

# 16. Function Libraries

## 16.1. Prerequisites and Objectives

**Before starting this chapter you should have:**

1. A knowledge of programming in C++.
2. A knowledge of component interface design.

**After completing this chapter you will have:**

1. A knowledge of how to design Function Library components.

## 16.2. Introduction

A function library is a component that provides a number of useful functions, but has little or nothing in the way of internal state. Function Library components are extremely rare for two reasons. First, there are a number of different ways to provide function libraries, most of which are much more efficient than using a component. Second, function libraries are more closely associated with functional programming than with object-oriented programming. The object-oriented components described in the preceding chapters are likely to be far more useful than a pure function library.

Nevertheless, we have found at least one application where function libraries are quite useful. This is in the implementation of wizards and common dialog boxes. Even so, these are not pure function libraries, but “information collectors.” When a function is called a dialog box, or a series of dialog boxes is displayed. The user supplies information to the dialog boxes, which is then recorded in the properties of the component. When the function completes its execution, the calling program extracts the

information from the properties. These properties are also used to provide initialization values for the dialog box controls.

### ***16.3. The Methodology***

No special methodology is used to create components of this type. The same principles apply here as would apply to any function library. One must make a collection of functions that are useful, and related in some way. There is the added dimension of having multiple instances of a common environment for all functions in the library, but this is useful only in specialized contexts. In many cases, it is more efficient to use a standard compile-time library, or a dynamic run-time library.

### ***16.4. The VDAL Common Dialogs Component***

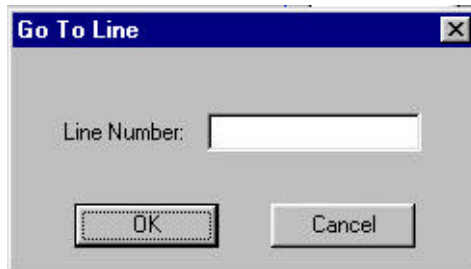
The Virtual Design Automation Laboratory (VDAL) Common Dialogs component was created to provide a set of dialog boxes for a set of web-based design automation applications. Several of these applications require the user to enter circuit descriptions and other data in text format. A commercial text-editing component was used to support these operations. The VDAL Common Dialogs component contains a set of dialog boxes that were specially designed for use with this component. Some of these boxes are similar to those provided by the Windows Operating System common dialogs. These dialogs are shown in Figure 16.1 through Figure 16.5.



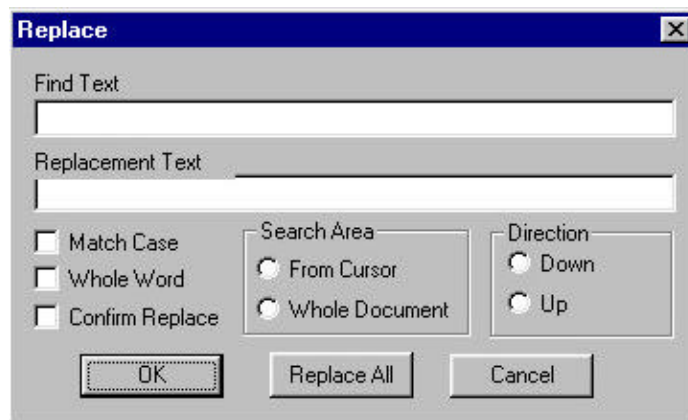
**Figure 16.1. The Confirm Replace Dialog.**



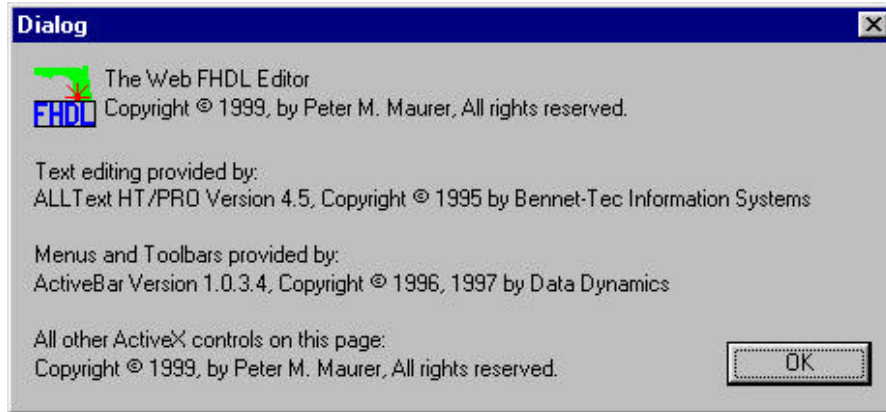
**Figure 16.2. The Find Dialog.**



**Figure 16.3. The Go To Dialog.**



**Figure 16.4. The Replace Dialog.**



**Figure 16.5. The About Dialog.**

Each of the dialogs has an associated method that is used to display the dialog, and an associated set of properties to hold the values of the text boxes and other controls on the dialog. The descriptions of these functions are given in Figure 16.6 through Figure 16.10.

Method Description			
<b>Name</b>	ShowConfirm		
<b>Return Value</b>	Void		
<b>Description</b>	Shows the Confirm Replace Dialog.		
<b>Arguments</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
Void			

**Figure 16.6. The ShowConfirm Method.**

Method Description			
<b>Name</b>	ShowFind		
<b>Return Value</b>	Void		
<b>Description</b>	Shows the Find Text Dialog.		
<b>Arguments</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
Void			

**Figure 16.7. The ShowFind Method.**

Method Description			
<b>Name</b>	ShowLineNumber		
<b>Return Value</b>	Void		
<b>Description</b>	Shows the Go To Line Dialog.		
<b>Arguments</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
Void			

**Figure 16.8. The ShowLineNumber Method.**

Method Description			
<b>Name</b>	ShowReplace		
<b>Return Value</b>	Void		
<b>Description</b>	Shows the Replace Text Dialog.		
<b>Arguments</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
Void			

**Figure 16.9. The ShowReplace Method.**

Method Description			
<b>Name</b>	ShowWebFHDL		
<b>Return Value</b>	Void		
<b>Description</b>	Shows the About Dialog.		
<b>Arguments</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
Void			

**Figure 16.10. The ShowWebFHDL Method.**

The component properties are used to supply initial values for the dialog boxes, and to retrieve the values entered by the user. These properties are listed in Figure 16.11.

<b>Property Design Table</b>		
<b>Name</b>	<b>Type</b>	<b>Function</b>
<b>LineNumber</b>	Long Integer	Go To Dialog text-box value.
<b>FindText</b>	String	Find/Replace dialog target text.
<b>ReplacementText</b>	String	Replace dialog replacement text
<b>MatchCase</b>	Boolean	Find/Replace Match Case check box
<b>SearchArea</b>	Enumerated From Cursor = 0 Whole Document = 1	Find/Replace Search Area radio buttons
<b>SearchDirection</b>	Enumerated Up = 0 Down = 1	Find/Replace Search Direction radio buttons.
<b>WholeWord</b>	Boolean	Find/Replace Whole Word check box
<b>ConfirmReplace</b>	Boolean	Replace Confirm replace check box
<b>UserCancelled</b>	Boolean	Set when cancel button is clicked on any dialog.
<b>SearchAll</b>	Boolean	Set when Replace All button is clicked on the Replace dialog.
<b>LastFind</b>	Enumerated None=0 Find=1 Replace=2	Used to support “Repeat Last Find” operations. Keeps track of the last find/replace operation.
<b>ConfirmYes</b>	Boolean	Set when Yes is clicked on Confirm Replace dialog.
<b>ConfirmNo</b>	Boolean	Set when No is clicked on Confirm Replace dialog.
<b>ConfirmCancel</b>	Boolean	Set when Cancel is clicked on Confirm Replace dialog.

**Figure 16.11. The VDAL Common Dialogs Property Design Table.**

The implementation of the VDAL Common Dialogs Component is straightforward, it will not be presented here. The source code for the component is available on the CD. The component has additional features that are not described here because they do not fit the paradigm of a function library.

## **16.5. Conclusion**

Because function libraries do not fit well with modern Object Oriented Design techniques, and because function libraries are available in many other forms, they are the

least commonly encountered type of component. They do, however, provide a means of encapsulating and distributing sets of similar functions. Although function access is less efficient than with other types of function libraries, component function libraries do not require header files, and may be somewhat easier to use for that reason.

### **16.6. Exercise.**

1. Make a list of your favorite functions, combine them into a component function library.