

State Machine Simulation

All material (c) Sparx Systems 2013

<http://www.sparxsystems.com>

Table of Contents

INTRODUCTION.....	3
WHAT YOU WILL NEED.....	3
WHAT YOU WILL LEARN.....	4
PREPARING A MODEL FOR SIMULATION	4
<i>Simulation using JavaScript.....</i>	<i>4</i>
<i>Initializing variables.....</i>	<i>4</i>
<i>Placement of scripting</i>	<i>6</i>
<i>Parallel flows.....</i>	<i>7</i>
<i>Summary.....</i>	<i>8</i>
RUNNING A SIMULATION.....	8
<i>Stepping through a simulation.....</i>	<i>9</i>
<i>Execution speed.....</i>	<i>10</i>
<i>View Local Variables and Call Stack.....</i>	<i>10</i>
<i>Breakpoints</i>	<i>10</i>
<i>Using Triggers.....</i>	<i>11</i>
<i>Summary.....</i>	<i>12</i>
INTERACT VIA A USER INTERFACE.....	12
<i>Summary.....</i>	<i>14</i>
CONCLUSION.....	14

Introduction

State Machine Simulation can be used for verifying the broad behavior of a system being modeled whether this be an application design or a large scale business process. When working on an application design, you can use simulation to identify any shortfalls in the early design stages which, if not resolved then, can be costly to resolve after implementation. In modeling business processes, simulation can be used to validate process changes, help mitigate and prevent failure, reduce risk, and calculate costing for use in cost reduction planning.

This paper discusses several core features of Enterprise Architect's modeling simulation for State Machines including:

- Scripting simulation behavior using JavaScript
- Simulating parallel flows
- Using Triggers to simulate user interaction in a flow
- Using user interface views to interact with a simulation

This paper makes reference to the example State Machine shown below. The model simulates a Digital Clock and its user interface, providing examples of the core simulation features.

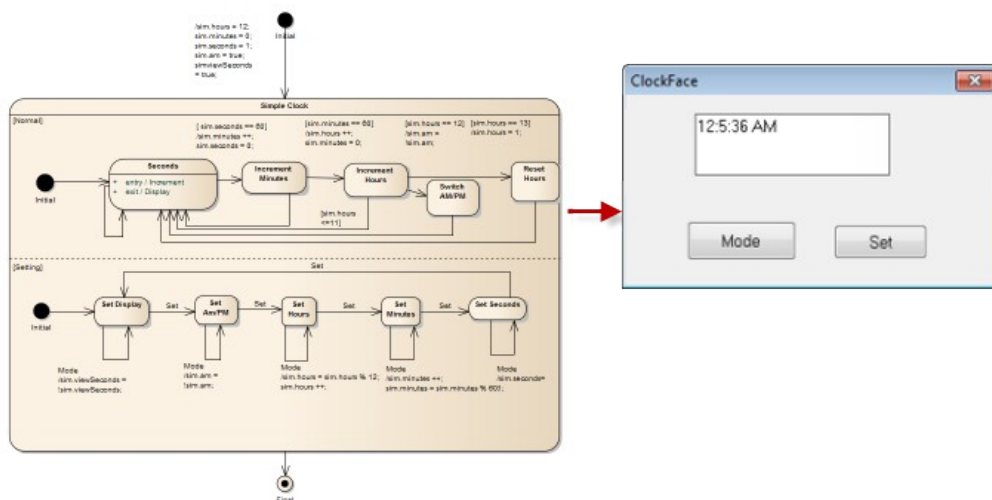


Figure 1: State Machine and User Interface diagram discussed in this paper

What you will need

For State Machine simulation you will need the Corporate edition or above of Enterprise Architect version 10 or later.

The example model referenced in this paper can be downloaded from:

<http://community.sparxsystems.com/white-papers>

What you will learn

In discussing a number of Enterprise Architect simulation capabilities you will learn more about:

- Setting up an initialization script
- Setting up script in the State Machine within:
 - Transitions: Guards & Effects or
 - State: Operations
- Using Triggers for altering flow
- Running parallel flows using a concurrent State Machine
- Viewing variables and the call stack
- Using Win32 screens for visual interaction involving:
 - Displaying simulation generated information
 - Instigating triggers using Win32 buttons

Preparing a model for simulation

In this section you will be introduced to several additions to a standard State Machine model that provide support for simulation. This section includes direction on how and where to add JavaScript statements to the model that will be interpreted during the simulation, and setting the model to simulate a parallel flow of execution.

Simulation using JavaScript

Enterprise Architect uses simple JavaScript statements to evaluate conditions and set the behavior. This involves defining variables prefixed with either *sim* or *this*.

The simplest definition uses the *sim* prefix. For example:

- `sim.minutes`

Variables with a *sim* prefix are set as a global variable that can be used across the current State Machine and its child elements. Any States in diagrams nested under your initial State Machine will have access to the global value.

Where you want to reference attributes of the parent Class you can use the *this* prefix:

- `this.minutes`

The *this* prefix applies globally to all behavioral diagrams nested under the Class. The prefix is not discussed in this example; for further details see the: [Dynamic Simulation with JavaScript](#) Help topic

Initializing variables

To run a simulation you need to initialize variables on start up. Enterprise Architect simulation offers two methods for starting a simulation, which provide different means for initializing variables:

1. Set the variables on the first connector or element in the model (preferred)
2. Use an Execution Analyzer script to initialize variables

1) With State Machines you can set variables in the first Transition after the initial element. As an example, Figure 4 shows the Properties dialog for the Transition connector containing an initializing script for the Digital Clock model.

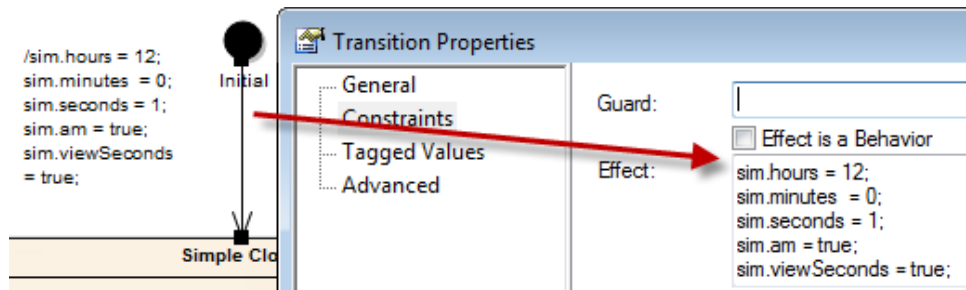


Figure 2: Initialization of variables in the first Transition

To run a simulation that initializes in the Transition, right click on the diagram background and select: **Execute Simulation | Interpreted**.

For State Machines, this is the simpler method as it allows a run to be initiated from the diagram.

2) When using an Execution Analyzer script you can initialize variables in the **Input** field of the Simulation Script:

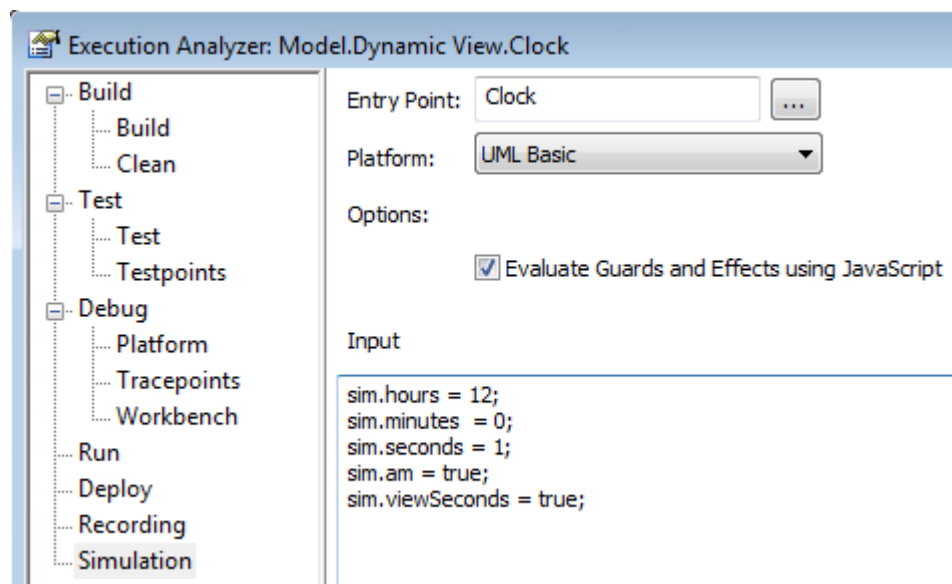


Figure 3: Initializing variables in the Execution Analyzer script

For more information on using the Execution Analyzer script see the [Set up Simulation Script](#) Help topic.

You can execute this simulation script from the diagram context menu: **Execute Simulation | Using Script | <Script Name>** or execute it from the Execution Analyzer view.

Placement of scripting

With State Machines there are a number of key points where JavaScript code can be entered and later interpreted during the simulation process. These points include:

- Transitions (Guards & Effects)
- State Operations (Behavior)

State Transitions have three points that affect the flow. These include:

- **Transition Guard**

The script for a Guard is simply a Boolean conditional statement. A **true** result allows a transition to be processed. Figure 6 is an example of the properties of a Transition from the Seconds State to the Minutes State:

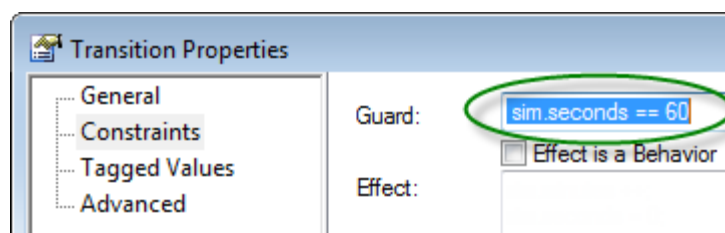


Figure 4: A conditional statement in the Transition Guard

In this case the Transition is testing if the variable **sim.minutes** is equal to **60**. If so, the simulation will flow on to the destination State called **Minutes**.

- **Transition Effects**

A script defined in an Effect will be executed when the Guard meets the Boolean condition. In the case above, where the count of seconds equals **60**, the Effect will be executed to increment the minutes.

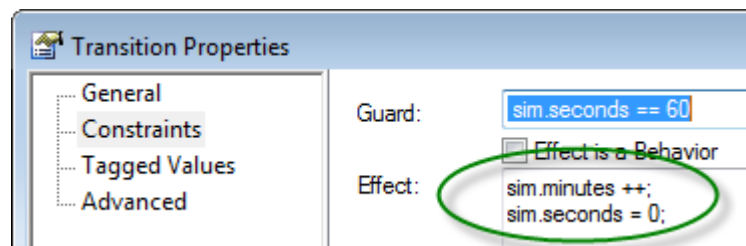


Figure 5: Using a script in the Effect

- **State Operations**

With State Operations the simulation script is defined in the Operation's Behavior. For any State you can define a script for *entry*, *exit* and *do* Actions.

In the Digital Clock example there are *entry* and *exit* Operations set for the **Seconds** state. Figure 8 shows the *entry* Operation along with the script in its Behavior section. This script performs a simple increment of the **Seconds** variable.

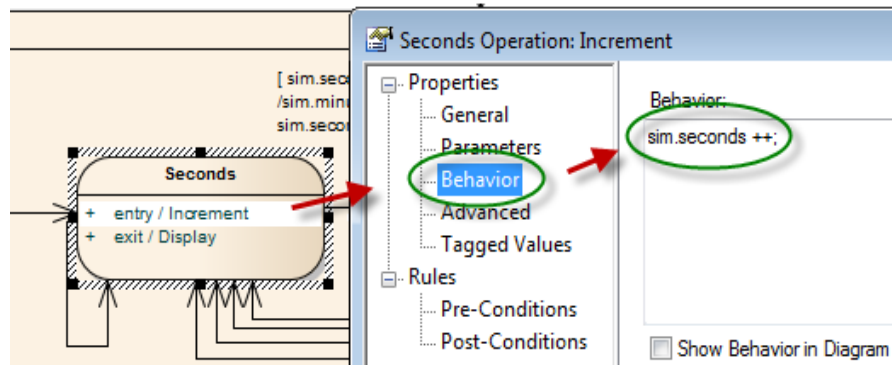


Figure 6: An example of an Operations script set in a State

The exit Operation called **Display** outputs the current time to the Screen UI. This will be covered in more detail below.

To access the behavior in the diagram, simply select the Operation and double-click on it, then select **Behavior** in the Properties dialog.

Parallel flows

Parallel flows of execution can be depicted and simulated in both State Machine models and Activity models. In an Activity model you simply split the flow into two streams, whereas with State Machines you define a State Machine with **concurrent substates**.

In the Digital Clock example there are two streams running in parallel:

- One for counting and displaying the time
- One for setting options for the time and the display mode

These parallel flows are defined by creating two concurrent substates in the State Machine, named **Time Display** and **Setting** respectively, as shown in Figure 7.

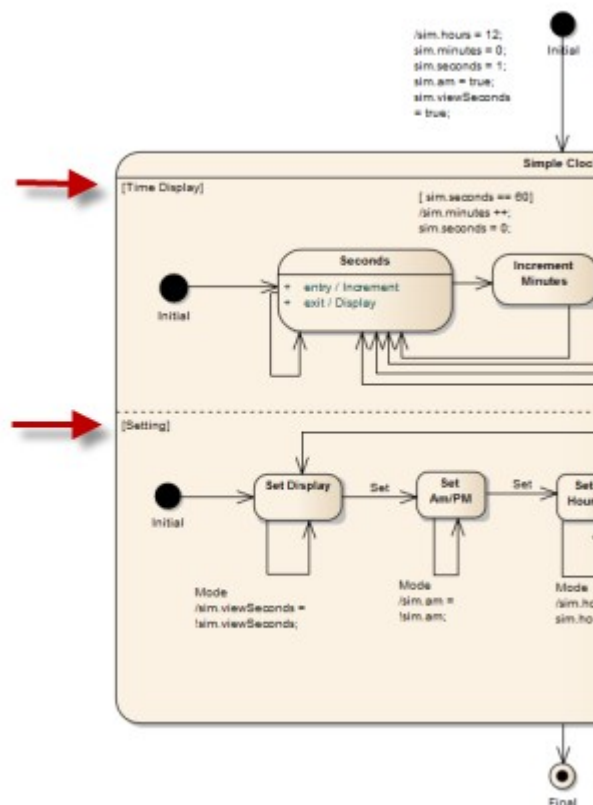


Figure 7: Using Concurrent States for parallel execution

To set Concurrency in a State, right-click on the State in the diagram and select the context menu option: **Advanced | Define Concurrent State Machines**.

For more details on this see the [Multi-Threading Concurrent State Machines](#) Help topic.

Summary

Using the Digital Clock example we have discussed several core simulation options:

- Initializing JavaScript variables used in the model
- Using State Operations for defining a script – here the initial seconds count is performed using a State Operation in the **Seconds** State.
- Using a script in a Trigger Guard and its Effect - a Trigger for incrementing minutes is set in a Transition out of the **Seconds** State.
- Setting up the simulation of a parallel flow in a State Machine.

Running a simulation

We recommend that prior to running a simulation you open the Simulation window in Enterprise Architect (main menu: **Analyzer | Simulation**).

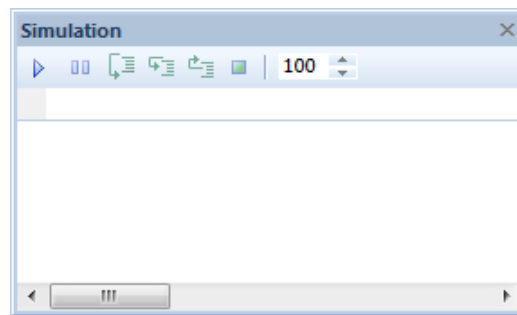


Figure 8: The Simulation window - used for executing a simulation

State Machines include Triggers defined in the Transition Connectors. These triggers determine the Transition flow. As you will be using these Triggers, you should also open the **Simulation Events** window (main menu: **Analyzer | Simulation Events**).

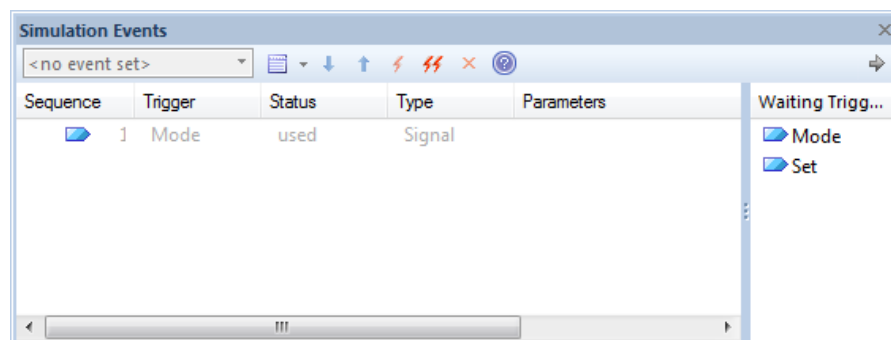



Figure 9: Accessing Triggers in the Simulation Events window

Open the example model supplied with the paper and follow the instructions on the default diagram.

Stepping through a simulation

When running a simulation you can perform standard debug actions such as Pause/Resume, Step Over, Step In, Step out.

Execution can be run from:

- The Simulation window toolbar using the icon  or
- The Diagram context menu option: **Execute | Simulation | Interpreted**

Using the context menu option, you can run the simulation in Interpreted mode or Manual mode. For most of this discussion we will use the Interpreted mode to evaluate JavaScript and any trigger signals. However, it is still necessary to display the Simulation window to start and stop the simulation run, as well as to view executed steps:

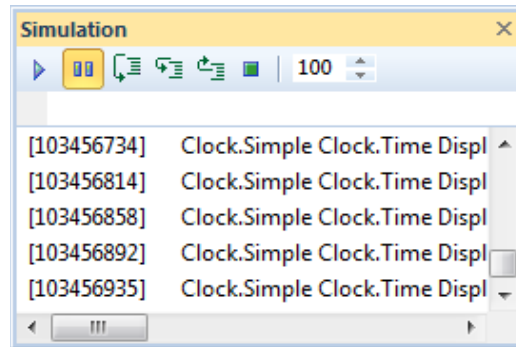


Figure 10: Simulation View with the Run, Pause and Stop options

Execution speed

When running a simulation you might want to adjust the speed of the simulation. The speed can be varied between 1 and 100, with the default set to 50. For the Digital Clock example it is worthwhile using a high value for a more realistic view of operation. For more details see the [Run Model Simulation](#) Help topic

View Local Variables and Call Stack

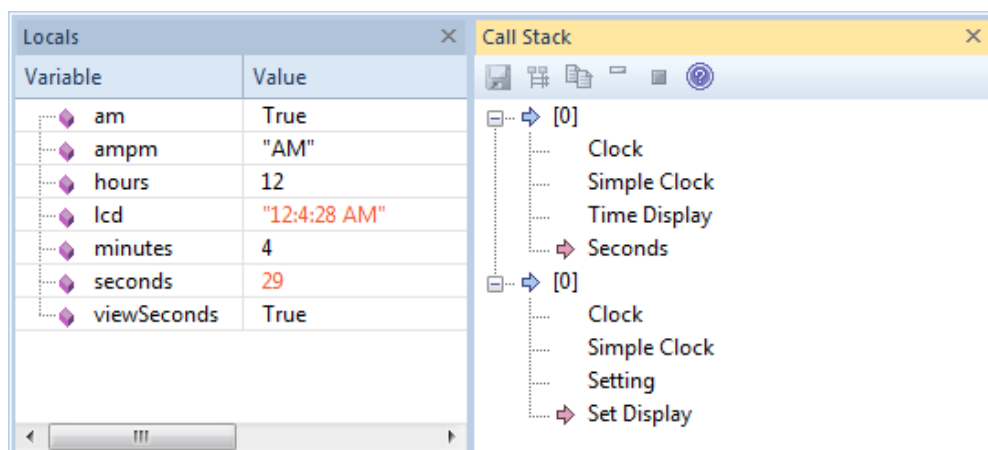


Figure 11: Locals and Call Stack windows for the Digital Clock simulation

Integral to simulation debugging is the ability to view, while executing, the values of variables and the state of the the Call Stack. These are viewable using the Locals window and the Call Stack window. The Call Stack window can be useful in checking multi-threaded parallel simulation runs. Figure 11 above shows the Locals window and the Call Stack window for the Digital Clock simulation. Note that, as this simulation is running two parallel streams (the **Time Display** and the **Settings** streams), the Call Stack is reflecting this by showing two threads.

Breakpoints

You can debug a simulation using breakpoints. These are set by dragging the element to be used as a breakpoint, from the Project Browser, on to the Breakpoints window. For more information see the [Simulation Breakpoints](#) Help topic.

Using Triggers

A Transition can be triggered from an external source. This trigger will instigate the Effects of the transition, as well as progress the flow on to the destination State.

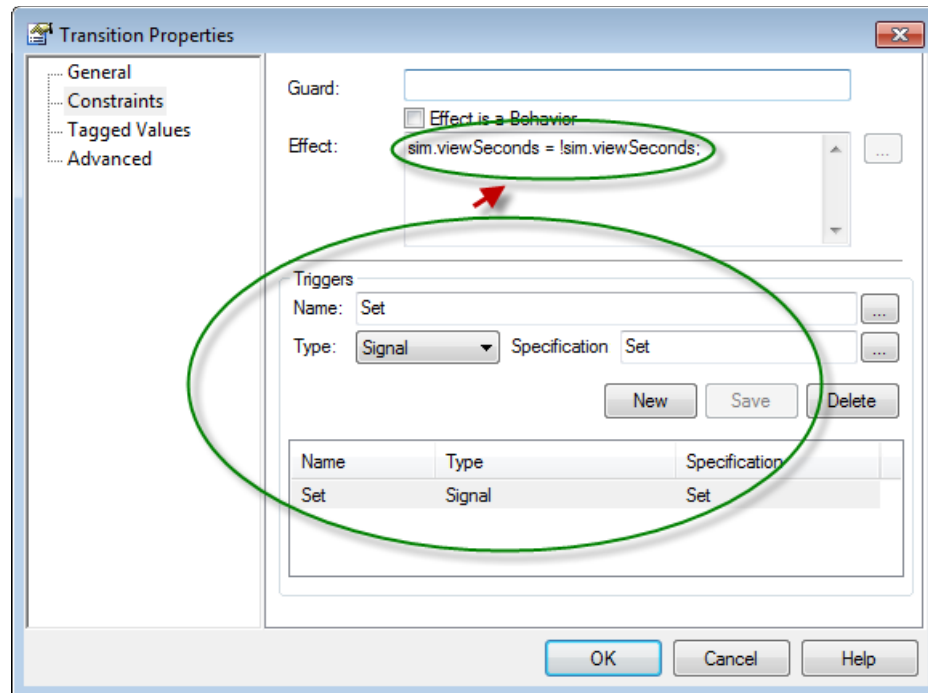


Figure 12: Setting a Transition Trigger to trigger code in the Effect

For example, the Digital Clock has several Transitions that are configured to be triggered when the **Set** Signal is user activated. The first of these 'Set' events is to change the display mode to show or not show the seconds on the display (toggle the display between HH:MM and HH:MM:SS).

When running the simulation these Triggers can be initiated from the Simulation Events window. To open this, select the main menu: **Analyzer | Simulation Events** option.

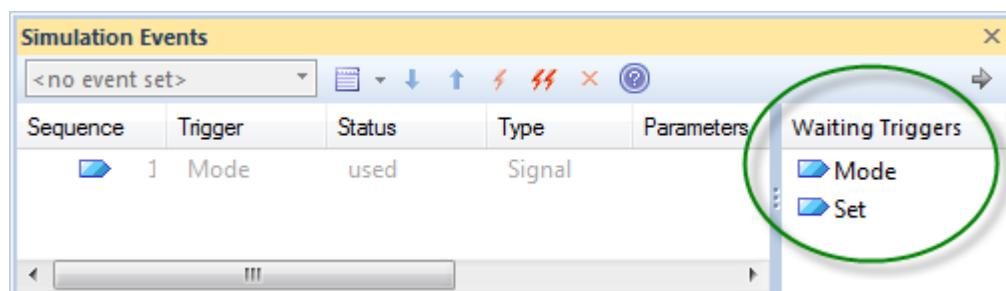


Figure 13: Simulation Events Window showing the Digital Clock Triggers

With the simulation running, when a Transition with a trigger is encountered, this will be displayed in the Waiting Triggers column. Double-clicking on the trigger will initiate that trigger in the model.

For more details on setting up and applying triggers and signals see the [Triggers](#) Help topic.

In the next section, we will discuss the means of setting a Button element on a User Interface diagram as the instigator of a trigger.

Summary

What we have discussed in terms of running a simulation includes:

- How to run or step through a simulation
- Setting the Execution speed for the simulation
- Viewing the variables and call stack
- Setting break-points for debugging a simulation
- Viewing and selecting a trigger when running a simulation

Interact via a User Interface

Using a Win32 screen model, you can include in your simulation an interactive screen simulation. On Win32 screens you can use the following element types to interact with the simulation:

- UI buttons for triggering events
- UI edit control to display the text and variables
- UI fields to support user input to variables

Using the Digital Clock example we discuss the first two of these: two buttons for setting the clock, and an edit-control to simulate an LCD display for viewing the time.

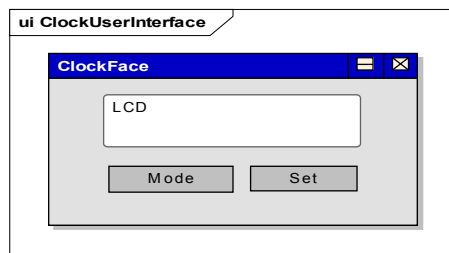


Figure 14: User Interface for the Digital Clock example

To establish a user interface for your simulation you can create a Win32 UI screen in the package in which the State Machine diagram is located. You can then add buttons and fields for triggers and input parameters.

Firstly, you create a Win32 UI diagram containing a Screen element (a Dialog). For more details see the [Win32 UI](#) Help topic.

You then create a UI button that signals a Trigger; that is, a Win32 UI element of type Button containing a Tagged Value named **OnClick**. You then set the value of the tag with the following format:

```
BroadcastSignal("NameOfSignal")
```

For the Digital Clock example the **Set** button has an **OnClick** Tagged Value of `BroadcastSignal("Set")`:

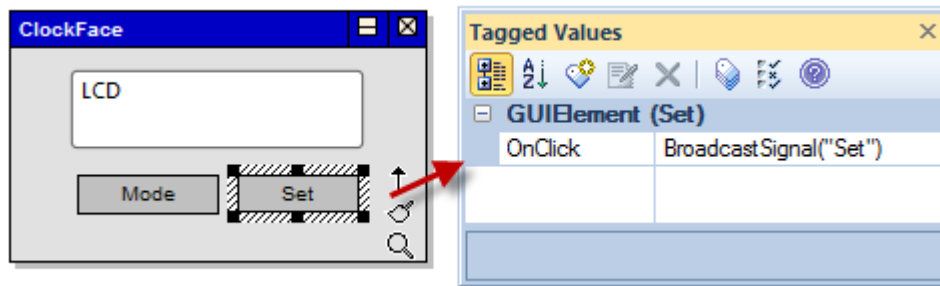


Figure 15: Setting a Signal Trigger from a UI Button

This references the **Set** Trigger shown in Figure 13.

Using a UI field you can output text to a Win32 Edit Control (a GUIEdit) element. In the Digital Clock example one of these is placed on the **ClockFace** dialog named **LCD** (see Figure 15).

Set the UI

A key point when initializing your script is to state which Win32 UI model you are using. In the example this is set in the entry connector (see Figure 4). The script for this is:

```
dialog.ClockFace.show=true;
```

Where **Clockface** is the name of the Screen element.

The ability to enable or disable these in the scripts gives you the option to use multiple screens in your simulation.

To interact with the UI element from the Simulation model you set the text of a UI field from your script. In the Digital Clock example, the **LCD** field is updated by script in the **Seconds** State element using a reference based on `dialog.Screen.Element.Text` illustrated in Figure 16.

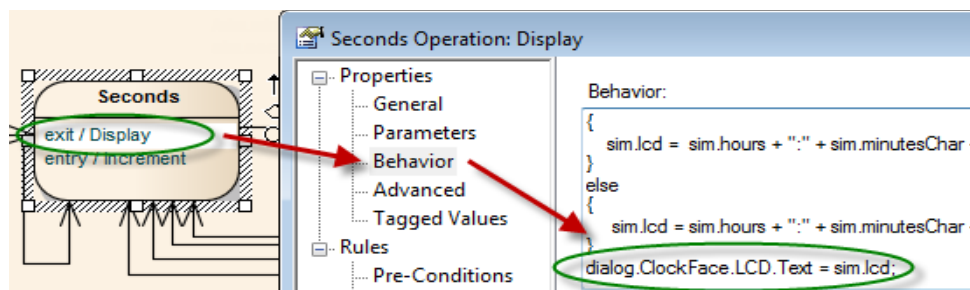


Figure 16: Example of updating UI field from script

With the script:

```
dialog.ClockFace.LCD.Text = sim.lcd;
```

the **LCD** field is updated in the Win32 diagram:

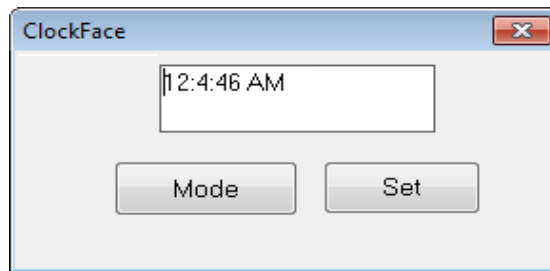


Figure 17: Clock Face with field updated while running

So to cover the user-functionality that is simulated with these two buttons:

- The **Mode** button sets the mode in which the **Set** button will be used to alter the time.
- The **Set** button is used for setting AM/PM, hours, and minutes, and for zeroing seconds

Summary

The points we have discussed for interfacing a simulation with a UI screen include:

- Setting up a Win32 diagram
- Creating buttons and text boxes
- Linking the buttons to Triggers
- Outputting text to a text-box
- Opening a Win32 dialog from a simulation

Conclusion

This white paper has discussed aspects of creating a State Machine that interacts with a user interface simulating a typical digital clock.

Using this example we reviewed several features available in Enterprise Architect's simulation – using JavaScript in the model, simulating a Parallel flow, using Triggers for user interaction of the flow and interacting with a User Interface using the Win32 views.

Having reviewed these simulation options you can now apply them in modeling an application or business process. For application modeling you can use these options for simple verification of design modeling, to validate a process and improve the UI design and general usability. For business processes, these options can be used to validate a process, simulate cost calculations based on varying parameters in the model, and mitigate against risk or failure determined by the simulation. So from application development through to business modeling, simulation can be a powerful tool for you to optimize any system being modeled.