

**Rapid Start**

## Rapid Start

© 2004 e-SIM Ltd. All rights reserved.

e-SIM Ltd.  
POB 45002  
Jerusalem  
91450  
Israel

Tel: 972-2-5870770

Fax: 972-2-5870773

Information in this manual is subject to change without notice and does not represent a commitment on the part of the vendor. The software described in this manual is furnished under a license agreement and may be used or copied only in accordance with the terms of that agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the express written permission of e-SIM Ltd.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Other product and company names mentioned in this manual may be trademarks or registered trademarks of their respective owners.

# Contents

About the RapidPLUS Documentation . . . . .	v
Conventions Used in This Manual . . . . .	vii
<b>CHAPTER 1: WHAT IS RAPIDPLUS? . . . . .</b>	<b>1</b>
RapidPLUS Concepts . . . . .	2
Building Simulations with RapidPLUS Tools . . . . .	4
Objects and the Object Layout . . . . .	6
Modes and the Mode Tree . . . . .	7
Logic and the Logic Editor . . . . .	9
Modes and the State Chart . . . . .	10
Runtime and the Prototyper . . . . .	11
Testing and the Debugger . . . . .	12
<b>CHAPTER 2: RAPIDPLUS TUTORIAL . . . . .</b>	<b>13</b>
Stage 1: Defining the Objects . . . . .	14
Opening the Object Layout . . . . .	15
Adding a Switch . . . . .	16
Adding Labels . . . . .	17
Adding a Lamp . . . . .	20
Adding a Text Display . . . . .	20
Saving the Application . . . . .	21
Stage 2: Defining the Modes . . . . .	21
Adding the Off and On Modes . . . . .	21

Stage 3: Defining the Transitions . . . . .	24
Creating the Transition from Off to On . . . . .	24
Creating the Transition from On to Off . . . . .	25
Summary of Stages 1 to 3. . . . .	26
Stage 4: Defining the Triggers . . . . .	27
Understanding the Properties and Functions . . . . .	27
Triggering the Transition from Off Mode to On Mode . . . . .	28
Triggering the Transition from On to Off . . . . .	30
Stage 5: Defining the Activities. . . . .	31
Types of Activities . . . . .	32
Defining the Entry Activities for On Mode . . . . .	32
Defining the Entry Activities for Off Mode . . . . .	34
Stage 6: Testing the Application . . . . .	35
<b>CHAPTER 3: FURTHER PRACTICE . . . . .</b>	<b>37</b>
Defining System Requirements. . . . .	38
Exercise 1 Requirements: Blinking Lamp . . . . .	38
Exercise 2 Requirements: Timed State . . . . .	38
Guidelines for Completing the Exercises . . . . .	39
Exercise 1: Creating a Blinking Lamp. . . . .	39
Exercise 2: Adding a Timed State . . . . .	42
<b>GLOSSARY . . . . .</b>	<b>45</b>

## ABOUT THE RAPIDPLUS DOCUMENTATION

Depending on the RapidPLUS package you've purchased, some of the printed manuals (and/or PDF files) described in the following table will be included.

MANUAL	DESCRIPTION
<i>Installation Guide</i>	Instructions for installing a licensed version of RapidPLUS on a single-user or client computer, and instructions for installing and managing floating (network) licenses on a Windows-based server or UNIX system server.
<i>Rapid Start (this manual)</i>	Introduction to basic concepts for building RapidPLUS applications, and a tutorial to introduce some of the tools and how they work together.
<i>User Manual</i>	In-depth information for building RapidPLUS applications, including: <ul style="list-style-type: none"><li>• General information on the RapidPLUS interface.</li><li>• Working with different kinds of objects.</li><li>• Building and checking logic.</li><li>• Building and reusing user functions and user objects.</li><li>• Working with the completed application.</li></ul>
<i>User Manual Supplement</i>	Information about objects, logic, tools, and features developed since Rapid 4.0, including: <ul style="list-style-type: none"><li>• Application management features.</li><li>• RapidPLUS applications as XML files.</li><li>• Logic evaluation and debugging features.</li><li>• Object features and enhancements, and new objects.</li></ul>

<b>MANUAL</b>	<b>DESCRIPTION</b>
<i>Generating Documents</i>	Information about the Document Manager tool, which is used to automatically generate specification documents and acceptance test documents directly from RapidPLUS applications.
<i>Generating Code for Embedded Systems</i>	Reference for developing RapidPLUS applications that will be generated as executable C code applications, which run on real embedded systems.
<i>Methodology Guide: Building Applications for Embedded Systems</i>	Guidelines for developing large-scale applications that will be generated as C code for integration into embedded systems.
<i>Generating Web Simulations</i>	Reference and methodology for developing RapidPLUS applications that will be generated as Java applications (applets) for use on the Internet, an intranet, or a CD-ROM.
<i>Using the Scenario Authoring Tool</i>	The Scenario Authoring Tool (SAT) is used in conjunction with RapidPLUS applications generated as applets. This reference is for creating scenarios—multimedia presentations and use cases that are recorded for a RapidPLUS-generated applet.
<i>RapidPLUS Xpress User Manual</i>	Tutorial and reference for using RapidPLUS Xpress to build interactive graphical models of future products, and generate screen transition charts from the prototypes.

## RapidPLUS Help



F1

In addition to the printed documentation or PDF files, RapidPLUS provides context-sensitive Help. Click the Help button in the toolbar and then click a button, menu command, window area, or a listed function or event.

## CONVENTIONS USED IN THIS MANUAL

This manual was written with the assumption that you are familiar with Microsoft® Windows® conventions.

The RapidPLUS documentation uses the following conventions:

- “Choose File|Save” means to select the Save command from the File menu.
- Names of properties, functions, and events are italicized:  
*blinkPeriod* property, *changeBy:* function, *cursorEntered* event
- File names are italicized:  
*MySystem.rpd*
- Complete phrases of RapidPLUS logic appear in bold, sans serif characters:  
**& Switch2.position3 is connected**





## *What is RapidPLUS?*

RapidPLUS is a comprehensive software package for the generation of simulations and prototypes of embedded systems. These prototypes are used to simulate, document, and generate code for products. Although you can use RapidPLUS to simulate any type of system, its primary target is the simulation of Man-Machine Interfaces (MMI) for electronic products such as mobile phones and digital cameras.

There are three distinct stages to building a RapidPLUS **application**:

- 1 Define the physical components and visual layout.
- 2 Build logic to formulate the behavior of the application.
- 3 Test the application, return to the physical and logical design to adjust or improve it, and test it again.

Through this interactive process, you generate a detailed, tested, and realistic simulation before engaging in the costly and time-consuming process of building a physical prototype. The virtual product simulations can be used to generate code for the physical devices, and for putting the simulations on the Web for training, support, and marketing.

## RAPIDPLUS CONCEPTS

A RapidPLUS application is a description of a system. A system can be described through its visual elements and its functionality.

Visual elements, such as switches, buttons, indicators, and displays, are used to manipulate the system. Nonvisual elements, such as timers, font definitions, and arrays, are also used to control the system. The graphic and nongraphic elements, called **objects**, give a system its physical appearance and control how users interact with the system. Every object has its own predefined set of properties and functions that describes everything it can do in a real-life situation. A lamp, for example, can be on, off, or blink. A 2-position rocker switch can be up or down.

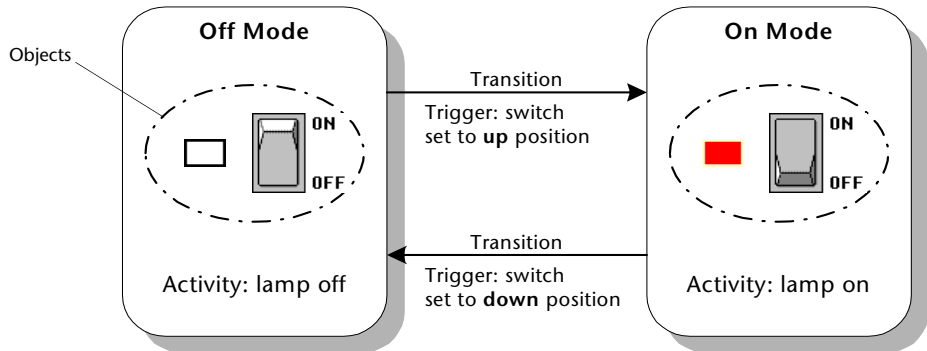
The system's overall behavior can be divided into separate units of functionality, called **modes**. For example, a television can be on or off; therefore, On and Off are both modes of the television. A description of a complex device, such as a mobile phone or an aircraft hydraulic system, would have many other modes besides On and Off. While a phone is on, for example, there are several possible modes that you could describe: Dialing, Line Status, Outgoing Call, Incoming Call, and so on. The telephone, as a system, could be in one or several of these modes at any given time.

After objects and modes are defined, the possible changes to the system are defined through the movement, or transitions, between the modes. A **transition** is a move from one (source) mode to another (destination) mode when a specified trigger takes place. **Triggers** are events or conditions that activate the transition. For example, the transition from Off mode to On mode occurs when *triggered* by the press of the Power pushbutton. Triggers can be expressed as statements such as "when the lamp is on" or "when the timer reaches zero." Triggers are constructed using the objects' properties and functions.

To define the behavior of the application—that is, what the simulation does in each mode—you can specify what operations are performed on objects when a mode is active. **Actions** are operations that are performed on an object when a particular transition is triggered. An action only occurs when its transition is triggered. **Activities** are operations performed on an object when a particular mode is active. Actions and activities can be expressed as statements such as "turn on the lamp," "start the timer" or "clear the display." Like triggers, actions and activities are constructed using the objects' properties and functions.

The following diagram shows the relationships among the elements in a simple application. The application contains:

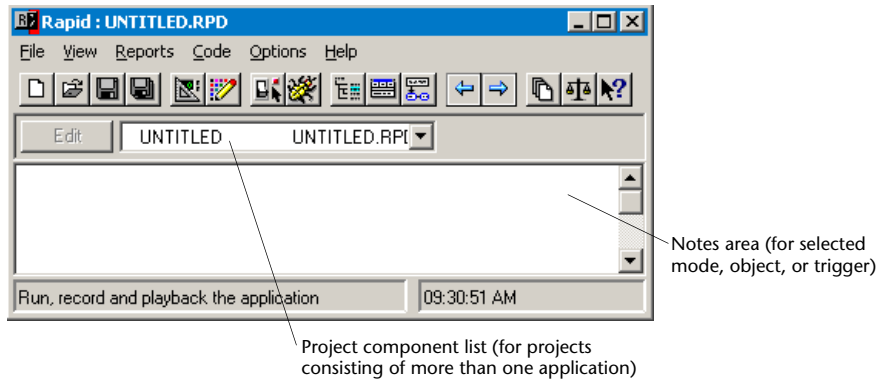
- Four objects: A lamp, rocker switch, and two labels for the switch.
- Two modes: Off and On.
- Two transitions: A transition from Off mode to On mode is triggered by moving the switch to the up position. A transition from On mode to Off mode is triggered by moving the switch to the down position.
- Two activities: An activity in Off mode turns off the lamp. An activity in On mode turns on the lamp.



## BUILDING SIMULATIONS WITH RAPIDPLUS TOOLS










The RapidPLUS design environment is comprised of tools and utilities, each in an independent window. The main window is referred to as the **Application Manager**. It is used to:

- Manage (open, close, save) files.
- Set general/global application parameters.
- Generate reports on various aspects of the application.
- Add notes about modes, objects, and transition triggers.
- Open all the other tools.



**Application Manager window**

You can open or activate each tool by clicking its button in the Application Manager. The primary design tools are summarized in the following table:

TOOL	PURPOSE
 <i>Object Layout</i>	Add graphic and nongraphic objects to the application and arrange the physical layout of the graphic objects.
 <i>Object Editor</i>	Edit object bitmaps, pointers, and active areas. Create certain types of new graphic objects.
 <i>Prototyper</i>	Test the application's physical and logical design.
 <i>Debugger</i>	Pause the application at specified breakpoints, execute logic step by step, and examine logic lines in the Call Stack and Logger panes.
 <i>Mode Tree</i>	Create a hierarchical tree of modes that represents the functionality of the prototype.
 <i>Logic Editor</i>	Build the logic that governs the simulation's functionality, such as transitions from mode to mode, triggers that activate the transitions, and the actions and activities that take place when a mode is active.
 <i>State Chart</i>	Examine a graphic representation of the mode tree and transitions in a nested chart format.
 <i>Document Manager</i>	Generate documents for product specification descriptions and test procedures.
 <i>Differencing Tool</i>	Compare objects, modes, and other elements of applications. It is particularly useful for comparing two versions of the same application or project.



## Objects and the Object Layout

Objects are visual and nonvisual elements of an application with predefined sets of properties and functions. They are reusable elements that represent real-life objects, such as pushbuttons, lamps, indicators, and timers.

When you create a new application, you will usually start with the Object Layout to define and arrange the objects that are required for:

- User control, such as switches and potentiometers.
- Output display, such as dials and alphanumeric displays.
- Status indication, such as lamps.
- General control of the application's behavior, such as timers.

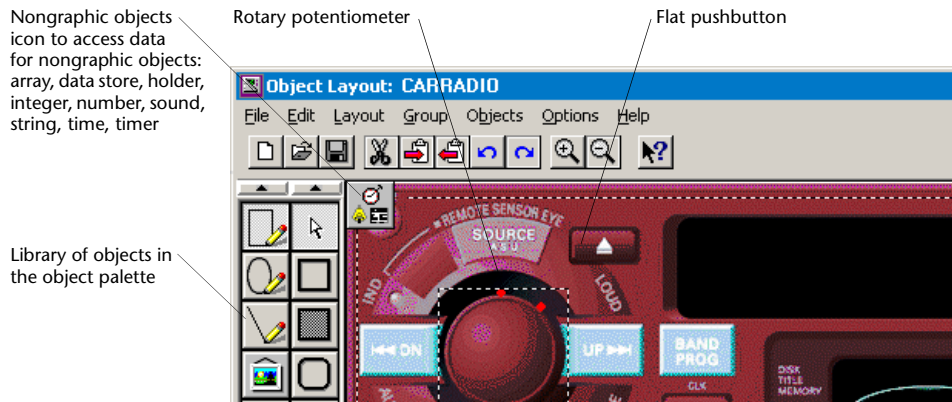
An application's objects are in a hierarchical (i.e., parent-child) relationship to each other; every object must have a **parent**. When a parent is hidden, shown, or moved during runtime, its **children** are similarly affected.

The Object Layout provides an extensive object palette that can be classified into several broad types of objects:

<b>OBJECT TYPE</b>	<b>DESCRIPTION</b>
<i>Active objects</i>	<b>Graphic objects</b> such as, switches, potentiometers, pushbuttons, dials, indicators, lamps, and displays. They can be manipulated in runtime; for example, the knob on a rotary switch can be turned with the click of the mouse.
<i>Primitive objects</i>	Graphic elements used to enhance the application's physical appearance. These objects include several styles of lines, frames, circles, and ellipses, as well as text labels. They are usually passive (i.e., cannot be manipulated during runtime).

OBJECT TYPE	DESCRIPTION
<i>Nongraphic Objects</i>	Nonvisual elements, such as timers, stopwatches, and data objects that can hold strings, real numbers, or integers. For example, a timer does not appear on the layout, but it can be used to cause a lamp to turn on.
<i>Additional objects</i>	Additional objects include communication, multimedia, external (RPX), ActiveX, and JavaBean objects, as well as user objects. These objects can greatly expand the ability to simulate complex systems. Refer to the user documentation for more information.

The following illustration is an application designed in the Object Layout. It contains an imported bitmap, nongraphic objects, and customized active objects:



## Modes and the Mode Tree



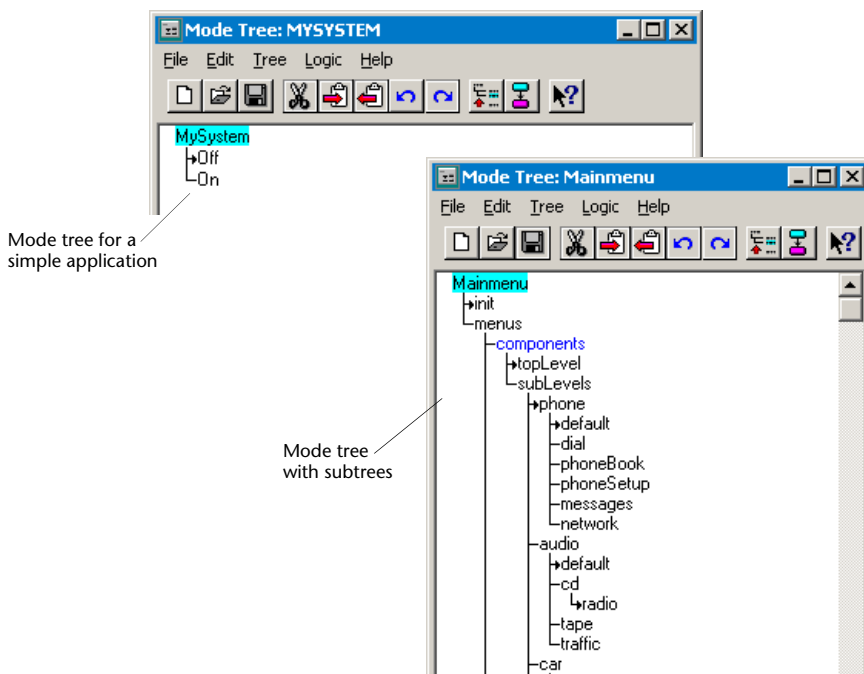
A mode is a distinct unit of functionality that describes what a system does at a specific moment. The system's behavior can be described through a single mode, or through a group of modes.

For example, a mobile phone can be on or off; therefore, On and Off are both modes of a mobile phone application. The behavior of On mode can be described by additional modes such as Place a Call, Talk, and End a Call. The mobile phone can be in one or several of these modes at any given time.

Each mode contains a collection of activities which RapidPLUS performs on the objects when the mode is active. Each activity can take place either upon entry into the mode, upon exit from the mode, or for the entire time that the mode is active. As your application moves from mode to mode, sometimes activating one set of modes, sometimes activating several sets of parallel (concurrent) modes, all of the relevant activities are performed on the objects.

In the Mode Tree, you define the application's modes and organize them into a hierarchical tree. The mode at the top of the tree is called the **root mode**. It is created automatically whenever you start a new application and it takes the application's name. All other modes that you add are descendants of the root mode.

A very simple application would have at least two modes (for example, Off and On), which would be children of the root mode. More complex applications have many modes, each mode containing a collection of related activities.





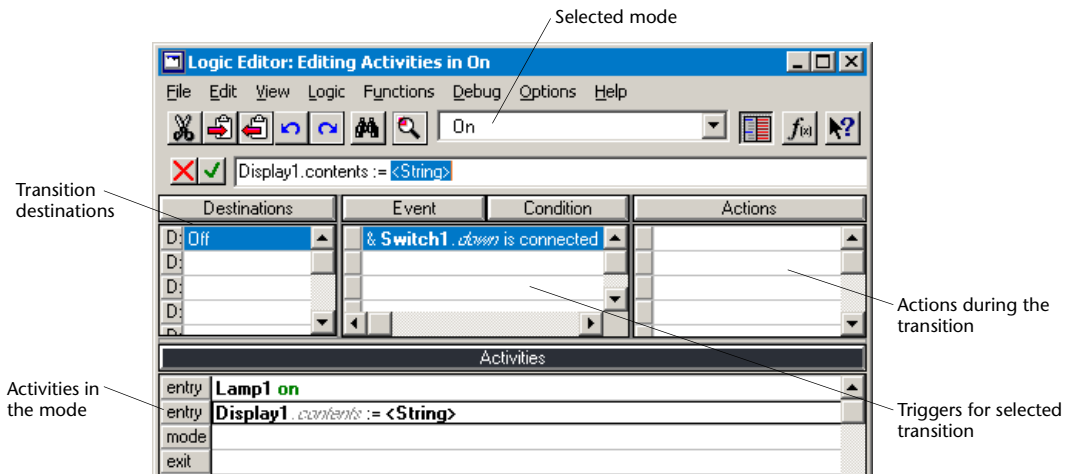


## Logic and the Logic Editor

**Logic** statements govern the behavior of the application. Logic statements are built using object-specific properties and functions, and they are organized and stored in the Logic Editor. The Logic Editor uses a table format, arranged according to each mode in the application. Another window, called the **Logic Palette**, provides a list of all the objects in the application, and lists all of the object-specific properties and functions.

There are four main kinds of logic:

<b>LOGIC TYPE</b>	<b>DESCRIPTION</b>
<i>Transitions</i>	<p>The possible system changes, that is, how the application moves from one mode to another.</p> <p>Here the mode hierarchy plays an important role because a parent mode becomes active when one of its children becomes active, and a child mode becomes inactive when there is a transition away from its parent.</p>
<i>Triggers</i>	<p>Inputs, called conditions and events, that trigger the transition.</p>
<i>Activities in Modes</i>	<p>Object manipulations that take place when a mode becomes active. Activities relate to the time frame in which are performed:</p> <ul style="list-style-type: none"> <li>• Entry activities occur as the mode becomes active.</li> <li>• Exit activities occur as the mode becomes inactive.</li> <li>• Mode activities occur continuously while the mode is active.</li> </ul>
<i>Actions</i>	<p>Object manipulations that take place only during a transition. An action is attached to a specific transition, and only occurs when that transition occurs.</p> <p>To understand the difference between activities and actions, think of activities as occurring “within” modes, and actions as occurring “in-between” modes.</p>



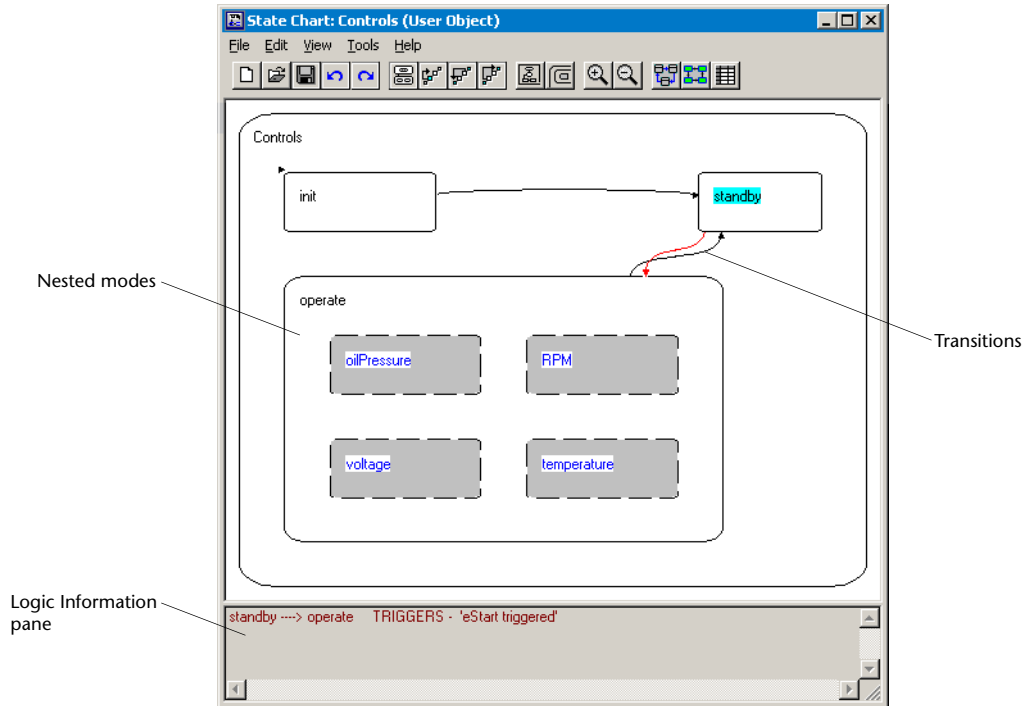
Logic Editor

## Modes and the State Chart



The State Chart is an auxiliary tool to the Mode Tree. The State Chart presents the application's modes—and the transitions between them—in chart form. In the State Chart, the mode hierarchy is represented by nested figures (as opposed to the branched mode tree in the Mode Tree window) and the transitions are represented as arrows from one mode to another.

The State Chart also includes a Logic Information pane that provides information on the activities and triggers of a selected mode or transition.



## Runtime and the Prototyper



The Prototyper is used to test the runtime operation of the application. Controls such as switches, potentiometers, and pushbuttons are operated by clicking an active area, or hotspot, on the graphic representation of the control. The output (text displays, dials, lamps, etc.) changes in response to your input, according to the logic defined in the Logic Editor.

You can stop the application in the Prototyper in order to make changes to the application's objects or logic. When you run the application again in the Prototyper, the modifications are immediately implemented. Through this interactive process, you can quickly and effectively improve your application's appearance and behavior.



## Testing and the Debugger

Running the Prototyper through the Debugger tool enables you to step through the application execution, log application activity, and view the application call stack.

Once you have tested and debugged the application, and it is working as you intend, the Prototyper becomes a useful demonstration and training tool.

## *RapidPLUS Tutorial*

This tutorial provides the most common ways to use the main editing tools. For more information about options and features, you should consult the user documentation.

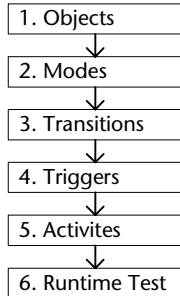
The simple application that you will build in this tutorial has the following components:

- A labeled switch that turns power on and off.
- A lamp that lights up when the power is on.
- A text display that displays the status of the switch.

You will turn your application into a tested prototype by focusing on six distinct aspects:

- 1 **Objects**—Create a visual representation of the application’s user interface (a rocker switch, a lamp, a text display, and labels for the switch positions).
- 2 **Modes**—Create the mode hierarchy that defines the application’s possible operational states (On and Off).
- 3 **Transitions**—Define the transitions between the application’s modes (the application can move from On mode to Off mode and vice versa).
- 4 **Triggers**—Define the triggers that activate the transitions (e.g., the application will move from Off mode to On mode when the On/Off switch is in the On position).
- 5 **Activities**—Define activities that take place when a mode is made active (e.g., when On mode is active, the lamp turns on).
- 6 **Runtime Test**—Test the application in the Prototyper to see that it works as specified.

The following chart will be used to track your progress through the tutorial:

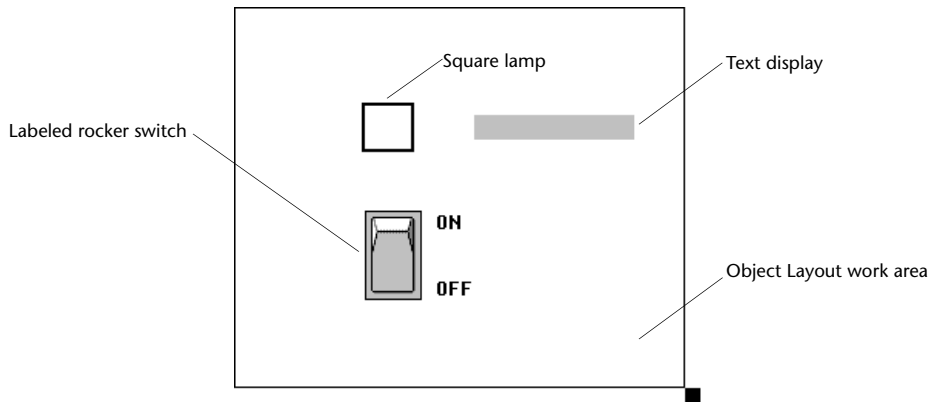


## STAGE 1: DEFINING THE OBJECTS

In this stage, you will use the Object Layout to create the application’s user interface. You will add the following objects:

- A rocker switch (an active object).
- ON and OFF labels for the rocker switch (primitive objects).
- A square lamp (an active object).
- A text display (an active object).

When you have finished, the layout will look similar to the following:

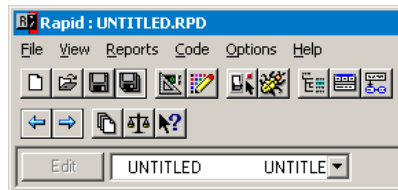


## Opening the Object Layout

The Object Layout window opens automatically with the RapidPLUS default settings, but if the default settings have been changed, you can activate the Object Layout (and other tools) through the Application Manager.

### To open the Object Layout:

- 1 Run RapidPLUS; the Application Manager opens, as well as some of the design tools.

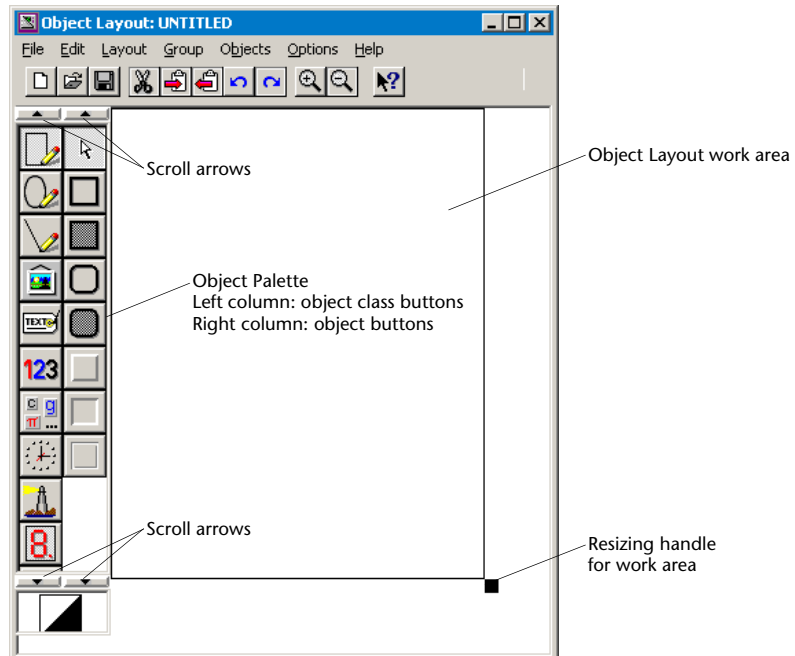


Application Manager window



Ctrl+A

- 2 Choose View|Object Layout, or click the Object Layout button to activate the Object Layout window.

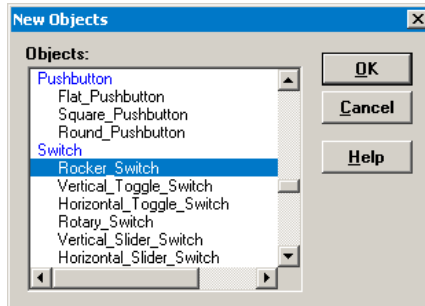


## Adding a Switch

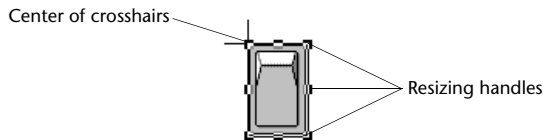
The first object you will add to the layout is a switch. RapidPLUS provides a variety of switches, any one of which could implement the On/Off function. For this tutorial, you will use a 2-position rocker switch.

To add a switch:

- 1 In the Object Layout, choose Objects|Add; the New Objects dialog box opens listing all the objects by class. Classes are displayed in blue text, objects are displayed in black text.
- 2 Scroll down the list until you see the Switch class. Select **Rocker Switch**, then click OK.



- 3 Over the Object Layout, the cursor changes to crosshairs. The center of the crosshairs represents the upper-left corner of the object's enclosing rectangle:



Move the cursor to the middle of the object layout work area and click once (without moving the mouse); the rocker switch is added to the layout at its default size.

- 4 (Optional) To resize the switch, drag one of the resizing handles (visible when the switch is selected) to the required size.
- 5 (Optional) To move the switch, drag it to a new position.





## Fixing mistakes

The RapidPLUS editing tools have multilevel, global Undo and Redo commands. If you make a mistake, you can click the Undo button, or choose Edit|Undo.

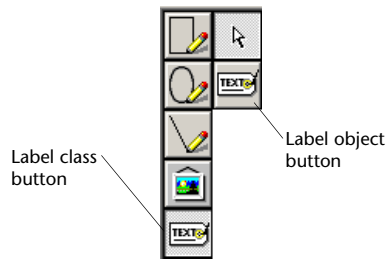
## Adding Labels

Now you will add labels to the switch's Off and On positions so that the user will know which switch position corresponds to which operating mode.

### Adding the Off Label

To add an Off label:

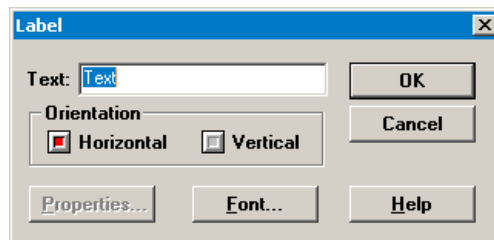
- 1 Click the Label class button in the left column of the Object Palette:



❖ *NOTE: You can also choose Objects|Add and select Label.*

The label object button (in the right column) is already selected.

- 2 Position the crosshairs to the right of the rocker switch's lower position and click; the Label dialog box opens:



- 3 Type **OFF** in the Text box, then click OK. The Off label appears where you positioned the crosshairs, with the default horizontal orientation and font characteristics. Your switch should look similar to the following:



### Adding the On Label

After you add an object to the layout, its button in the right column of the Object Palette is deselected. There is an option (Objects|Continuous Adding) that causes the button to remain selected in order to continuously add objects of the same type. For the purposes of this tutorial, however, the Continuous Adding option is not necessary.

**To add an On label:**

- 1 Click the label object button in the right column of the Object Palette.
- 2 Position the crosshairs to the right of the rocker switch's upper position and click; the Label dialog box opens.
- 3 Type **ON** in the Text box, then click OK. Your final switch should look similar to the following:



- ❖ *NOTE: If the objects are not perfectly aligned, you can use the alignment features in the Layout menu.*

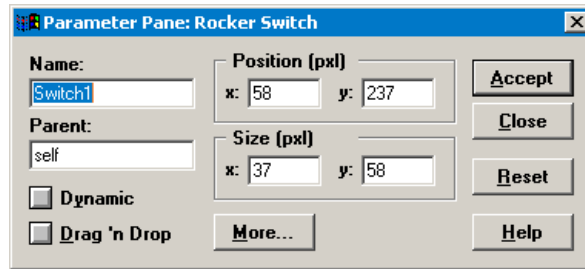
---

---

## What happens when you double-click an object?

For every object in the layout, RapidPLUS stores the size and location information. If the object is an active object, it is assigned a default name. All active objects must have names so that they can be manipulated in the logic statements. Names are not assigned to primitive objects.

You can view and change the size and position information by double-clicking the object to open its parameter pane:



Notice the Parent box, which by default identifies the application (“self”) as the parent of Switch1. Every active and primitive object that is added to the layout must have a parent and the default parent is the application itself.

The parameter pane holds information for all active and primitive objects. At this point in the tutorial you do not need to use it; however, you may want to view the parameter pane for each object in your application.

### To view the parameter pane for each object:

- 1 Double-click an object on the layout; its parameter pane opens.
- 2 Click a different object. Notice that the parameter pane stays open, but the information changes.

The parameter pane has many purposes. For more details, refer to the user documentation.

---

---

## Adding a Lamp

The next object to add to the layout is a lamp that will light when the switch is in the On position.



To add a lamp:

- 1 Click the Lamp class button in the left column of the Object Palette, then click the square lamp button in the right column.
- 2 Position the crosshairs above the On/Off switch where you would like to position the lamp's upper-left corner.
- 3 Drag the lamp's enclosing rectangle until you are satisfied with its size and shape.

The square lamp now appears on the layout at the size that you specified:



## Adding a Text Display

The last object to add is a text display that will display the status of the switch.



To add a text display:

- 1 Click the Display class button, then click the text display button.
- 2 Position the crosshairs to the right of the lamp and click. A blank text display is added to the layout.



## Saving the Application

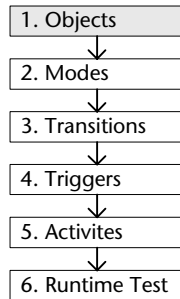


To save an application:

- 1 Choose File|Save, or click the Save button.
- 2 Save the application as “MySystem.rpd”.

❖ *NOTE: For the rest of the tutorial, you should save the application at regular intervals.*

You have finished building the application’s layout. Next you will define the distinct operational states—that is, the modes—that describe the system’s behavior.



## STAGE 2: DEFINING THE MODES

The On/Off switch implements two operational possibilities: Off (no power connected) and On (power connected). In order to operate the application, each of these possibilities must be defined as a mode in the Mode Tree.

The Mode Tree tool—the one in which you will build the application’s hierarchical mode tree—displays the application name as the root mode. In this stage, you will add two modes, Off and On, as children of the root mode.

### Adding the Off and On Modes

The Mode Tree window opens automatically with the RapidPLUS default settings, but if it is closed, you can activate the Mode Tree through the Application Manager.



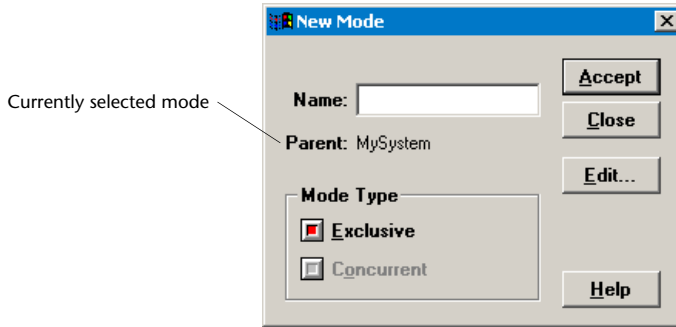
Ctrl+T



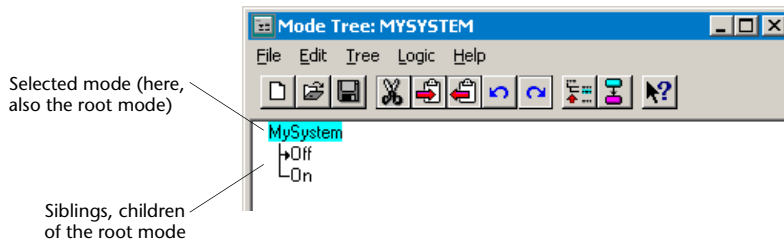
Ctrl+W

**To add modes to the Mode Tree:**

- 1 In the Application Manager, choose View|Mode Tree or click the Mode Tree button. This activates the Mode Tree window. The application name is displayed as the root mode.
- 2 In the Mode Tree, choose Tree|New Mode, or click the New Mode button, to open the New Mode dialog box:



- 3 Type **Off** in the Name box. Note that its parent is the currently selected mode (in this case, the root mode).
- 4 Click Accept to add the Off mode to the mode tree as a child of the root mode. The Name text box clears automatically.
- 5 Now type **On** in the Name box. Note that its parent is the root mode.
- 6 Click Accept to add the On mode to the mode tree. It is a child of the root mode, and a **sibling** of the Off mode.
- 7 Click Close. Your mode tree should now look like:



---

---

## About exclusive modes

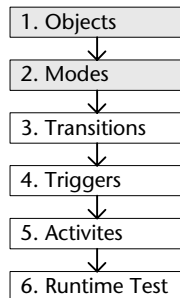
The Exclusive mode type button was selected when you added the Off mode. An **exclusive mode** is a mode that cannot be active at the same time that a sibling mode is active. Exclusive modes are used for specific states of the system that cannot run simultaneously. For example, the system cannot be on and off at the same time.

Notice that the Off mode has an arrow pointing to it. This arrow indicates that it is the **default mode**, i.e., the mode that becomes active when its parent becomes active. When the application is started in the Prototyper, the root mode (MySystem) and its default child mode (Off) are both active.

---

---

You have finished defining the modes. Next you will define the transitions between the modes.



## STAGE 3: DEFINING THE TRANSITIONS

Now that you have defined the layout and have added modes to the Mode Tree, you can define pathways (transitions) between the modes. In this stage, you will create a transition from Off mode to On mode, and vice versa.

### Creating the Transition from Off to On

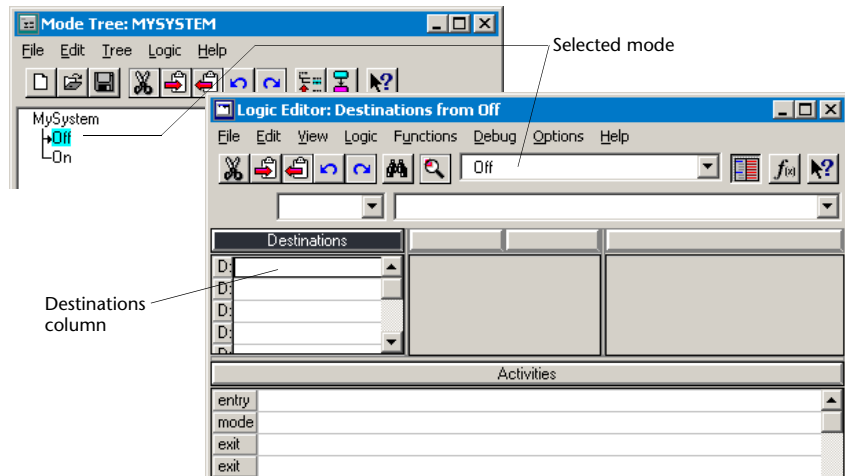
Transitions can be created through the Mode Tree and Logic Editor.

To create a transition from the Mode Tree:

- 1 In the Mode Tree, select Off.
- 2 Choose Logic|Transition (do not choose the Make Transition command) or click the New Transition button. The Logic Editor opens.



Ctrl+T

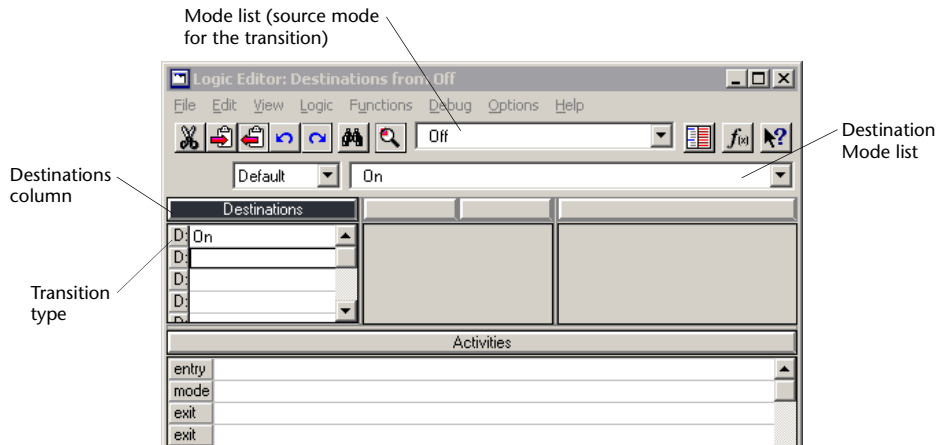


The Logic Editor's title bar indicates that the selected mode is Off and that the Logic Editor is in destination-editing mode. The first empty line in the Destinations column is selected. Note that the Logic Palette also opens, but it is not yet necessary.



- 3 In the Mode Tree, Alt-click the On mode. The selection color of the On mode flashes magenta while clicking it.

In the Logic Editor, On appears in the Destinations column and in the Destination Mode list:



For an external destination (from one mode to another), a letter at the head of the row indicates the type of transition; in this case, "D:" is for a default-type transition. The types of transitions are explained in the user documentation.

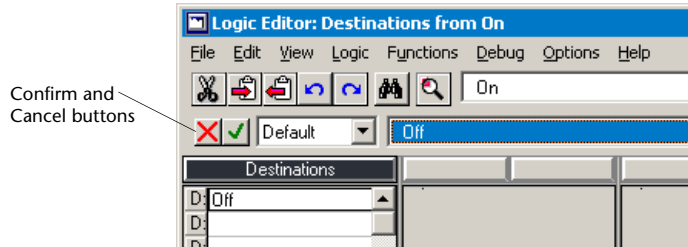
## Creating the Transition from On to Off

The first transition was created in the Mode Tree. To create the transition from On mode to Off mode, you will use the Logic Editor.

**To create a transition from the Logic Editor:**

- 1 In the Logic Editor, select On from the Mode list, which is the uppermost drop-down list (see above illustration).
- 2 Select Off from the Destination Mode list.

Off appears in the Destinations column and the Confirm and Cancel buttons appear:



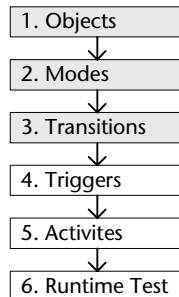
- 3 Click the Confirm button. RapidPLUS accepts the logic statement and the line remains active.

## Summary of Stages 1 to 3

To summarize the stages that you have completed so far:

- 1 You have designed the layout of the application by adding various objects in the Object Layout.
- 2 You have defined the different modes and built a hierarchical mode tree.
- 3 You have constructed the transitions that are to occur between the two modes.

The next stage is to define the triggers that will activate each transition.



## STAGE 4: DEFINING THE TRIGGERS

In this stage, you will define the triggers that activate the transitions between the On and Off modes. You will use the Logic Editor to build RapidPLUS logic—the logic statements that govern the behavior of the application.

### Understanding the Properties and Functions

Logic is built using an object's properties and functions. Properties control specific aspects of an object's behavior. Functions are commands that can adjust an object, a property, or a property value.

#### Properties

Properties of active and nongraphic objects are characteristics that can be viewed and manipulated during runtime. Each property controls a specific aspect of the object's behavior. For example, the rocker of a real 2-position rocker switch can be in either the up or down position; the equivalent RapidPLUS switch has the properties *up* and *down*. Another example would be a lamp, which has the property *blinkPeriod*. This property assigns the period of time the lamp blinks.

#### Functions

Every object and property has functions that can adjust the object, a property, or a property value.

##### *Functions that adjust a property's value*

Most properties have a value that is either an integer, a number, or a string. For example, a RapidPLUS lamp property, *blinkPeriod*, might be set to 500 msec., meaning the lamp will blink at 500 millisecond intervals. There are many arithmetic functions for *blinkPeriod*, such as the multiplication (\*) function and the *changeBy*: function, that enable you to alter the lamp's blink period during runtime.

##### *Functions that adjust the property itself*

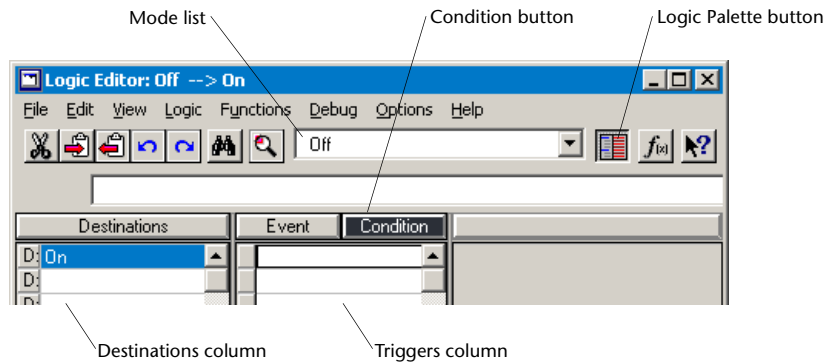
The rocker switch's *up* and *down* properties each have the function *connect*. When the *up* property receives the *connect* function, the rocker moves to the up position; when the *down* property receives the *connect* function, the rocker moves to the down position.

## Triggering the Transition from Off Mode to On Mode

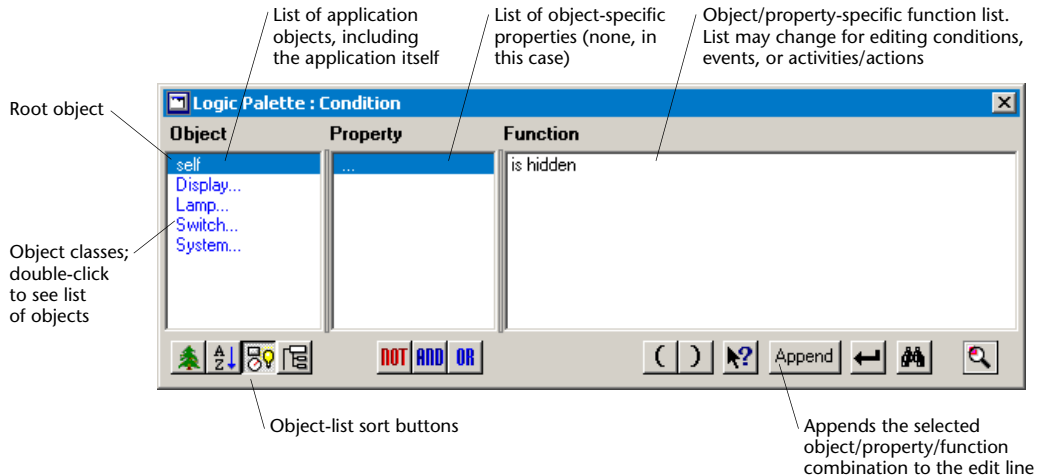
The circumstances that cause a transition to take place are known as a trigger. A trigger can be a condition that is true and/or an event that occurs. The trigger that initiates the transition between Off and On is the condition, "Switch1 is in the up position," being true. A discussion of condition triggers and event triggers is on p. 30.

### To define the trigger from Off to On:

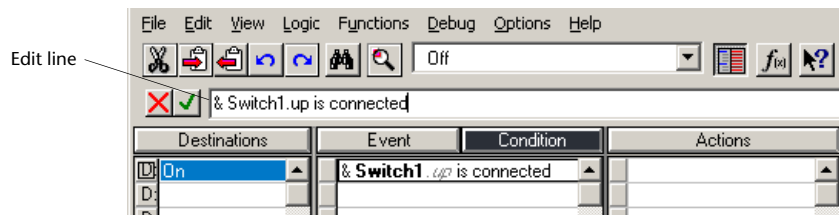
- 1 Select Off in the Logic Editor's Mode list. The Logic Editor now provides information about Off mode.
- 2 To trigger the transition to On mode, select On in the Destinations column, then click the Condition button at the top of the Triggers column:



- 3 If the Logic Palette is closed, click the Logic Palette button. The Logic Palette opens:



- 4 Click the Object Tree sort button to display the objects in a hierarchical tree, beginning with the root object. Notice that the active objects are listed with their default names. (Nongraphic objects are displayed at the bottom of the tree.)
- 5 Select Switch1 in the Object list. The object's properties (*down* and *up*) appear in the Property list, and the functions specific to the selected object or property appear in the Function list.
- 6 Select *up* in the Property list, then select *is connected* in the Function list.
- 7 Click the Append button. In the Logic Editor, the selected condition appears in both the edit line and the line in the Triggers column. Notice that conditions are preceded by an ampersand (&):



- 8 Click the Confirm button to confirm the logic.

## Triggering the Transition from On to Off

The trigger that initiates the On-to-Off transition is the condition, “Switch1 is in the down position,” being true.

### To define the trigger from On to Off:

- 1 Select On in the Logic Editor’s Mode list. The Logic Editor now provides information about On mode.
- 2 Select Off in the Destination column, then click the Condition button.
- 3 Switch1 should still be selected in the Logic Palette’s Object list (if not, select it). Select *down* in the Property list, then select *is connected* in the Function list.
- 4 Click the Append button. The condition appears in the Logic Editor’s edit line and the Triggers column.
- 5 Click the Confirm button to confirm the logic.

---

---

## Triggering with a condition or an event?

A transition takes place when a condition is true and/or an event occurs. You built triggers based on conditions. Here you will examine options for building triggers based on events.

To see the switch’s events, select Switch1 in the Object list and *down* in the Property list, then click the Event button. The switch’s events, *break* and *make*, are displayed (*cursorEntered* and *cursorExited* are also displayed; they are events of the mouse object and appear for all graphic objects).

When a switch is connected to any of its positions, it generates the *make* event, and this event can be used to trigger a transition. Logically, the *Switch1.down is connected* condition and the *Switch1.down make* event appear to have the same effect. What, if any, is the difference?

- A **condition** continually tests the status of an object’s property. As long as the switch is in the down position, the *& Switch1.down is connected* condition is always true.
- An **event** signifies a momentary change in an object’s status. When the switch is moved to the down position, the *Switch1.down make* event is generated once.

In our simple application, using an event or a condition would have the same effect. The difference in nuance between them, though, can sometimes be crucial.

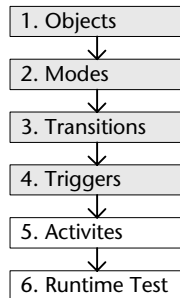
For example, consider a real lamp switch. When you turn the lamp on, the switch is connected to a new position. If there were to be a power outage, the lamp would go off. When the power returns, the lamp turns on again because the switch *is connected* to the “on” position.

If the “transition” from on to off in a real lamp switch were based on a momentary event, the lamp would not turn on again when the power returned. You would have to “reset” the lamp by flicking the switch from on to off and back again.

For this reason, it is “better” to base the transitions in this application on conditions, not on events.

---

The next stage is to define activities for the modes, i.e., what actually happens when each mode is active.



## STAGE 5: DEFINING THE ACTIVITIES

An activity is an operation performed on an object when a specific mode is activated. It is completely independent of transition destinations and triggers.

In this stage, you will define the following activities:

- When the application is in On mode (when Switch1 is in the up position), the lamp will turn on and the text display will display the word On.
- When the application is in Off mode (when Switch1 is in the down position), the lamp will turn off and the text display will be cleared.

## Types of Activities

There are three types of activities that relate to the time frame in which the activity is performed:

- **Entry activity**—occurs as the mode becomes active.
- **Mode activity**—occurs continuously while the mode is active.
- **Exit activity**—occurs as the mode becomes inactive.

Use entry and exit activities for operations that occur only once. Use mode activities for operations that occur continuously as long as the mode is active, and cease when the mode becomes inactive.

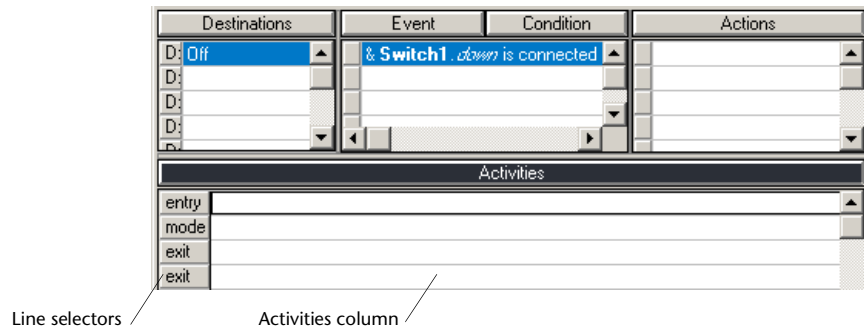
## Defining the Entry Activities for On Mode

You will define two entry activities to occur when On mode is entered:

- The lamp turns on.
- The word “On” is displayed.

**To turn on the lamp:**

- 1 Select On in the Logic Editor’s Mode list (if it is not already selected). The Logic Editor displays information about On mode.
- 2 Click the first blank line in the Activities column. Notice that the line selector indicates that the activity will be an entry activity:



Notice, too, that the Logic Palette’s Function list changed to show functions that apply to the activity type.

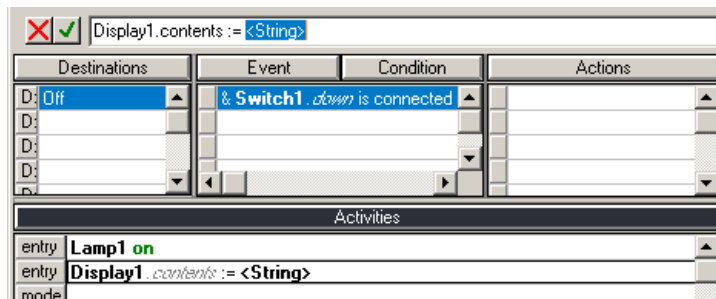
- 3 Select Lamp1 in the Object list and then select the *on* function from the Function list.



- 4 Click the Append button. The activity **Lamp1 on** appears in the edit line and in the line in the Activities column.
- 5 Press Enter. The logic line is confirmed and a blank line is added below it for a new entry activity.

#### To display the text:

- 1 Select Display1 in the Object list. Select *contents* in the Property list and *:= (assign)* in the Function list.
- 2 Click the Append button. The following incomplete activity appears in the edit line and in the line in the Activities column:



- 3 You are prompted to add a string. Overtyping <String> with 'On' (including the single quotation marks).
- 4 Click the Confirm button.

---

### Building logic that contains strings

Whenever you type a logic phrase that requires a string, you must place the string within single quotation marks. For example:

```
Display1.contents := 'Calibrating...'  
& String = 'The Smith Company'
```

If the string is made up exclusively of numbers (e.g., 123), you do not need to use single quotation marks.

---

## Defining the Entry Activities for Off Mode

You will define two entry activities to occur when Off mode is entered:

- The lamp turns off.
- The text display is cleared.

### To turn off the lamp:

- 1 Select Off in the Logic Editor's Mode list.
- 2 Click the first blank line in the Activities column.
- 3 In the Logic Palette, select Lamp1 in the Object list, then double-click *off* in the Function list. The activity **Lamp1 off** appears in the edit line and in the line in the Activities column.
- 4 Click the New Line button in the Logic Palette. The logic is confirmed and the next blank line is selected (same as pressing Enter).



### To clear the text display:

- 1 Select Display1 in the Object list, *contents* in the Property list, then double-click *clear* in the Function list.

The activity **Display1.contents clear** appears in the edit line and in the line in the Activities column.

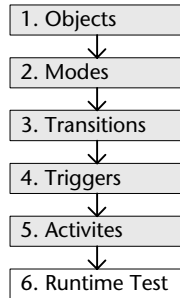
- 2 Click the Confirm button.

The Activities column should look like:

Activities	
entry	<b>Lamp1 off</b>
entry	<b>Display1.contents clear</b>
mode	
exit	

❖ NOTE: *Be sure to save your application.*

You have finished building the logic and can now go on to testing the application.



## STAGE 6: TESTING THE APPLICATION

Now you will use the Prototyper to test the application.

### To open the Prototyper and test the application:



Ctrl+R



- 1 In the Application Manager, either choose View|Prototyper or click the Prototyper button.

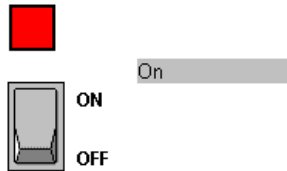
The Prototyper window opens—empty—because you only see the application when you run it.

- 2 Click the Prototyper's Start button. The application is displayed in the Prototyper. If the entire layout is not visible, resize the Prototyper window.

Check the Prototyper Options menu and be sure that the **Trace** option is selected. The Trace option highlights **active modes** in the Mode Tree as the Prototyper runs, so you can check that each mode is being entered and exited. The modes being traced are highlighted in gray; the mode that is highlighted in cyan is the mode that was last selected in the Mode Tree.

The system enters the root mode and the Off mode; these modes are highlighted in the Mode Tree.

- Click the top portion of the On/Off switch. The switch moves to the up position, the lamp goes on, and the word On is displayed:



On mode is highlighted in the Mode Tree.

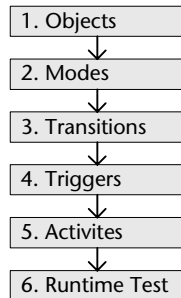
- Click the lower portion of the On/Off switch. The switch changes to the down position, the lamp goes off, and the text display is cleared.

Off mode is highlighted in the Mode Tree.

❖ *NOTE: If your application does not behave as expected, retrace your steps and ensure that you carried out the instructions as required.*



- Click the Prototyper's Stop button. The application disappears from the Prototyper window.



Congratulations! You have completed the six basic stages required to develop a RapidPLUS application. Along the way, you learned many features of the main editing tools.

## *Further Practice*

Experimenting with different objects, properties, and functions can help you better understand the variety and complexity of simulations that can be built with RapidPLUS.

This chapter presents two exercises to enhance the application that you built in the previous chapter. After defining the exercise requirements, there are guidelines to help you work through the recommended stages of development: objects, modes, transitions, triggers, activities, and application testing.

Although the provided examples have simple changes, they are intended to introduce you to more RapidPLUS concepts, such as:

- Parent-child objects.
- Parent-child (nested) modes.
- Event triggers.
- Timer object (a nongraphic object).

## DEFINING SYSTEM REQUIREMENTS

The first exercise builds on to your tutorial application, *MySystem.rpd*. The second exercise changes the functionality of the first one.

### Exercise 1 Requirements: Blinking Lamp

When the switch is On and a pushbutton is pressed in, the lamp blinks. When the pushbutton is released, the lamp reverts to its normal state (i.e., the lamp is on and not blinking).

For this exercise, you will need to add the following new elements:

- Labeled pushbutton object.
- Subtree of modes.
- Event triggers.

### Exercise 2 Requirements: Timed State

When the switch is on and the pushbutton is pressed in, the lamp blinks for five seconds and then returns to normal.

For this exercise, you will need to make the following changes:

- Add a timer object (a nongraphic object).
- Change an event trigger.
- Add an entry activity.

## GUIDELINES FOR COMPLETING THE EXERCISES

The guidelines will help you implement the new system requirements.

### Exercise 1: Creating a Blinking Lamp

**Requirements:** When the switch is On and a pushbutton is pressed in, the lamp blinks. When the pushbutton is released, the lamp reverts to its normal state (i.e., the lamp is on and not blinking).

First you will add a labeled pushbutton. Then you will add modes and edit the logic to make the lamp blink.

**To add a labeled pushbutton:**

- 1 In the Object Layout, add a pushbutton object to the layout.
- 2 Add the label “Blink” on top of the pushbutton. You may have to enlarge the pushbutton so that it can contain the text:



- 3 Double-click the label to open its parameter pane. Notice that “self,” i.e., the application itself, appears in the Parent box.
- 4 Make the pushbutton the parent of the label. To do so, Alt-click the pushbutton. The pushbutton’s name now appears in the Parent box.
- 5 Click Accept.

---

---

## Assigning parents to label objects

Whenever you add a label on top of an active object, you should make it a child of the active object. If the label is not a child of the active object, it will be hidden during runtime when the active object is manipulated (e.g., when the pushbutton is pressed). In addition, if a parent object is shown, hidden, or moved in the logic, the child object will do the same.

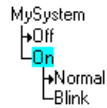
---

---

The next step is to add the modes and logic that define the system behavior.

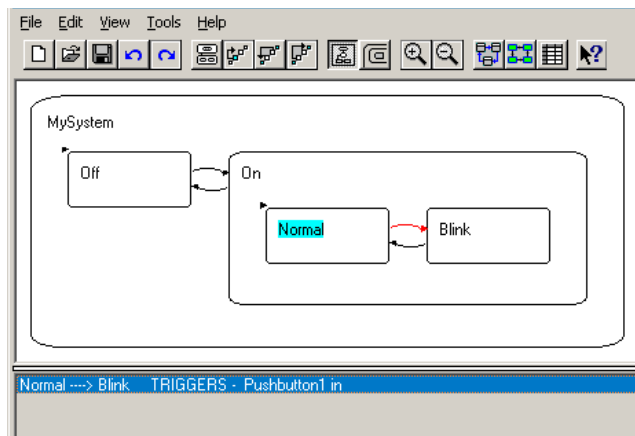
### To make the lamp blink:

- 1 In the Mode Tree, add two modes as children of On mode. The first mode will be the default mode in which the lamp is on as normal. In the sibling mode, the lamp will blink. Your mode tree should look something like:



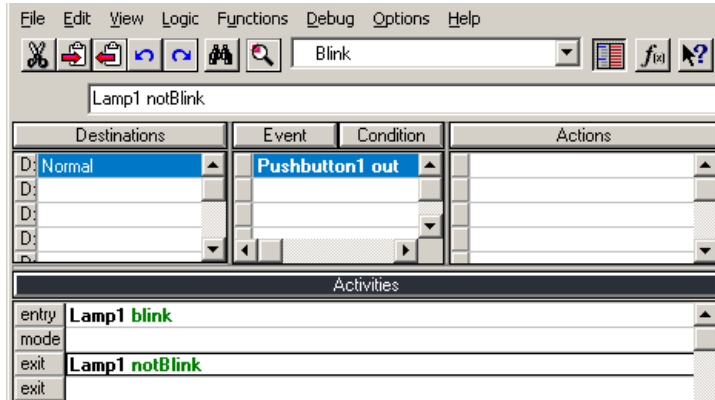
- 2 In the Logic Editor, define transitions between the two new modes so that when the pushbutton is pressed in, the lamp blinks. When the pushbutton is released, the lamp reverts to Normal mode. Use the pushbutton's *in* and *out* events to trigger the transitions.

The State Chart below shows the transitions between the new modes. The trigger of the Normal-to-Blink transition is displayed in the Logic Information pane at the bottom of the chart:

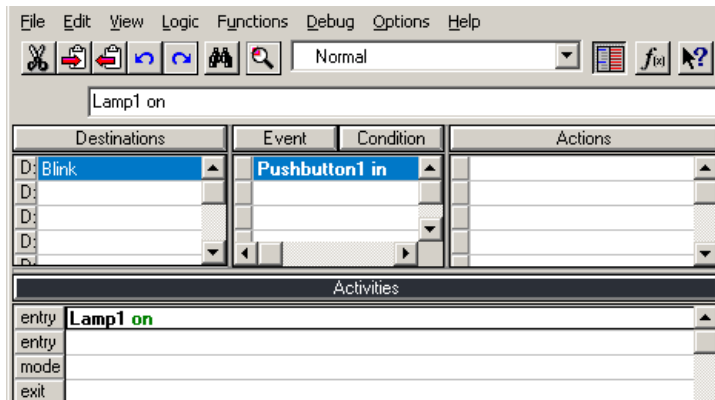




- 3 Add entry and exit activities to Blink mode, using the lamp's *blink* and *notBlink* functions:



- 4 In **On** mode click the activity line (**Lamp1 on**), then right-click and choose Cut.
- 5 In **Normal** mode, click the first line in the Activities column, right-click and choose Paste. Normal mode should look like:



Since this application is simple, you do not have to perform the last two steps for the application to run properly. However, in a more complex application in which a parent mode has children, the location of the activities can have a significant impact on how the application runs. For example, in this application you did not move the `display1.contents := 'On'` logic phrase because you want the display to appear when the application is running in both Normal and Blink modes.

Finally, you should test your application to be sure that it functions as expected.

**To test the application:**

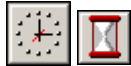
- In the Prototyper, test the application by turning the switch on and clicking the pushbutton. When you are finished, stop the Prototyper.

## Exercise 2: Adding a Timed State

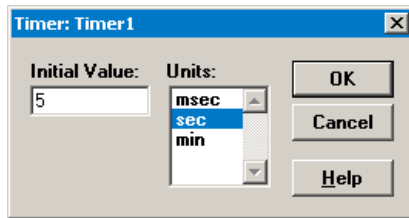
**Requirements:** When the switch is On and the pushbutton is pressed in, the lamp blinks for five seconds and then returns to normal.

First you will add the required object—a timer.

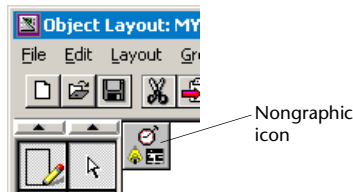
**To add a timer:**



- 1 In the Object Layout, add a timer object. The dialog box that opens when you click the timer object button is used for changing the object's default name. Click the More button; the selected object's dialog box opens.
- 2 In the Timer dialog box, set its initial value to five seconds:



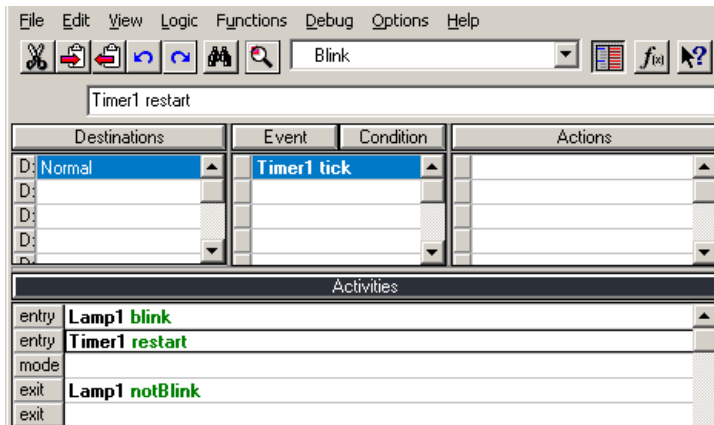
After you close the dialog boxes, notice that the nongraphic icon appears in the upper-left corner of the layout work area. This icon is for access to nongraphic object and does not appear in the runtime application.



Next, add the logic that defines the system behavior.

#### To define the timer behavior:

- 1 In the Logic Editor, replace the current trigger of the Blink-to-Normal transition (*Pushbutton1 out*), with the event that the timer generates when it reaches zero (*Timer1 tick*).
- 2 Add an entry activity to Blink mode that starts the timer (use the *restart* function):



### What's the logic?

When you click the pushbutton, the application goes into Blink mode; the lamp begins to blink and the timer begins to count down.

When the timer reaches zero, it generates the tick event and the transition back to Normal is triggered; the lamp stops blinking and stays on.

Finally, you should test the application.

#### To test the application:

- In the Prototyper, test the application by turning the switch on and then clicking the pushbutton once. Notice when the lamp stops blinking. When you are finished testing the application, stop the Prototyper.



## Glossary

<b>action</b>	Application operation that is performed on an object when a particular transition is triggered. An action is built using an object's properties and functions.
<b>active mode</b>	Mode containing the activities that are currently being executed. As an application runs in the Prototyper, you can trace active modes. <i>See also mode.</i>
<b>active object</b>	Graphic element that can be manipulated and plays an active role in an application's logic. Examples include switches and displays.
<b>activity</b>	Application operation performed on an object when a particular mode is active. An activity is built using an object's properties and functions.
<b>application</b>	Prototype or simulation of an embedded system that is designed with RapidPLUS.
<b>Application Manager</b>	The main window that is used to open other tools, set general and global application parameters, generate application reports, and hold notes about modes, objects and triggers. The window title is "Rapid: <ApplicationName>".
<b>child</b>	Mode or object that is allocated a specific parent. Every mode or object is a child of another, except for the root mode or root object.
<b>concurrent modes</b>	Modes that are always active, with all their sibling modes, when their parent is active.

---

<b>condition</b>	Logic statement that tests the status of an object or a mode. Conditions can be used alone or with events to trigger transitions.
<b>Debugger</b>	Tool used to pause the application at breakpoints, execute logic one step (one logic statement or block of logic) at a time, and examine logic in the Call Stack and Logger panes.
<b>default mode</b>	Exclusive child mode that becomes active when a default transition occurs to its parent. A default mode is marked by an arrow in the Mode Tree and State Chart.
<b>Differencing Tool</b>	Tool used to compare objects, modes, and other elements of applications; especially useful for comparing different versions of an application or project.
<b>Document Manager</b>	Tool used to generate production specification documents and acceptance test procedure documents.
<b>entry activity</b>	Activity that occurs as the mode becomes active.
<b>event</b>	Momentary change in the status of an object or mode. Events can be used alone or with conditions to trigger a transition.
<b>exclusive mode</b>	Mode that cannot be active when a sibling mode is active.
<b>exit activity</b>	Activity that occurs as the mode becomes inactive.
<b>function</b>	A command that adjusts an object, a property, or a property value.
<b>graphic object</b>	Visual object that can be manipulated by the user during runtime. Graphic objects give an application its physical appearance. Examples include switches, lamps, and displays.
<b>logic</b>	Statements that govern the behavior of the application and are built using an object's properties and functions.
<b>Logic Editor</b>	Tool in which you build an application's logic for transitions, events, conditions, actions, and activities.
<b>Logic Palette</b>	A window of the Logic Editor that provides a list of all the objects in the application, and all of the object-specific properties, functions, conditions, and events.

---

<b>mode</b>	Separate unit of functionality, or operational state, of a system. For example, “On” mode is entered when a machine is powered on. “Off” mode is entered when a machine is powered off. The mode hierarchy is created in the Mode Tree.
<b>mode activity</b>	Activity that occurs continuously while the mode is active.
<b>Mode Tree</b>	Tool in which you build the application’s hierarchical tree of modes.
<b>nongraphic object</b>	Nonvisual, behind-the-scenes object that can be manipulated by, and play an active role in, an application’s logic. Examples include timers and data stores.
<b>object</b>	Visual and nonvisual elements of an application. Objects have predefined sets of properties and functions that define the application’s logical behavior.
<b>Object Editor</b>	Tool used to edit objects’ bitmaps, pointers, and active areas.
<b>Object Layout</b>	Tool used to add objects to the application, and determine their physical layout.
<b>parameter pane</b>	Dialog box that defines the basic parameters (e.g., object name, parent object, position, and size) of a graphic object in the Object Layout. Through the parameter pane, you can access further options for the selected object.
<b>parent</b>	Mode or object that transmits certain characteristics to its child. Every object or mode in an application has a parent object or mode. The only exceptions are the root object and root mode.
<b>primitive object</b>	Graphic element used to enhance the application’s physical appearance. Examples include frames and lines. You can make a primitive object active by assigning it a name.
<b>property</b>	Characteristic of an object that describes a specific aspect of an object’s behavior. Each property holds a set of functions and can be used to define triggers and actions.
<b>Prototyper</b>	Tool used to run the application and test its behavior.
<b>root mode</b>	Highest mode in the mode tree hierarchy. It automatically bears the name of the application.
<b>sibling</b>	Object or mode that shares the same parent as another object or mode.

<b>State Chart</b>	Tool that presents the application's modes, and the transitions between them, in chart form. The State Chart shows mode hierarchy as nested shapes.
<b>Trace command</b>	Prototyper option that highlights the active mode or modes in the Mode Tree while the application runs.
<b>transition</b>	Path from a source mode to a destination mode when a specific trigger takes place.
<b>trigger</b>	Event or condition (or both) that cause a transition to take place.