuous Graphic Editor (VisSim) opens showing a continuous diagram, with a single empty compound block.



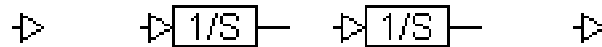
3. Double-click the **Compound** block to open it. The block contains two arrows.



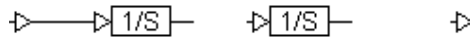
Note: To define an ANIMAL's behavior, assume that its motion is governed by the simple equation:

$$Distance = \iint Acceleration$$

4. Select **Blocks > Integration > integrator**, position the pointer between the arrows, and click to create an integrator block. Then add a second integrator block to the right of the first one.



5. Position the pointer next to the leftmost arrow until it becomes an up-arrow (↑).
6. Hold down the **Select** button and drag the pointer (drawing a green connector line) to the first integrator. When you release the button, the connector changes from green to black.



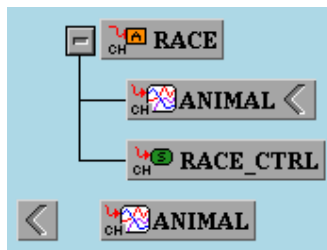
7. In the same manner, connect the first integrator to the second integrator, and the second integrator to the right-most arrow.



8. Right-click to close the compound block.




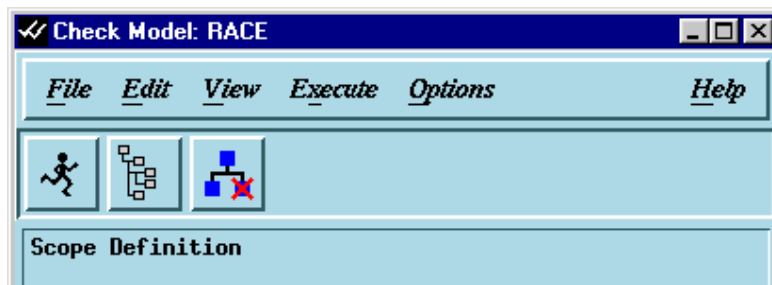
9. Close the editor and save your design. The workarea tree now looks like this:



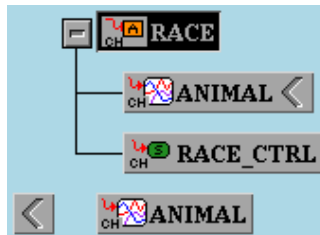
Check the Model

Normally, you would not use the Check Model tool at this point since you have not yet bound the continuous block's parameters to variables in the StateMate model. The purpose of running it here is to demonstrate that the scope of the tool does indeed include continuous blocks.

1. Click **Check Model**. 
2. Select **File > New Profile...** and name the new profile **RACE**. The profile is initially empty.



3. Click **Connect this Profile Editor to Workarea Browser**.
4. In the Workarea Browser, select **RACE**.



5. In the Profile Editor, click **Add Selected Chart (WAB) to Profile With Descendants**.
6. Select **File > Save** to save the profile.
7. Click **Execute Current Check Model Profile**.

Note

The report indicates that both instances of the continuous diagram (ANIMAL) have **bindings missing**. Ignore the rest of the report.

Correctness:**o (C3135) Continuous instances with inconsistent parameter binding:**

- activity CAT instance of ANIMAL

Inconsistent number of ports:

2 ports in model, while 0 ports in its instance.

IN_REAL1 has no actual parameter binding.

OUT_REAL1 has no actual parameter binding.

- activity MOUSE instance of ANIMAL

Inconsistent number of ports:

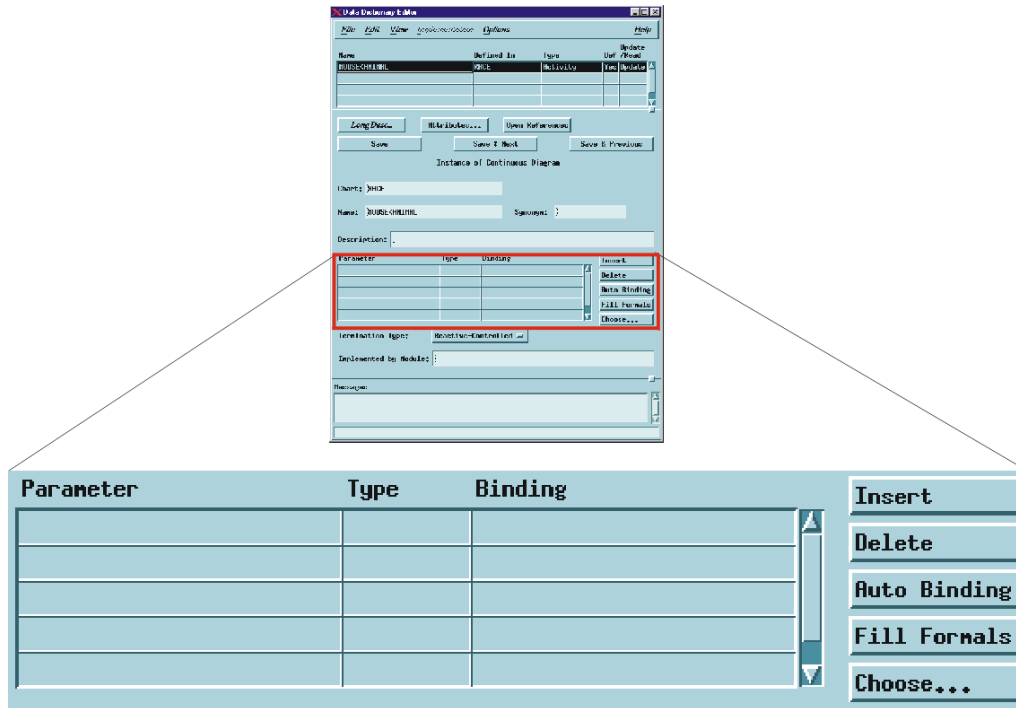
2 ports in model, while 0 ports in its instance.

IN_REAL1 has no actual parameter binding.

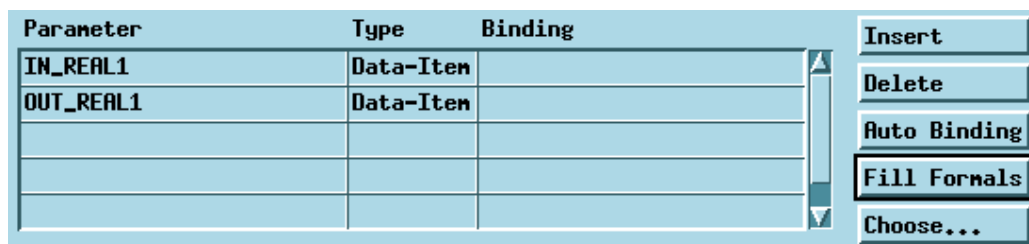
OUT_REAL1 has no actual parameter binding.

Bind Formal Parameters

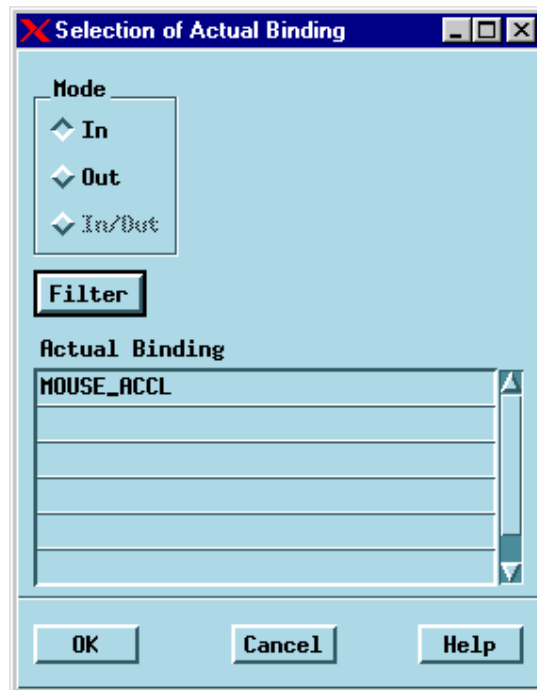
1. In the Activity Chart Graphic Editor, select **MOUSE** and click **Tools > Properties**.



2. In the Parameter section, click **Fill Formals**. The continuous block's parameters appear in the list.



3. In the parameter list, select **IN_REAL1** and click **Choose...**
4. When the **Selection of Actual Binding** dialog opens, click **Filter**.



5. Select the matching parameter (in this case **MOUSE_ACCL**) and click **OK**.
6. In the parameter list, select **OUT_REAL1** and click **Choose...**
7. Set the **Mode** to **Out** and click **Filter**.
8. Select **MOUSE_DIST** and click **OK**.
9. Save the activity chart (**File > Save**).

Note

You may optionally click **Check Model**



to check the bindings.

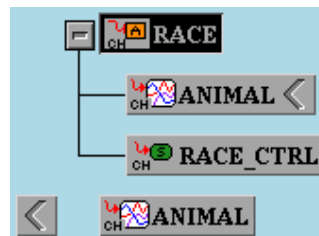
Run the Simulation

1. Start the **Simulation Profile Editor**.
2. Click **File > New Profile...** and name the new profile **RACE**. The profile is initially empty.



3. Click **Connect this Profile Editor to Workarea Browser**.

4. In the Workarea Browser, select **RACE**



5. Click **Add Selected Chart (WAB) to Profile With Descendants** in the Profile Editor.

6. In the Workarea Browser, select the **panel RACE** (visible only in the files area).

RACE_CTRL	Statechart	Update	1	
RACE	Activity-chart	Update	1	Modified
ANIMAL	Cont. Diagram(G)	Update	1	Modified
RACE	Panel	Update	1	
RACE	Analysis Profile	Update	1	
RACE	Chk.Model Profile	Update	1	



7. Click **Add Selected Panel (WAB) to Profile** in the Profile Editor.

8. Select **File > Save** to save the profile.



9. Click **Invoke a Simulation Based on the Current Profile**.

A command prompt window opens as the simulation code compiles.

```

C:\WINNT\system32\cmd.exe - make_script.bat
D:\Projects\Statemate\Workareas\race\ana\race2>nmake
Microsoft (R) Program Maintenance Utility   Version 1.62.7022
Copyright (C) Microsoft Corp 1988-1997. All rights reserved.

        cl -ID:\stmm\2.0\vsm\cg\include -ID:\stmm\2.0\vsm\vsdk\include /c user_code.c
ode.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 11.00.7022 for 80x86
Copyright (C) Microsoft Corp 1984-1997. All rights reserved.

user_code.c
        cl -ID:\stmm\2.0\vsm\cg\include -ID:\stmm\2.0\vsm\vsdk\include /c animal
.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 11.00.7022 for 80x86
Copyright (C) Microsoft Corp 1984-1997. All rights reserved.

animal.c
        cl -ID:\stmm\2.0\vsm\cg\include -ID:\stmm\2.0\vsm\vsdk\include /c vsm_cg
_file_animal.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 11.00.7022 for 80x86
Copyright (C) Microsoft Corp 1984-1997. All rights reserved.

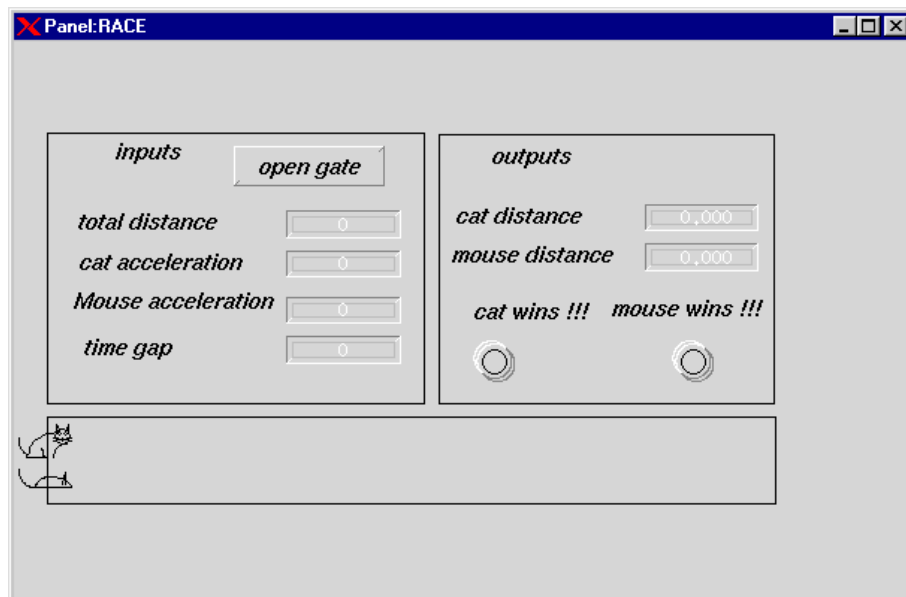
vsm_cg_file_animal.c
        cl /nologo D:\stmm\2.0\lib\communicate.obj D:\stmm\2.0\lib\prepare_vsm
task.obj user_code.obj animal.obj vsm_cg_file_animal.obj  D:\stmm\2.0\lib\libsc
heduler.lib D:\stmm\2.0\lib\X11.lib D:\stmm\2.0\vsm\cg\lib\cgvsmstm.lib  kernel3
2.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.lib ol
e32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib libcd.lib /o user_code.exe
        Creating library user_code.lib and object user_code.exp
Press any key to continue . . .

```

A Simple Example

10. When the window displays **Press any key to continue...** make sure the code compiled and linked successfully, then press a key and the window closes.

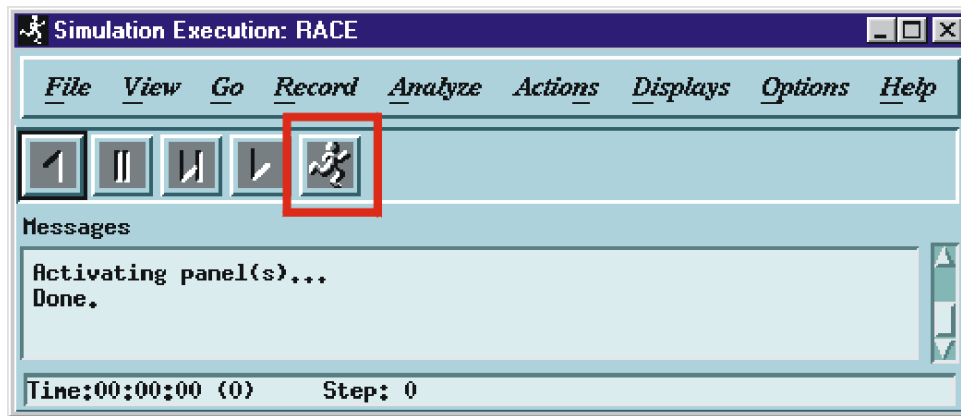
The RACE panel opens.



11. Try various sets of inputs starting with:

total distance	10
cat acceleration	0.5
mouse acceleration	0.3
time gap	1 or 3

12. Click **Auto Run Command** in the Simulation Execution window.



13. In the RACE panel, click **open gate** and watch the race.

open gate

Execution Semantics

This section describes the mathematical assumptions and methods of combined simulation and code generation.

Note

Simulation and generated code have the same behavior.

Key Concepts

As you have seen, combined continuous/discrete modeling is strongly based on the Statemate mechanism of generic charts. It is very important to understand that simulation and code generation are also strongly based on an existing Statemate mechanism: external code and tasks/subroutines. When you simulate a Statemate scope containing a continuous diagram, Statemate automatically generates code for the continuous part and links it to your simulation.

The semantics are those of a sampling system: values of parameters are exchanged between Statemate and VisSim, using a sampling rate defined in the simulation/code generation profile. (Refer to [Time Settings](#).)

Basics of Execution

- ◆ An instance of a continuous diagram can be (like any other activity) active or not active (refer to [Continuous Activity Behavior](#) for details).
- ◆ As an instance of a continuous diagram is activated at time T , it registers a scheduled event to time $T + \text{sampling_rate_unit}$, and thus requests Statemate to activate it as it reaches time $T + \text{sampling_rate_unit}$.
- ◆ This scheduled event registration can be viewed using the Statemate simulation “show future” option.
- ◆ When Statemate runs the continuous instance, it runs from time T to time $T + \text{sampling_rate_unit}$ (which is, in fact, the current Statemate time) and registers another scheduled event to time $T + 2 * \text{sampling_rate_unit}$.
- ◆ Therefore, such an activity is running only when the system decides to advance time. If your system is doing a series of zero-time steps, the continuous diagram waits during these steps, and does not run.
- ◆ This sampling rate is not the integration step size, which is a parameter internal to the continuous diagram. In a Sample Example, for instance, Statemate samples each second, but the integration step size is 0.01 second.
- ◆ The sampling rate may affect both performance and accuracy: the smaller the sampling gap, the higher the frequency of data exchange between the continuous diagram and the rest of the model. Thus, reducing the gap decreases performance but increases the precision of the sampled values.
- ◆ :The integration step size set inside VisSim may affect system stability and behavior.
- ◆ When running in synchronous mode (as opposed to asynchronous), the step size is used also as sampling rate. This is true for both simulation and code generation. Running a synchronous simulation is not recommended.

Continuous Activity Behavior

The behavior of a continuous block as it is started, stopped, suspended and resumed is defined as follows:

- ◆ When started, a continuous activity is active until stopped or suspended.
- ◆ When stopped, a continuous activity halts its execution and initializes any internal data to the starting point position.
- ◆ When suspended, a continuous activity halts its execution and maintains any internal data until resumed. Both the outputs and the internal state maintain their values.
- ◆ When resumed, a continuous activity is active (using data maintained while suspended) until stopped or suspended again.

Index

A

accuracy 30
Activity Behavior, Continuous 30
activity chart 6, 15, 16, 17, 18, 22, 23

B

Basics of Execution 30
Behavior, Continuous Activity 30
bind formal parameters 22
binding 4, 7, 8, 10, 22
block, compound 7, 18, 19
Bottom Up Workflow, Top Down or 6

C

Changing the VisSim Location 3
chart, activity 6, 15, 16, 17, 18, 22, 23
Check Model 10, 20
Client/Server Installation 3
Code Generation, Simulation and 7
code, external 3, 7, 29
compound block 7, 18, 19
Concepts, Key 29
Continuous Activity Behavior 30
Continuous Diagram, Create the 17
Continuous Diagrams, Printing 10
Create the Continuous Diagram 17

D

Databank, Workarea and 10
Dataport Queries, New 13
Debugging Tools 9
DGL Queries, New Documentor 12
Diagram, Create the Continuous 17
Diagrams, Printing Continuous 10
Discrete Logic System, Pure 2
Document Portability, Output 12
Documentor 11
Documentor DGL Queries, New 12
DOORS 14

DOORS/RTM Interface 14
Down or Bottom Up Workflow, Top 6

E

Execution, Basics of 30
external code 3, 7, 29
external files 10

F

files, external 10
formal parameters, bind 22

G

Generation, Simulation and Code 7
Graphical Representation 5

H

Hybrid Modeling 4
Hybrid System Type A
 Discrete Logic System Interacting with a Physical
 Environment 2
Hybrid System Type B
 Discrete Logic System Containing Some Dynamics
 and Reacting to a Physical Environment 2
Hybrid Systems 1

I

Implementation Overview 3
Installation, Client/Server 3
Installation, Windows 3
Installation, Windows NT 3
integrator 19
Interface, DOORS/RTM 14

K

Key Concepts 29

L

Licensing 3
Limitations 9
Location, Changing the VisSim 3
Logic System, Pure Discrete 2

M

metafile, Windows 12
Model, Check 10, 20
Modeling, Hybrid 4
MODULE, RT 14

N

New Dataport Queries 13
New Documentor DGL Queries 12

O

or Bottom Up Workflow, Top Down 6
Output Document Portability 12
Overview, Implementation 3

P

panel 2, 15, 16, 25, 26, 27
parameters, bind formal 22
performance 30
placeholder subroutine 14
port 4, 6, 7, 8, 10
Portability, Output Document 12
Postscript 10, 11
Printing Continuous Diagrams 10
Project, Set Up the 16
properties 4, 7, 14, 22
Pure Discrete Logic System 2

Q

Queries, New Dataport 13
Queries, New Documentor DGL 12

R

rate, sampling 8, 9, 29, 30
Reports Tool 13
Representation, Graphical 5
RT MODULE 14

RTF 11, 12
RTM 14
Run the Simulation 24

S

sampling rate 8, 9, 29, 30
sampling system 29
Session Status 13
Set Up the Project 16
Settings, Time 8
Simulation and Code Generation 7
Simulation, Run the 24
size, step 8, 9, 30
STAMPS, TIME 9
Status, Session 13
step size 8, 9, 30
subroutine, placeholder 14
System, Pure Discrete Logic 2
system, sampling 29
Systems, Hybrid 1

T

Time Settings 8
TIME STAMPS 9
Tool, Reports 13
Tools, Debugging 9
Top Down or Bottom Up Workflow 6

U

Up the Project, Set 16
Up Workflow, Top Down or Bottom 6

V

VisSim Location, Changing the 3
VSM_BLOCKS_PLOT 11
VSM_NONGENERIC_PORT_NAMES 6, 7

W

waveform 9
Windows metafile 12
Windows NT Installation 3
Workarea and Databank 10
Workflow, Top Down or Bottom Up 6