



Using Altia Design with I-Logix Rhapsody



Connection Tutorial

1.0 Introduction

Altia Design simulation graphics and user interface software can be used in conjunction with I-Logix's Rhapsody. The Altia interface to Rhapsody allows users to create interactive graphical panels with the state-of-the-art Altia Design editor.

The Altia Design package includes an editor, runtime engine and numerous libraries of components for quickly creating user interfaces to Rhapsody simulations. In addition to using the supplied components to create graphical front ends, the Altia Design product allows users to make modified versions of existing components and create their own components in the editor without programming. The benefits of such graphical front panels include involving the customer in the design process, increasing collaborative design and debugging models more efficiently.

This step-by-step tutorial will guide you through the process of creating an Altia Design interface and then building a simple Rhapsody model that communicates with it.

2.0 Tutorial Overview

Before you begin this hour-long tutorial, make sure you have *Rhapsody* and *Altia Design 8.0* or later installed on your machine. If you need a new copy of Altia Design, please contact Altia, Inc. at 719-598-4299 (US) or via the web at www.altia.com.

- The first section, titled [Use Altia Design to Create a Simple GUI](#), will step through the procedures required to build a virtual front panel to drive and monitor our Rhapsody model.
- The process of creating the Rhapsody model will be addressed in the section titled [Build the Rhapsody Model](#).
- Finally, the [Test Your Altia/Rhapsody Interaction](#) section will show you how to verify that your GUI and the Rhapsody model are connected correctly.

3.0 Use Altia Design to Create a Simple GUI

1. Open Altia Design by choosing its icon from the **Altia Design** program group.
2. From the **File** menu, choose **Open Model Library...**

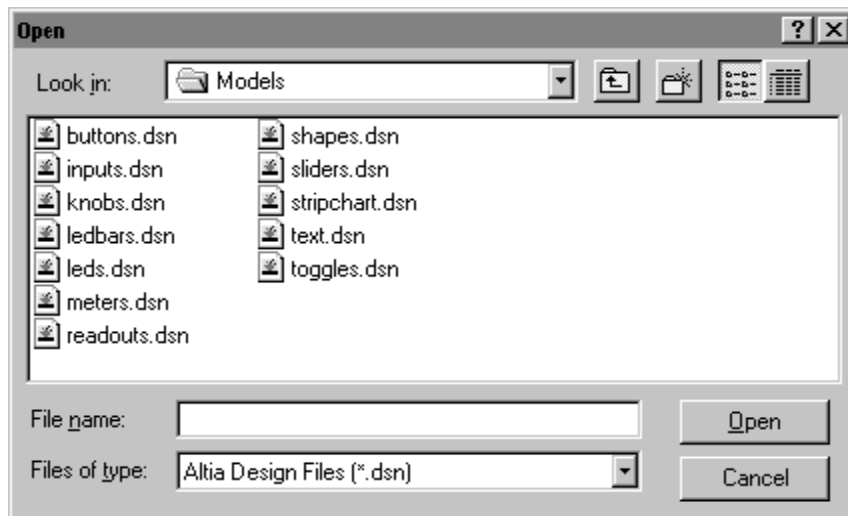


FIGURE 3-1 Open Libraries Dialog

3. Choose the **toggles.dsn** library and then click **Open** to view a collection of pre-built toggle buttons. Pull the grey, two-position toggle (or whichever toggle appeals to you most) into your design by left-clicking on it and dragging it into the Design window. Close the Models View window.

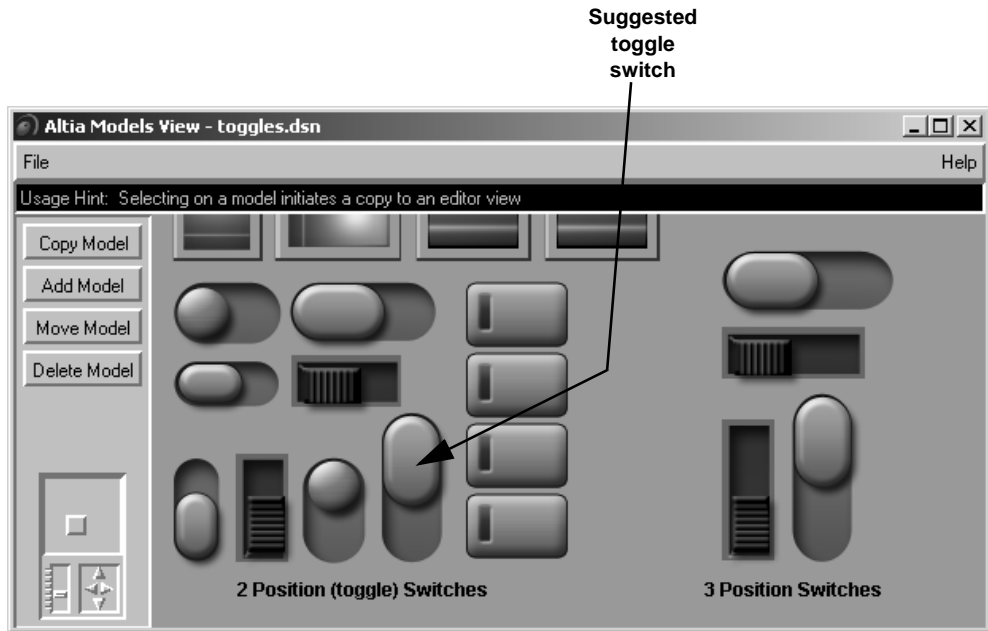


FIGURE 3-2 Library of Toggle Switches

The Rhapsody model that we will build in the next section will be a simple gain loop that multiplies an incoming value by a constant and then outputs the result. This toggle will control whether this virtual machine is on or off.

4. Because we know what this component will do in our Rhapsody model, we can modify its properties to identify it. Double click on the toggle button component in Altia to open its Property Dialog (or select it and choose **Properties...** from the **Object** menu).
5. Change the **Label** property from **Toggle** (or whatever yours says) to **Power**.

- Next, change the **Show Label** property drop-down box to **Yes** (Figure 3-3).



FIGURE 3-3 Toggle Switch Property Dialog

There are several other properties associated with this toggle button. We won't be altering any of the other properties for this tutorial, but feel free to experiment with them. Properties make changing the behavior and appearance of the component extremely easy.

- When you are done, your button will look similar to the one in Figure 3-4. If you want to replace your button, just select it, press the **Delete** key and drag in a new one from the library.

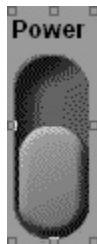


FIGURE 3-4 Finished Toggle Switch

8. In order to show if our virtual machine is on, let's add an LED indicator. Open the LED library by choosing **Open Model Library...** from the **File** menu.
9. Select the **leds.dsn** file and then click the **Open** button. Drag and drop one of the LED indicators into the Altia Design window and then close the Models View.

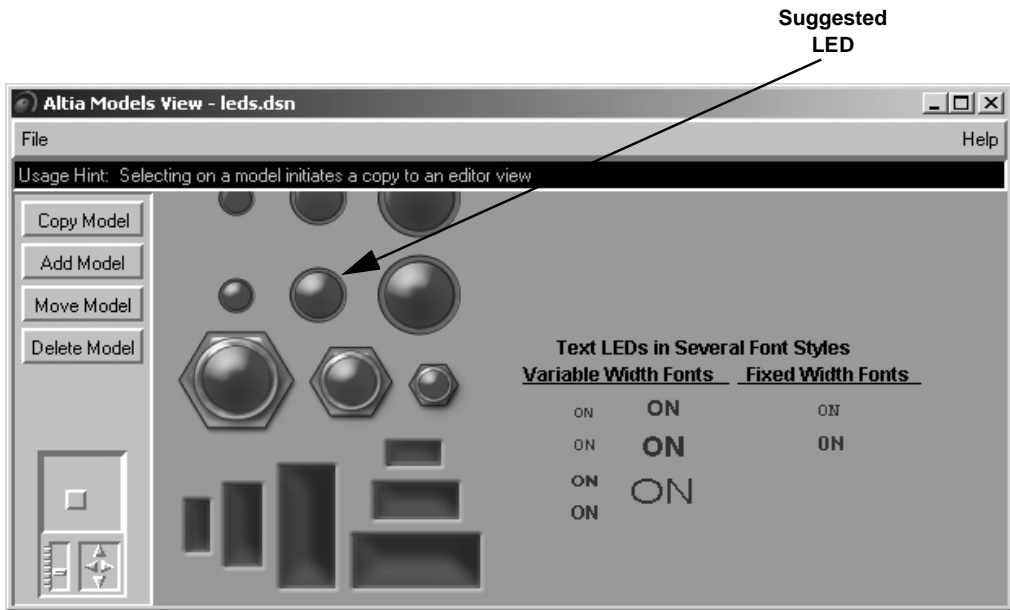


FIGURE 3-5 Library of LEDs

10. Next, let's add an object whose function is to set the value to be multiplied. A natural choice for this component would be a slider. To open the library of sliders, click on the **File** menu and choose **Open Model Library...**

11. Choose the **sliders.dsn** library and then click **Open** to view the available pre-built sliders. Drag and drop a horizontal slider into the Altia Design window and then close the Models View.

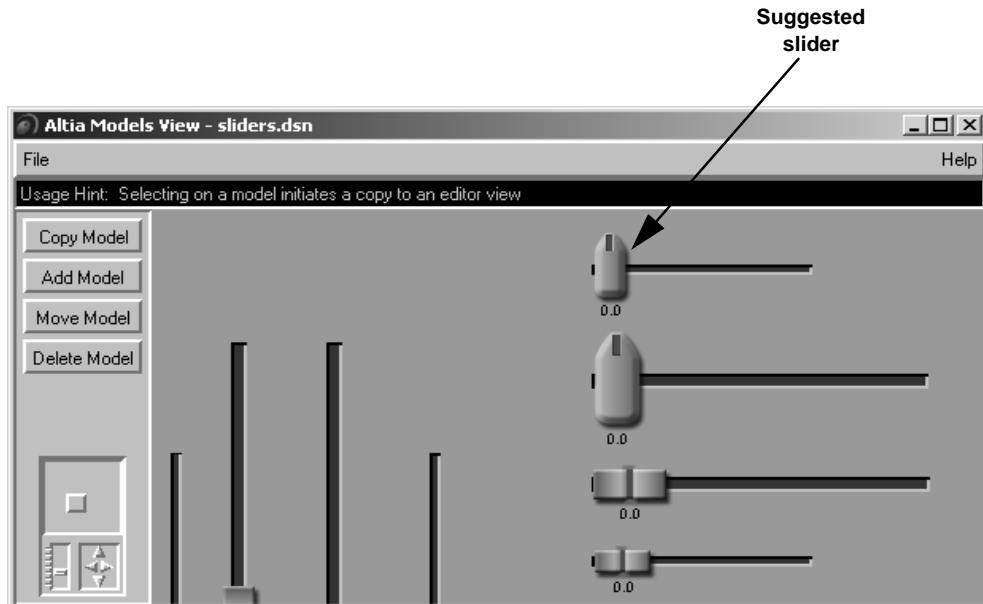


FIGURE 3-6 Library of Sliders

In Design, the **Property Dialog** should still be open and will show the properties of the slider. If you need to reopen the Property Dialog, just double click on the slider.

Change the slider's **Label** and **Show Label** properties to **Gain Feed** and **Yes**, respectively.

The finished slider should look something like the one shown in [Figure 3-7](#).



FIGURE 3-7 Finished Slider

12. Let's use a gauge to display the models output. Open the `meters.dsn` library and drag in a needle gauge.

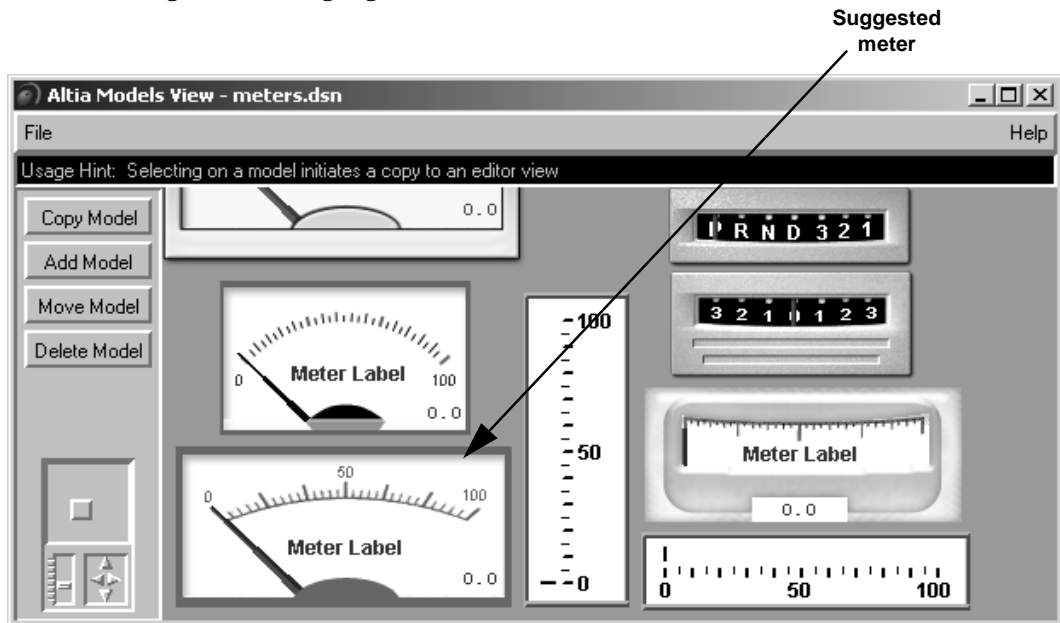


FIGURE 3-8 Library of Meters

- For the meter, change the **Text Label** property from **Meter Label** to **Gain Result**. Your final design should have all of the components shown in [Figure 3-9](#).

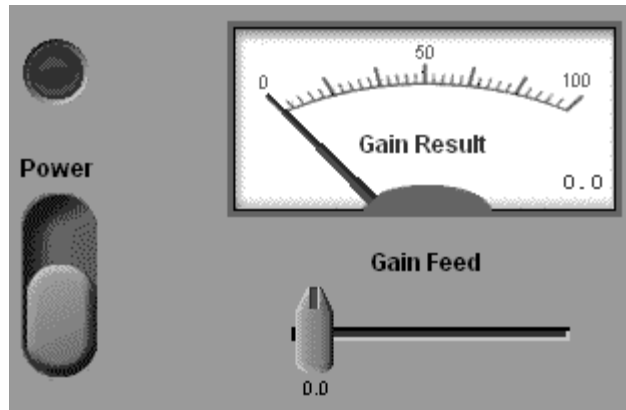


FIGURE 3-9 Final Layout

- Now, close the Altia Libraries window and Property Dialog.
- We are not done with our design just yet, but let's save before moving on to the next section. In the Altia Design editor, choose **Save** from the **File** menu.
- Change to a directory where you can put your Rhapsody project (the images in this tutorial will use `c:\work\tutorial`). Then, in the File name box, type **tutorial.dsn**. Press the **Save** button. We have now saved this simple design to the file **tutorial.dsn**.

4.0 Creating External Connections For the GUI

- The process of sending data from our Rhapsody model to our Altia design is simplified by the use of external signals (or connectors). These connectors offer easily accessible entry points for external programs. Let's add a few connectors to our sample design.
- From the Altia Design **Connections** menu, choose **External Signals** to open the external signals dialog. Initially, there will be no external signals in the dialog.
- From the **Edit** menu of the external signals dialog, choose the **Add Connection...** option. This will open a blank **Edit Connection** dialog.

- In the **I/O Name** field, type `slider`. In the **Animation** field, type `slider` (as shown below) then press the **OK** button. The name in the **Animation** field (`slider`) is the case-sensitive name we will later use to connect the slider to our Rhapsody model.

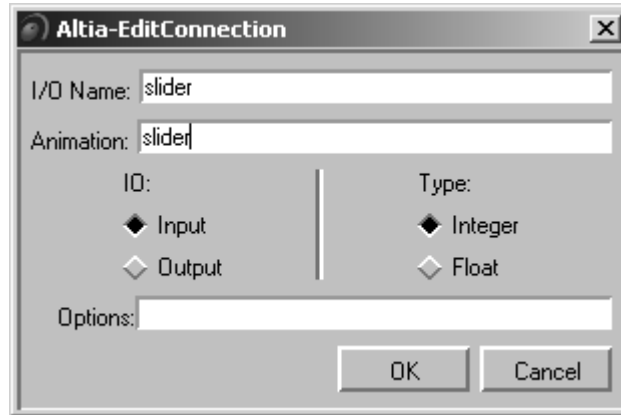


FIGURE 4-10 Slider External Connector

- Let's add another Rhapsody model input for the power button. Once again, from the **Edit** menu of the external signals dialog, choose the **Add Connection...** option. This will open another blank **Edit Connection** dialog.
- In the **I/O Name** field, type `power`. In the **Animation** field, also type `power` then press the **OK** button. The name in the **Animation** field (`power`) is the case-sensitive name we will later use to connect the power button to our Rhapsody model.
- Now let's add a Rhapsody output connector to control our meter. From the **Edit** menu of the external signals dialog, choose the **Add Connection...** item.
- In the **I/O Name** field, type `meter`. In the **Animation** field, type `meter`. Click the **Output** radio button and then press the **OK** button. The name in the **Animation**

field (**meter**) is the case-sensitive name we will later use to connect our Rhapsody model to our meter.

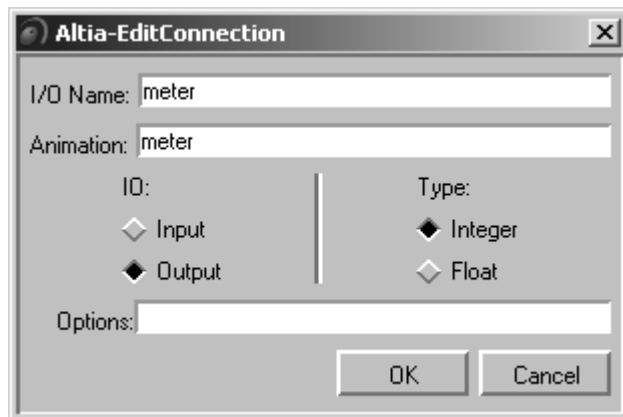


FIGURE 4-11 Meter External Connector

9. Finally, let's add an output connector to control our LED indicator. From the **Edit** menu of the external signals dialog, choose the **Add Connection...** item.
10. In the **I/O Name** field, type **power_led**. In the **Animation** field, type **power_led**. Click the **Output** radio button and then press the **OK** button. The name in the **Animation** field (**power_led**) is the case-sensitive name we will later use to connect our Rhapsody model to our LED.

At this point, we have the following entries in our external connections dialog.

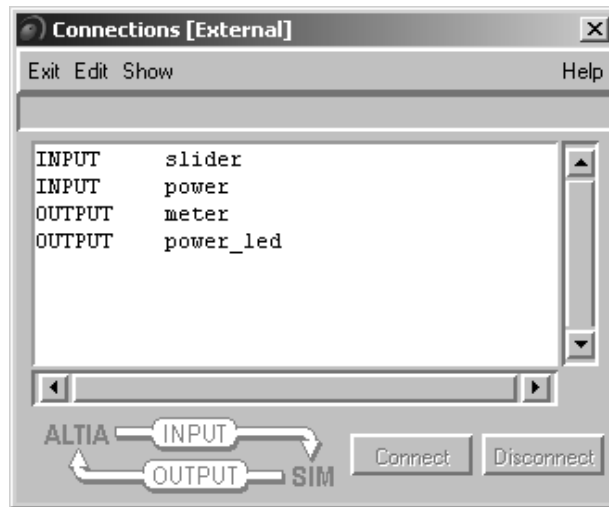


FIGURE 4-12 External Connectors

11. Now we must connect these external signals that we have built to our Altia objects. This is easily done in Altia.

12. From the Altia Design editor's **Connections** menu, choose **All Objects**. A window containing the available connections of our four objects will open.

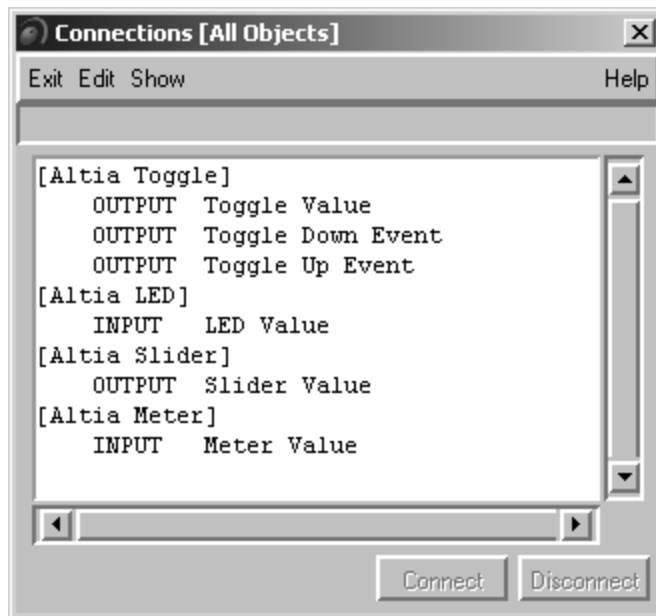


FIGURE 4-13 Connectors of All Altia Objects

13. In the connection dialog that shows all of the objects, click on the signal labeled **OUTPUT Slider Value**, then click on the signal in the external connectors dialog labeled **INPUT slider**.
14. When this is done the **Connect** button will become available. Press it to connect the slider to the external signal which will go to the Rhapsody model (which we will build in the next section). The connections dialogs should reflect the fact that these objects are now connected (see [Figure 4-14](#)).
15. To connect the power button, click on the signal labeled **INPUT power** in the external connections dialog, then click on the signal labeled **OUTPUT Toggle Value**. Press the **Connect** button to connect the toggle button.
16. To connect the analog meter, click on the signal labeled **OUTPUT meter** in the external connections dialog, then click on the signal labeled **INPUT Meter Value**. Press the **Connect** button to connect the Rhapsody output to the meter.
17. Finally, connect the **OUTPUT power_led** in the external connections dialog to the signal labeled **INPUT LED Value**.

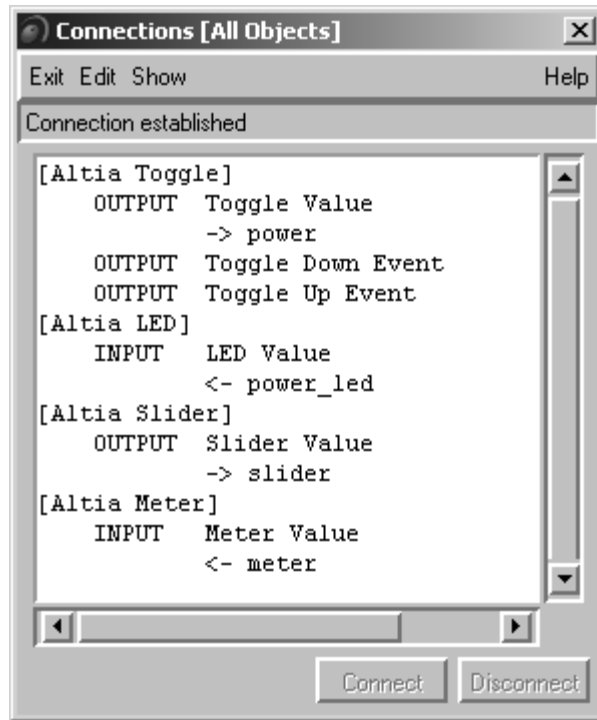


FIGURE 4-14 Connection of All Objects

18. Now, let's save our design again by choosing **Save** from the **File** menu.
19. After saving, exit Altia Design by choosing **Exit** from the **File** menu.
20. Although the graphics look neat, right now they don't do much because we haven't built the Rhapsody model to control them. This is our next task.

5.0 Build the Rhapsody Model

1. Open Rhapsody in C++.
2. From the **File** menu, choose **New** to begin a new project. In the ensuing dialog, enter the information such that the project will get created into the same directory where you saved your Altia design then press **OK**.

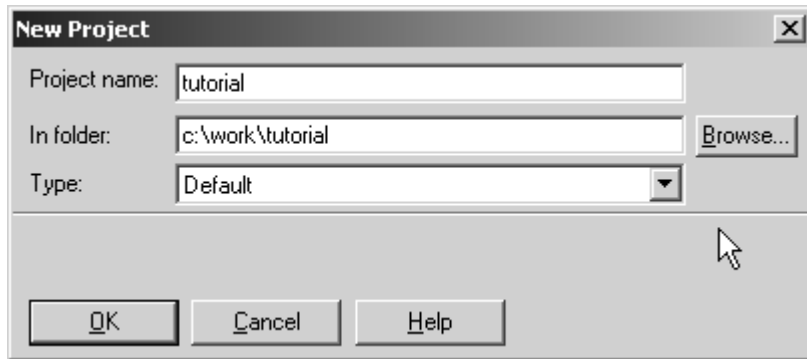


FIGURE 5-15 Create New Rhapsody Project

3. Now copy the files and sub-directory from the Altia/Rhapsody Connection package (`altiaart.exe`, `colors.ali`, `fonts.ali`, `AltiaFunctions.c`, `AltiaFunctions.h`, `AltiaPkgTemplate.rpy`, and the `AltiaPkgTemplate_rpy` sub-directory) into this same directory (e.g., `c:\work\tutorial`).

- From the Rhapsody **File** menu, choose **Add to Model...**, select the **AltiaPkgTemplate.rpy** file and then press **Open**. In the next dialog, press the **Select All** button and then click **OK** to import the component and the package.

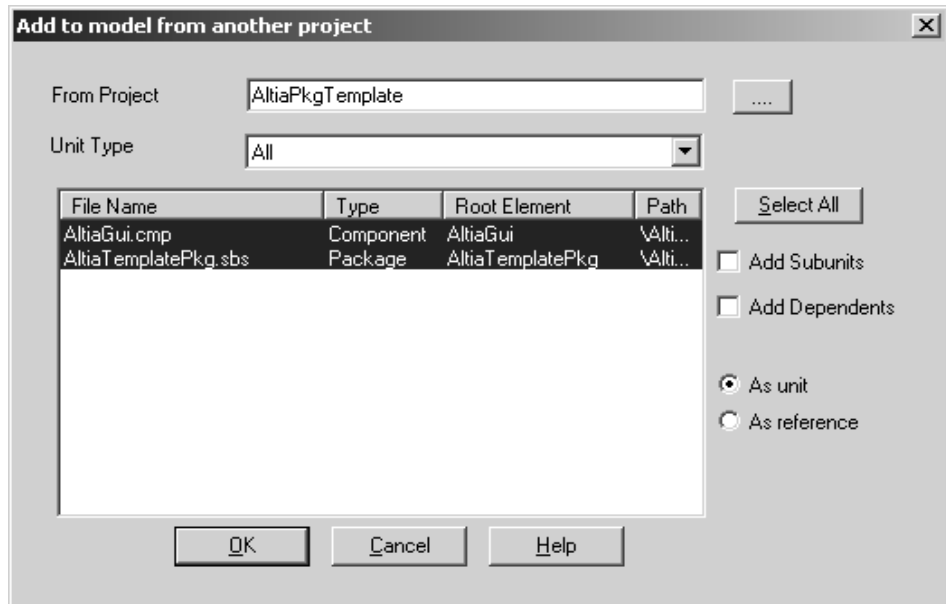
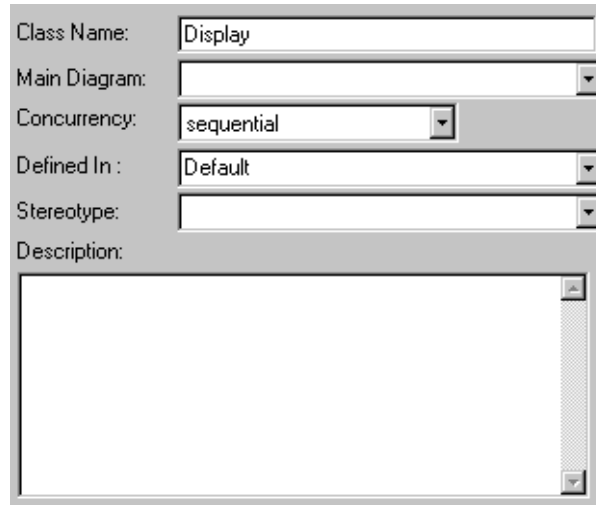


FIGURE 5-16 Import the Altia Component and Package

- All of the building blocks are now in place, so let's create a simple Rhapsody model from scratch.

6. Add a new class to the **Default** package by right-clicking on it and then choosing **Add New/Class**. Type in **Display** as the Class Name then press Enter to rename this newly created class.



The image shows a dialog box for renaming a class. It has several fields and dropdown menus:

- Class Name:** A text input field containing the word "Display".
- Main Diagram:** A dropdown menu that is currently empty.
- Concurrency:** A dropdown menu with "sequential" selected.
- Defined In:** A dropdown menu with "Default" selected.
- Stereotype:** A dropdown menu that is currently empty.
- Description:** A large, empty text area for entering a description.

FIGURE 5-17 Rename Class to Display

7. In the left-hand side of the browser, right-click on the topmost project node (which will be labelled **tutorial** if you have followed this tutorial exactly) and choose **Add New/Object Model Diagram**. Name this newly created Object Model Diagram (OMD) **Overview**.
8. Double-click on the **Overview** entry in the browser to open it (the OMD should be blank).
9. Drag the **Display** class from the browser into the top part of the Object Model Diagram. Right-click on the block in the OMD and choose **New Statechart**.
10. From the statechart toolbar on the left, select the State tool. Click and drag inside the statechart window to create a state then change the state name to **Off**

(see [Figure 5-19](#) below). Right-click on the Off state and select **Features...** In the **Action on entry** box, type `showPower (0) ;` then click OK.

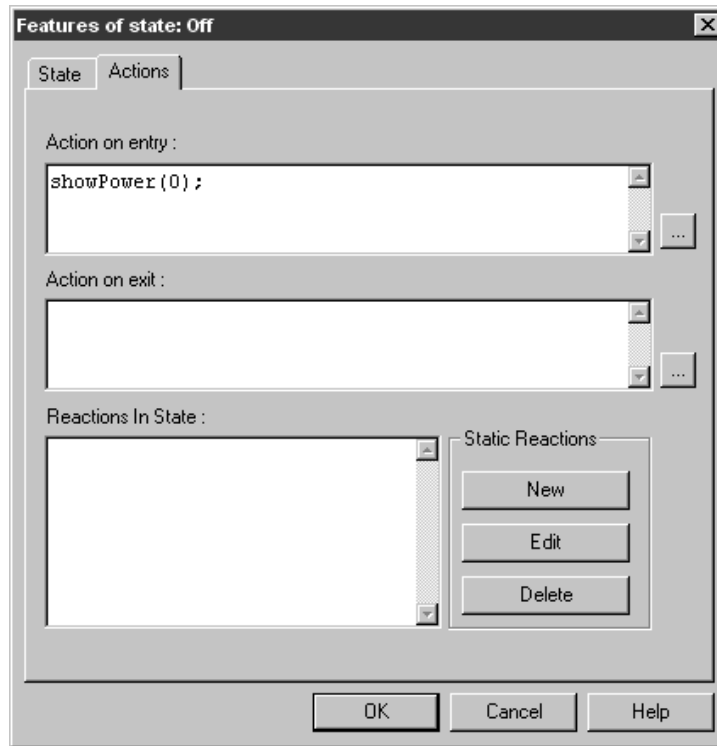


FIGURE 5-18 Off State Action On Entry

11. Create another state and change its name to **On**. In its **Action on entry** box, type `showPower (1) ;`.
12. Choose the Default Connector tool from the toolbar and draw a transition into the Off state (leave the transition name blank). Choose the Transition tool from the toolbar and draw a transition from Off to On (name it `onPowerUp`) and from On to Off (name it `onPowerDown`). See [Figure 5-19](#) to see how the finished stat-chart should look

13. Lastly, we want to add a transition from On back to itself. For its name, type `onUpdate/showValue(0.5*Value)`. The finished statechart should resemble the one shown in [Figure 5-19](#).

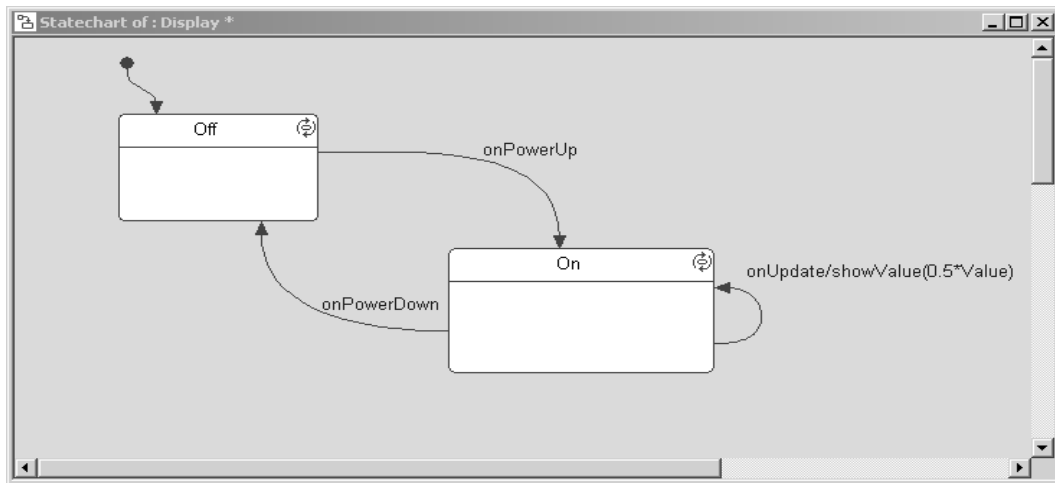


FIGURE 5-19 Finished Statechart

14. Go back to the browser, right-click on the **Display** class and choose **Add New/Attribute**. Rename the attribute `value`.

- Right-click on the **Display** class and choose **Add New/Operation**. Rename the operation to **showValue**. Double click it to open and then add an integer argument named **showInt**. Leave the body of this operation blank.



FIGURE 5-20 showValue Operation

- Again, right-click on the **Display** class and choose **Add New/Operation**. Rename the operation to **showPower** and add an integer argument named **showInt**. Leave the body of this operation blank. Click OK to close the Primitive Operation Viewer
- Our **Display** class is now fully developed. It switches states whenever it receives one of the "on" events and it calls the "show" functions whenever it switches states. Now let's drop in the code that will connect these to Altia.

18. In the browser, drag the **AltiaDisplay** class from the **AltiaTemplatePkg** package into the **Overview** OMD. Choose the Inheritance tool and draw a line from the **AltiaDisplay** class to the **Display** class.

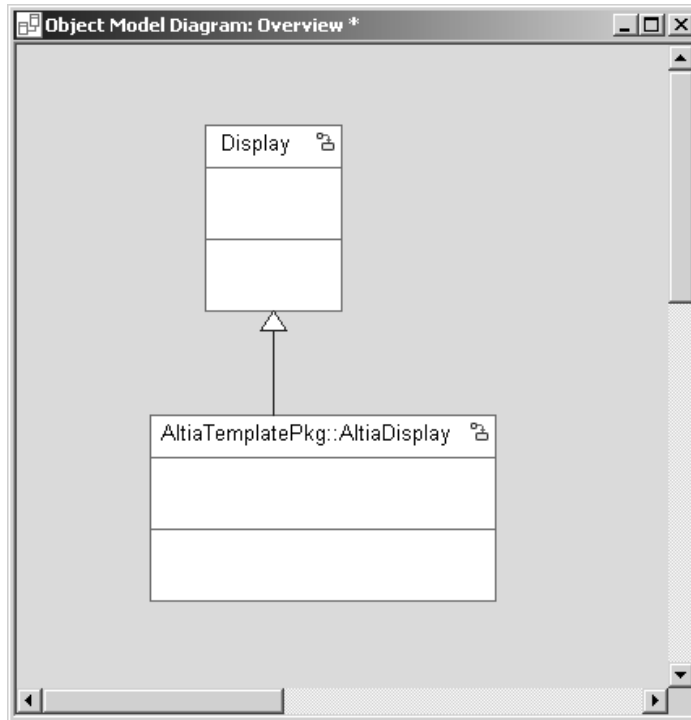


FIGURE 5-21 Finished Object Model Diagram

19. Minimize all of the Rhapsody windows except for the browser and then double click the **AltiaDisplay** constructor. The constructor is responsible for opening up an Altia design and registering callbacks for signals from Altia that we want to receive.
20. In the body of the constructor, edit the **initPanel** call to open **tutorial.dsn**. For example, edit the line to read:

```
initPanel("tutorial.dsn");
```

21. Further down in the body of the constructor, edit the sample “AddCallback” statements to add callbacks only for **slider** and **power**. When you are done, there should only be two `RPYAddCallback` lines and they should read:

```
RPYAddCallback("slider", (void*)onSliderCBK);
RPYAddCallback("power", (void*)onPowerCBK);
```

Be sure to delete any “AddCallback” statements that you are not using.

22. Now let’s create the **onSliderCBK** operation that the first AddCallback statement references. Since one of the callback templates is specifically designed for slider-type connections, we can just edit it.
23. Select the **onSliderCBK_template** operation and change its name to **onSlider-CBK**. At the end of its body, add the following statements:

```
me->Value = (int) val;
me->GEN(onUpdate());
```

This will set the attribute **Value** to the current value of the slider and then cause the statechart to transition from On back to itself (if it is in the On state).

24. Now, let’s create the **onPowerCBK** operation. Select the **onEventCBK_template** operation and change its name to **onPowerCBK**. At the end of its body, add the following statements:

```
if (eventValue)
    me->GEN(onPowerUp());
else
    me->GEN(onPowerDown());
```

This will send an **onPowerUp** or **onPowerDown** event based on the value coming in from the power button.

25. Let’s turn our attention to the functions that will send data to Altia. Select the **showValue_template** operation and change its name to **showValue**. Edit its `RPYSendEvent` statement to read:

```
RPYSendEvent("meter", (AltiaEventType) Value);
```

This will send the value that is passed in as an argument to the Altia **meter** animation.

26. Copy the **showValue** operation created in the previous step by holding down the Control key, clicking on the **showValue** operation and dragging it up to the **AI-**

AltiaDisplay class' **Operations** node. Rename it **showPower** and change the **RPY-SendEvent** statement to read:

```
RPYSendEvent("power_led", (AltiaEventType) Value);
```

This will send the value that is passed in as an argument to the Altia **power_led** animation.

27. We won't be using any of the extra operation templates. You can delete them if you like, but they won't hurt anything if you just leave them.
28. Right-click on the **AltiaGui** Component and choose **Set as Active Component**. Double click the component to open it. Change the **Libraries** and **Include Path** directories to match your Altia installation path (by default, they will be something like `c:\usr\altia`). Also, at the end of the **Libraries** line, make sure that it has `user32.lib`.
29. Save your Rhapsody model as **tutorial.rpy**. From the **Code** menu choose **Generate/release**. Then, from the **Code** menu, choose **Build AltiaGui.exe**. Finally, from the **Code** menu, choose **Run AltiaGui.exe**.

6.0 Test Your Altia/Rhapsody Interaction

1. The **release** configuration is set to animate the Rhapsody statecharts. To see this in action, click the **Go Idle** button on the Animation toolbar.
2. After the Altia design is opened in Runtime only mode, choose **Animated Statechart** from the **Tools** menu. In the dialog that opens, choose the **AltiaDisplay** instance and press **OK** to bring up the animated statechart.
3. Press the **Go** button to let the code run.
4. Go to the Altia window and press the **Power** button to put the model in the "On" state and begin operating the Altia objects with your mouse.
5. When you are finished interacting with the model, press the **Animation Break** button and then press the **Quit Animation** button. You will have to close your Altia Runtime window manually.

7.0 Tutorial Summary

In this tutorial, we have created an Altia GUI using libraries of pre-built components. We then created a simple Rhapsody model and used the connection dialogs in

Altia Design to connect it to the GUI. Finally, in order to be sure that our connections were correct, we ran the model and stimulated it using our Altia interface.

