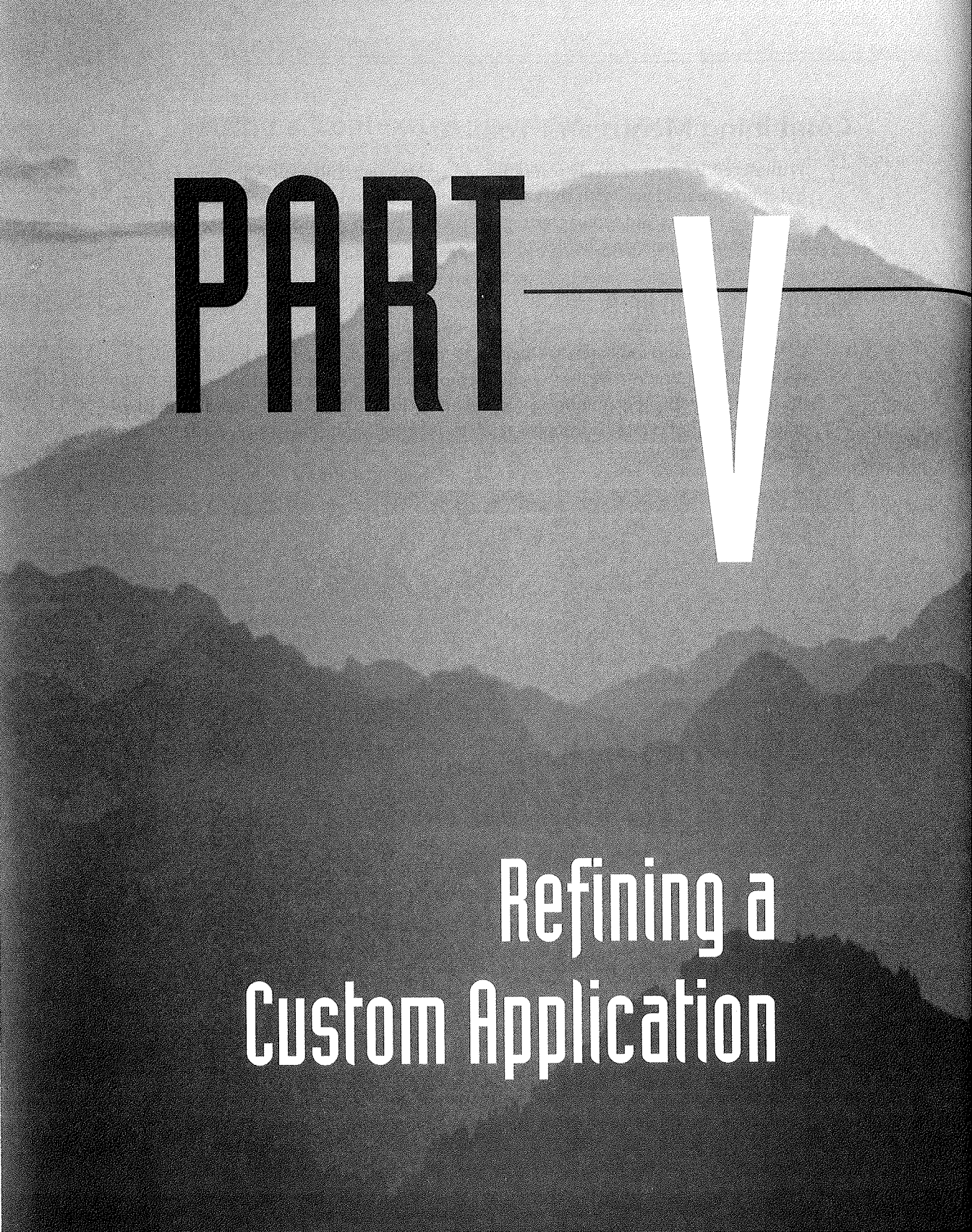
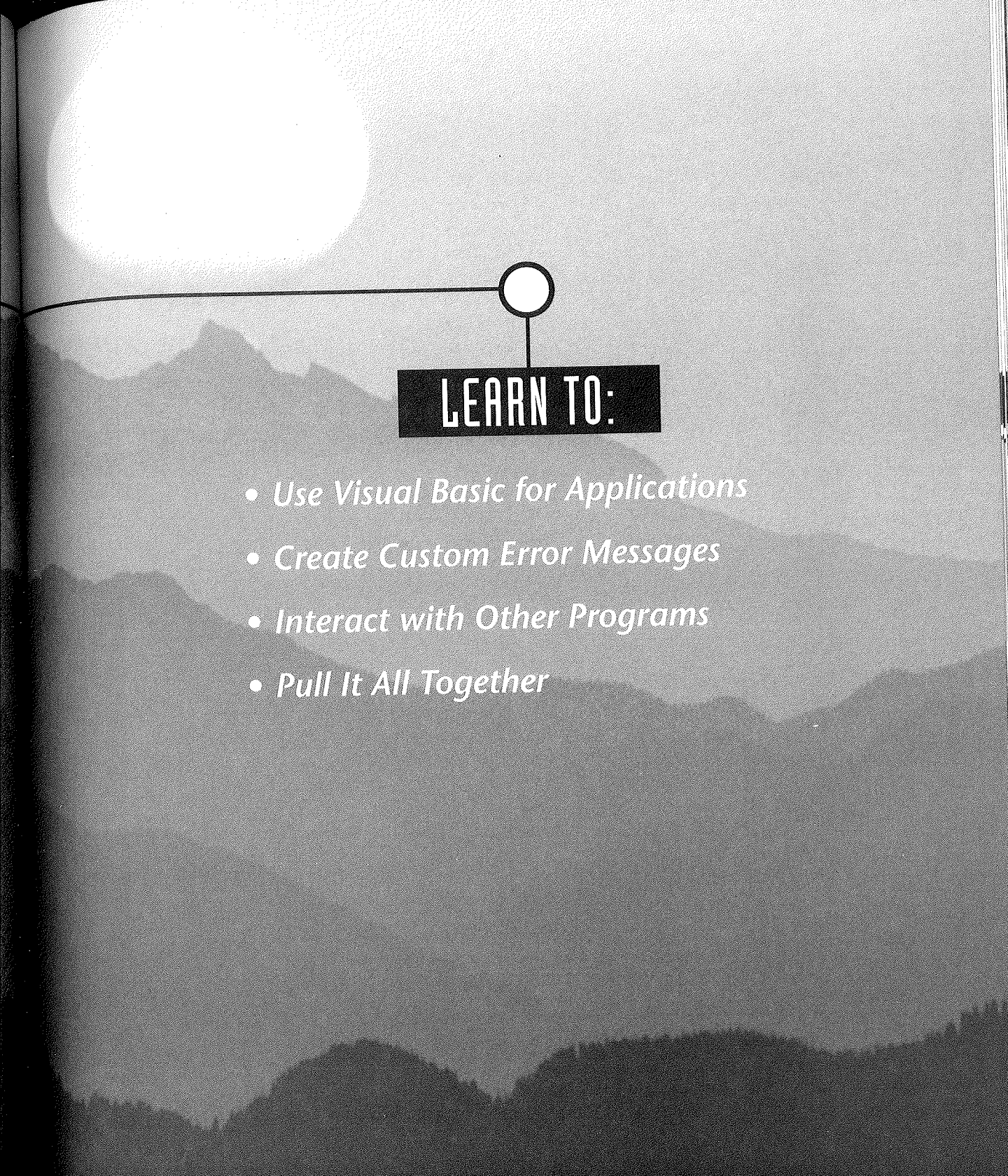
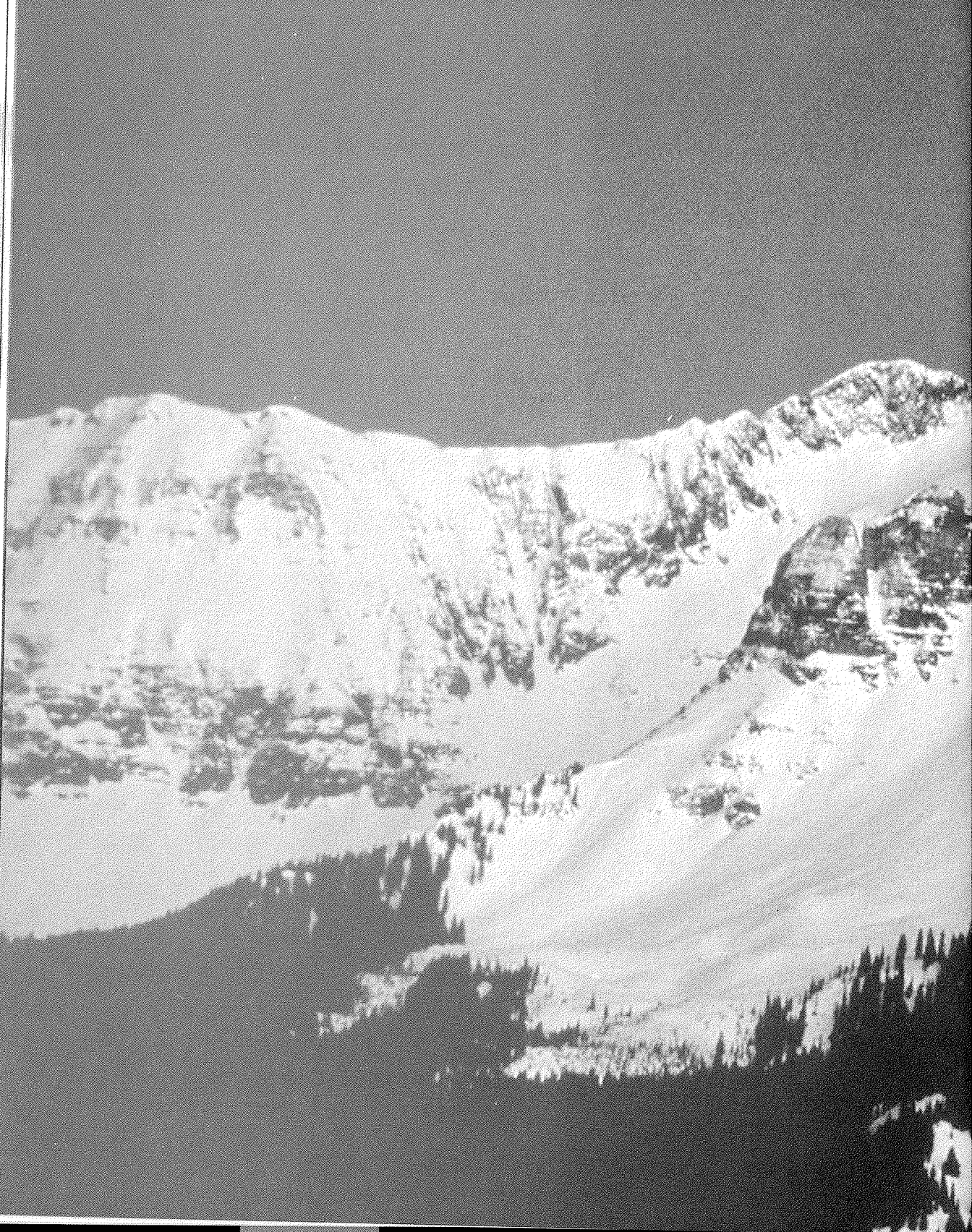
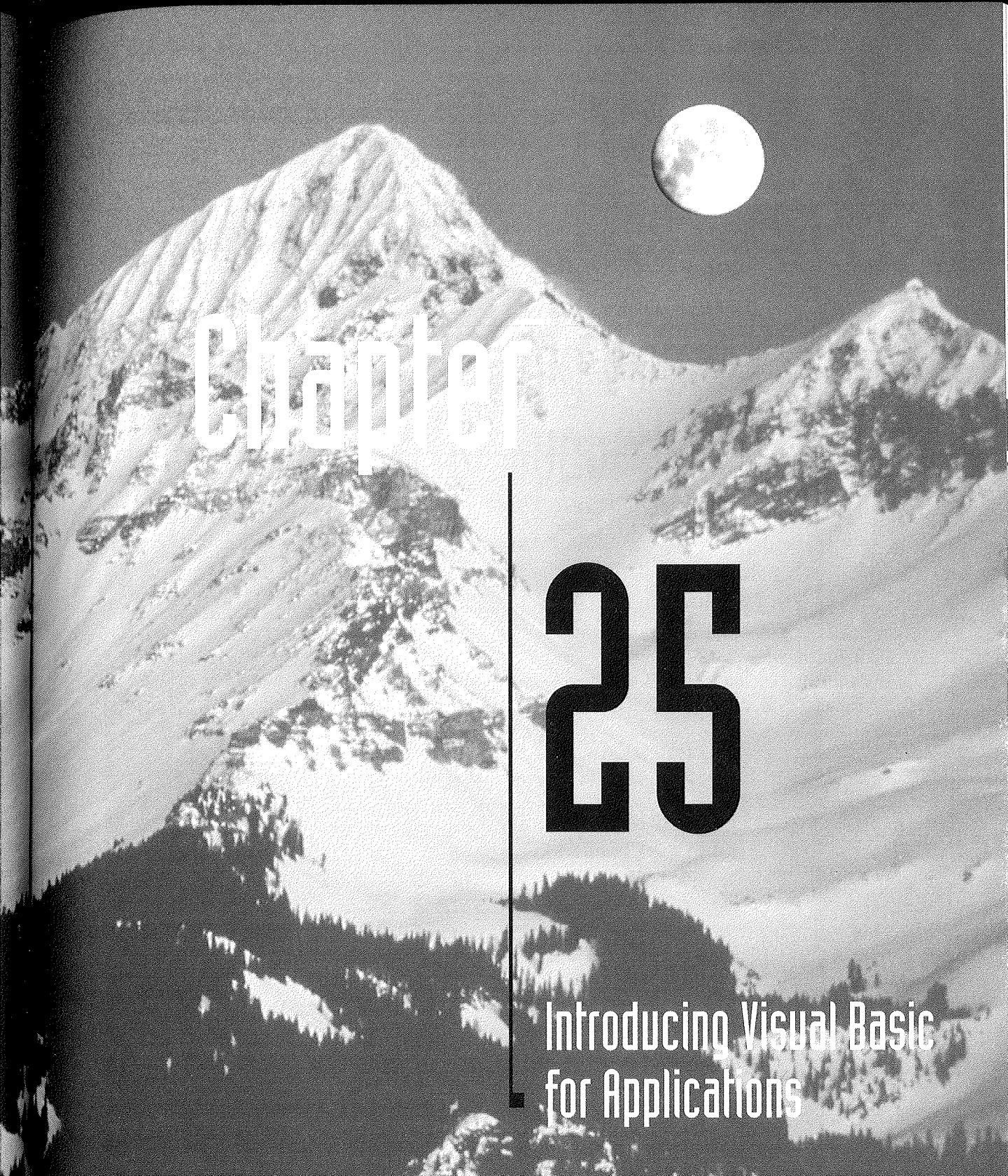
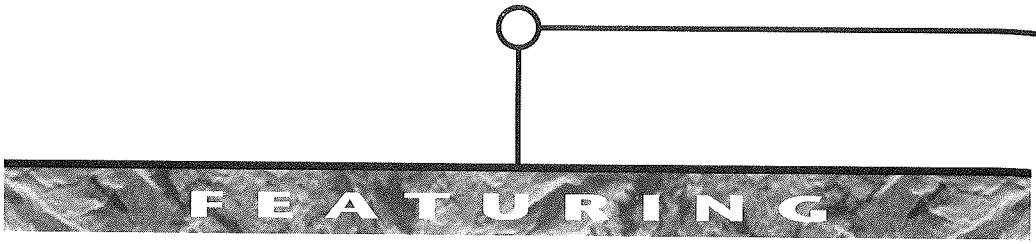

PART V
Refining a
Custom Application


LEARN TO:
• Use Visual Basic for Applications
• Create Custom Error Messages
• Interact with Other Programs
• Pull It All Together




Chapter
25
Introducing Visual Basic
for Applications



Introducing Visual Basic for Applications

If you've experimented with hyperlinks and macros, you know that you can use them to automate lots of different database tasks. However, there are some cases where a macro or a hyperlink just isn't enough to do the task at hand. For example, if you want to prompt someone for the name of a file to import instead of using a macro to import a specific file, you need to don your propeller beanie and start programming.

The programming language for Access 97 for Windows is Visual Basic for Applications. This programming language is shared by Visual Basic, Access, and the other Microsoft Office applications.

Why Use VBA?

Access provides macros that are easy to use and properties that can be set to run macros. Therefore, a legitimate question is why you would need to program in Visual Basic at all. In Chapters 20 through 24, for example, we showed you how to accomplish lots of tasks with macros. Can't you do everything you need with a macro?

The answer to that question is both yes and no. Yes, many Access users will never have to go beyond macros to accomplish what they need to do. However, no, you can't

do everything with macros. Macros have some bad habits that prevent them from being ideal for every purpose. In general, you should use macros under the following conditions:

- When your focus is simplicity. Macros provide an extremely visual programming style. You do not have to learn syntax. You simply select from among the options provided and the action you desire is programmed.
- When you want to create a toolbar or menu. Visual Basic does not provide an alternative way of creating these objects.
- When you want to undertake an action at the time the database opens using the AutoExec macro. In this circumstance, you must use the macro.
- When the built-in error messages from Access are insufficient in case of trouble.

You should use Visual Basic for Applications when you have these goals in mind:

- When your focus is ease of maintenance. Unlike macros, Visual Basic procedures can be a part of the forms or reports that contain them. When you copy a form or report from one database to another, all of the Visual Basic procedures stored with the object are copied with it.
- When Access does not provide a function that you need. If no built-in function can perform a particular calculation, you can write your own function in Visual Basic.
- When you want to respond to error messages creatively. Using Visual Basic, you can create your own error messages or take an action that corrects the error without user intervention.
- When you need to check the state of the system. Macros will allow you to run another application, but they don't provide access to system level information. You cannot check, for instance, whether a file exists using a macro. Visual Basic, however, allows you access to system level information and actions.
- When you need to work with records one at a time. Macros perform actions on sets of records. Visual Basic allows you to step through records and perform actions on each single record while it is in focus.
- When you need to pass arguments between procedures. You can set the initial arguments for macros, but you can't change them while the macro runs. You can make such changes, or use variables for arguments, using Visual Basic.

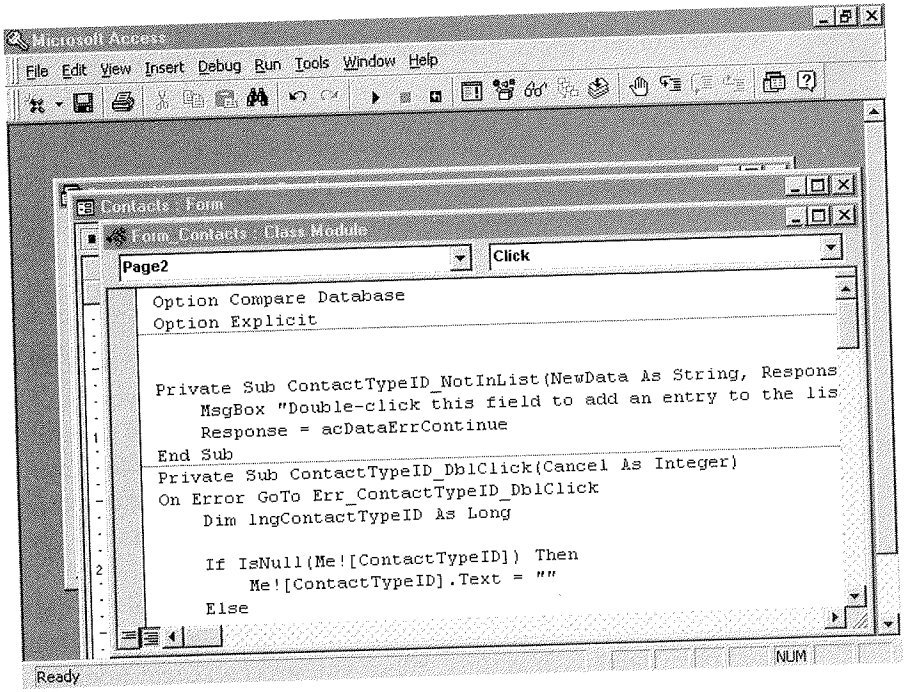
What Is the Shape of Visual Basic?

So where does Visual Basic for Applications hide within the overall structure of Access? How do you use all of its wonderful features? To find Visual Basic, you need only look behind the Code button on the Form Design or Report Design toolbar. Clicking on this

button opens the code window for the form or report in focus (see Figure 25.1). Using this window, you write your custom procedures and functions.

The code that you insert into a form or report applies only to that form or report or to the objects contained within it. Although you can call these procedures from any other procedure included in your database, the code still applies only to the form or report and the objects contained therein. For code that needs to be accessible to any object in your database, create a module using the New button on the Modules tab in the database window. Place your global procedures and functions in this code window.

FIGURE 25.1
The code window for the Contacts form in the Contact Management database generated by the Database Wizard.



Visual Basic and Objects

Visual Basic is an object-oriented programming environment. An *object* has a set of procedures and data associated with it. Some objects also have a visual representation, though others are available only in Visual Basic code. A good example of an object is a form. You draw the form on the screen; it is associated with a set of procedures, and data is displayed in the form.

Given this definition, reports are also objects, as are any of the controls that you use to build both forms and reports. (Technically, tables and queries can also be considered objects.) Each object has associated with it *properties,* which govern the appearance and behavior of the object. Each object also has a set of *methods* defined for it, which are actions the object can take. Some objects, including forms, reports, and controls, also respond to a set of *events*.

You are already familiar with properties for objects. They are the same properties that you have been setting for forms, reports, and controls as you have worked with databases throughout this book. Methods should also be familiar, since you have used the methods (such as Recalc, GoToPage, and SetFocus) associated with forms. Events are very similar to event properties into which you could insert a macro; however, you can write code that responds to each event, instead of relying on predefined macro actions.

Visual Basic and Events

Visual Basic uses event procedures to respond to events. By default, VBA never does anything when an event occurs. If you want an object to respond to an event, say a button click or the activation of a form, you have to write the code for the procedure. Your code overrides the default action, allowing the action you define to take place in response to the event.

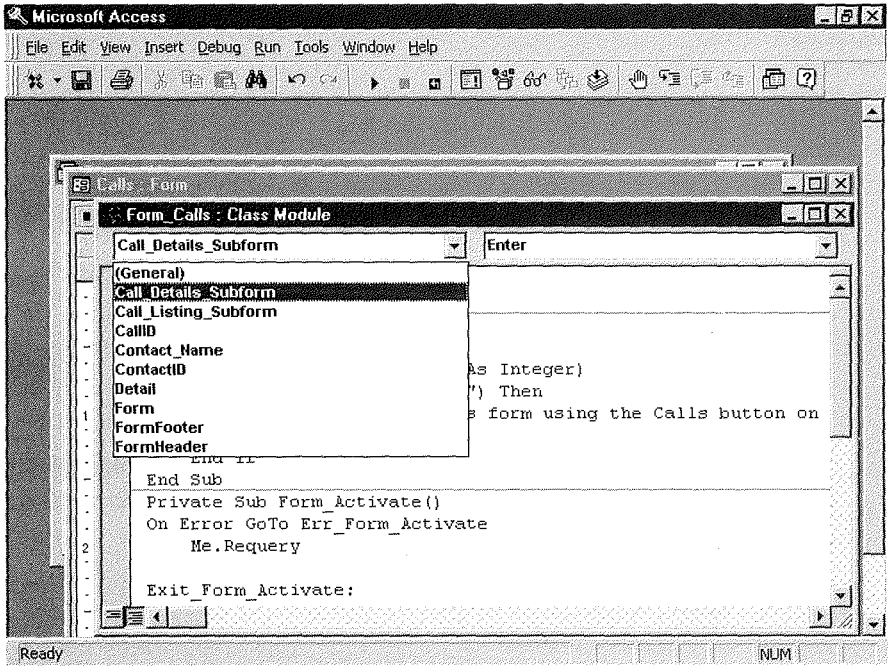
Each object defines the events to which it responds. If you look in the code window for the Calls form in the Contact Management database, you can get a sense of how these events are made available to database developers (see Figure 25.2). The form contains several objects. The drop-down list box in the upper-left corner of the window lists all the objects associated with the form. The (General) object represents code that affects all the objects in the list. Each object, including the form itself, is designated by a name, which is one of an object's properties.

The drop-down list box in the upper-right corner of the code window lists all the events to which an object responds, as shown in Figure 25.3. Each event is given a descriptive name such as Click. When you select an event from the list, the code window displays the frame of the event procedure, consisting of a Private Sub Calls_Event (Arguments) statement and an End Sub statement. These two statements are required to frame the code responding to the event. They frame the block of code known as an *event procedure,* the set of statements that causes an action in response to an event.

Visual Basic and Statements

In order to build an event procedure, you have to use Visual Basic statements and functions. If you are not familiar with programming, you may feel overwhelmed by the number

FIGURE 25.2
The list of objects in the code window for the Calls form in the Contact Management database.



of statements available to you. Visual Basic provides flow control statements, such as If...Then...Else..., which govern the order in which other statements execute. (The two statements that frame an event procedure are, in fact, flow control statements.) VBA also provides statements that cause an action to take place, such as Beep (which causes the system to send a beep to the speaker) or ChDir (which changes the current folder or directory). In total, VBA uses 81 such statements. (Help contains a complete reference for them.)

The next chapter covers flow control in more detail.

In addition, you can write statements that set an object property, assign a value to a variable, use a function, or use a method. To assign a value to an object property, you write a statement in the following form:

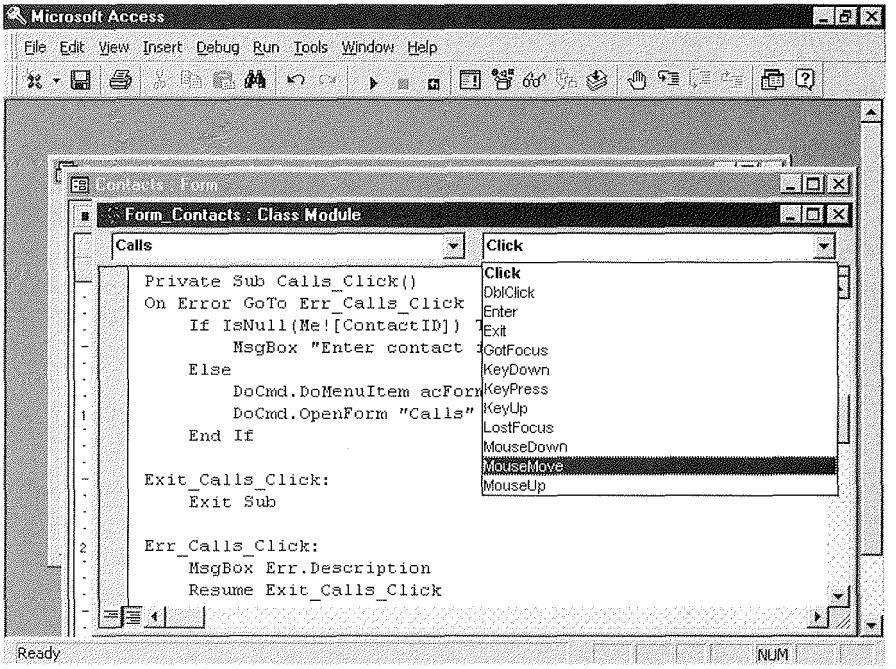
```
Set Object.Property = Value
```

To assign a value to a variable, you use a statement of this form:

```
Set Variable = Value
```

FIGURE 25.3

The Proc list and the event procedure for the Click event associated with the Calls object on the Contacts form.



To use a function, first you select a function that calculates something or takes an action. Typically, functions return the value they calculate or a code indicating success for your program to use. As a result, you often assign the value of a function to a variable for future use. The following statement, for example, uses the Date function to collect the system date and stores it in a variable named Today:

```
Today = Date()
```

To use a method, you use the same kind of syntax that you use to set a property—but without an equal sign. You combine the name of the object with the method using dot notation to form the statement. One of the most useful object names is Me, which names the object currently in focus. The following statement causes the current object to redraw itself using the Refresh method associated with it:

```
Me.Refresh
```

Visual Basic and Variables

We mentioned variables just in passing. A *variable* is a name you give to Access. Access sets up an area of memory for storing items and gives it that name. A variable is said to have a *type*, which describes the nature of the item that can be stored in it. By default, Visual Basic creates variables of the *variant* type, which means that anything can be stored in it. To create a variable name, you simply use it. Or you can explicitly name it in a statement called a *declaration* at the beginning of your procedure:

```
Dim strDialStr As String
```

This statement says to create, or "dimension" (hence the Dim statement), a variable named strDialStr as type String. StrDialStr can therefore contain only strings of characters. Attempting to assign data of some other type to it will cause an error. Visual Basic supports the following variable types:

- **Integer** Whole numbers between –32,768 and 32,768
- **Long** Whole numbers between –2,147,483,648 and 2,147,483,647
- **String** Text up to approximately 65,500 characters in length
- **Currency** Numbers with up to four decimal places between -955,337,203,685.5808 and 955,337,203,685.5807
- **Single** Real numbers in the range $\pm 1.40 \times 10^{-45}$ to $\pm 3.40 \times 10^{38}$
- **Double** Real numbers in the range $\pm 4.94 (10^{-324}$ to $\pm 1.79 \times 10^{308}$
- **Variant** Can contain any of the preceding data types

When you name a variable, you should include the first three letters of its type in its name, as shown above. The reason for including the type in the name is that you can always tell what can be stored in a variable by looking at its name. Late at night on a long project, or coming back to maintain code after two months away from it, you will appreciate this convention.

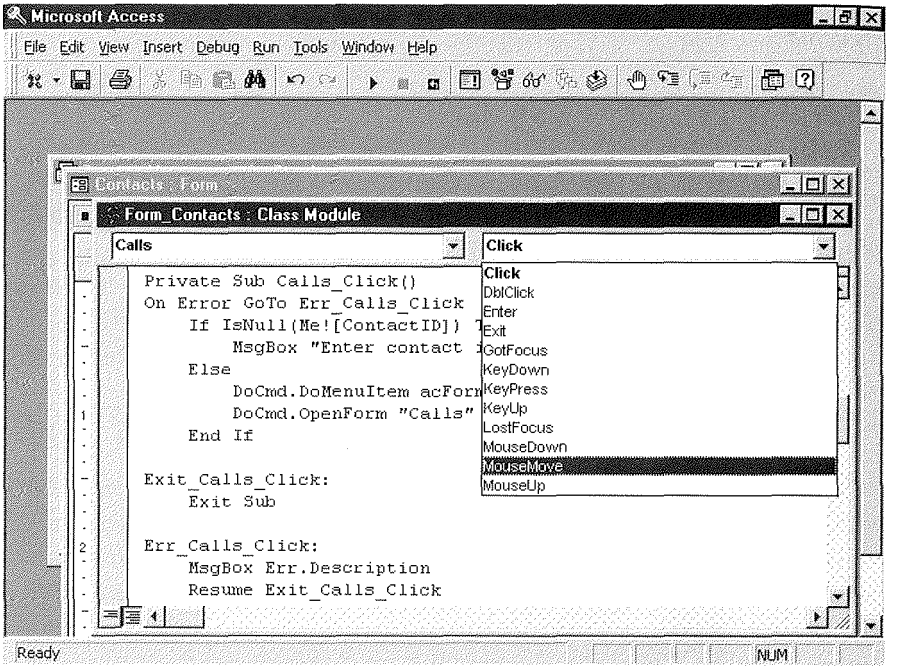
NOTE To make a variable available to all procedures and functions in your application, place it in a global code module and precede it with the keyword Public.

Visual Basic and Procedures

Event procedures are not the only procedures you can write in Visual Basic. You can also write a procedure as a part of a global code module or at the General level of any form or report, and you can call that procedure any time you need to perform the action undertaken by the procedure. The Contact Management database uses this type

FIGURE 25.3

The Proc list and the event procedure for the Click event associated with the Calls object on the Contacts form.



To use a function, first you select a function that calculates something or takes an action. Typically, functions return the value they calculate or a code indicating success for your program to use. As a result, you often assign the value of a function to a variable for future use. The following statement, for example, uses the Date function to collect the system date and stores it in a variable named Today:

```
Today = Date()
```

To use a method, you use the same kind of syntax that you use to set a property—but without an equal sign. You combine the name of the object with the method using dot notation to form the statement. One of the most useful object names is Me, which names the object currently in focus. The following statement causes the current object to redraw itself using the Refresh method associated with it:

```
Me.Refresh
```

Visual Basic and Variables

We mentioned variables just in passing. A *variable* is a name you give to Access. Access sets up an area of memory for storing items and gives it that name. A variable is said to have a *type*, which describes the nature of the item that can be stored in it. By default, Visual Basic creates variables of the *variant* type, which means that anything can be stored in it. To create a variable name, you simply use it. Or you can explicitly name it in a statement called a *declaration* at the beginning of your procedure:

```
Dim strDialStr As String
```

This statement says to create, or "dimension" (hence the Dim statement), a variable named strDialStr as type String. StrDialStr can therefore contain only strings of characters. Attempting to assign data of some other type to it will cause an error. Visual Basic supports the following variable types:

- **Integer** Whole numbers between –32,768 and 32,768
- **Long** Whole numbers between –2,147,483,648 and 2,147,483,647
- **String** Text up to approximately 65,500 characters in length
- **Currency** Numbers with up to four decimal places between -955,337,203,685.5808 and 955,337,203,685.5807
- **Single** Real numbers in the range $\pm 1.40 \times 10^{-45}$ to $\pm 3.40 \times 10^{38}$
- **Double** Real numbers in the range $\pm 4.94 (10^{-324}$ to $\pm 1.79 \times 10^{308}$
- **Variant** Can contain any of the preceding data types

When you name a variable, you should include the first three letters of its type in its name, as shown above. The reason for including the type in the name is that you can always tell what can be stored in a variable by looking at its name. Late at night on a long project, or coming back to maintain code after two months away from it, you will appreciate this convention.

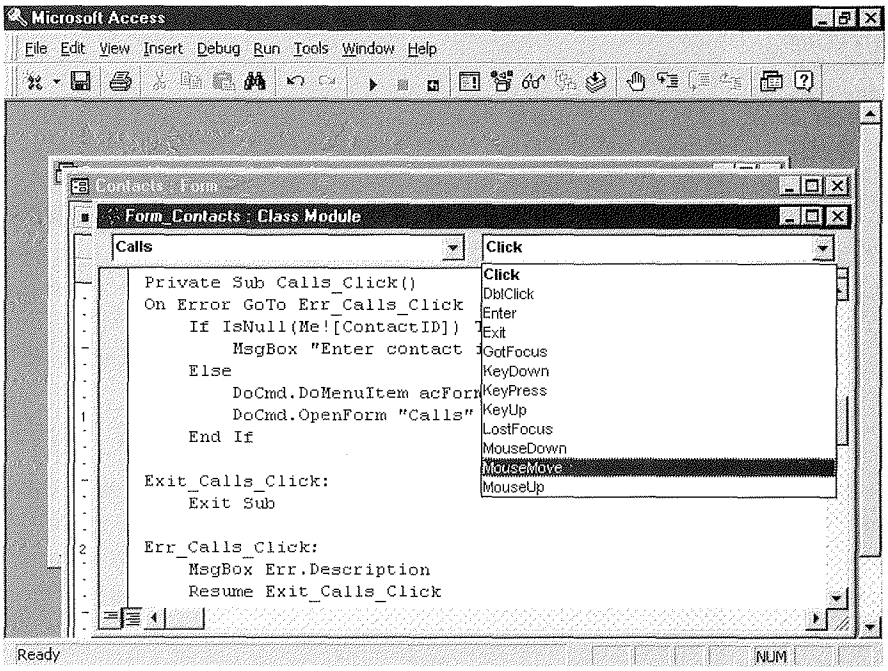
NOTE To make a variable available to all procedures and functions in your application, place it in a global code module and precede it with the keyword Public.

Visual Basic and Procedures

Event procedures are not the only procedures you can write in Visual Basic. You can also write a procedure as a part of a global code module or at the General level of any form or report, and you can call that procedure any time you need to perform the action undertaken by the procedure. The Contact Management database uses this type

FIGURE 25.3

The Proc list and the event procedure for the Click event associated with the Calls object on the Contacts form.



To use a function, first you select a function that calculates something or takes an action. Typically, functions return the value they calculate or a code indicating success for your program to use. As a result, you often assign the value of a function to a variable for future use. The following statement, for example, uses the Date function to collect the system date and stores it in a variable named Today:

```
Today = Date()
```

To use a method, you use the same kind of syntax that you use to set a property—but without an equal sign. You combine the name of the object with the method using dot notation to form the statement. One of the most useful object names is Me, which names the object currently in focus. The following statement causes the current object to redraw itself using the Refresh method associated with it:

```
Me.Refresh
```

Visual Basic and Variables

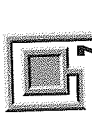
We mentioned variables just in passing. A *variable* is a name you give to Access. Access sets up an area of memory for storing items and gives it that name. A variable is said to have a *type*, which describes the nature of the item that can be stored in it. By default, Visual Basic creates variables of the *variant* type, which means that anything can be stored in it. To create a variable name, you simply use it. Or you can explicitly name it in a statement called a *declaration* at the beginning of your procedure:

```
Dim strDialStr As String
```

This statement says to create, or "dimension" (hence the Dim statement), a variable named strDialStr as type String. StrDialStr can therefore contain only strings of characters. Attempting to assign data of some other type to it will cause an error. Visual Basic supports the following variable types:

- **Integer** Whole numbers between –32,768 and 32,768
- **Long** Whole numbers between –2,147,483,648 and 2,147,483,647
- **String** Text up to approximately 65,500 characters in length
- **Currency** Numbers with up to four decimal places between -955,337,203,685.5808 and 955,337,203,685.5807
- **Single** Real numbers in the range $\pm 1.40 \times 10^{-45}$ to $\pm 3.40 \times 10^{38}$
- **Double** Real numbers in the range $\pm 4.94 (10^{-324}$ to $\pm 1.79 \times 10^{308}$
- **Variant** Can contain any of the preceding data types

When you name a variable, you should include the first three letters of its type in its name, as shown above. The reason for including the type in the name is that you can always tell what can be stored in a variable by looking at its name. Late at night on a long project, or coming back to maintain code after two months away from it, you will appreciate this convention.



NOTE To make a variable available to all procedures and functions in your application, place it in a global code module and precede it with the keyword Public.

Visual Basic and Procedures

Event procedures are not the only procedures you can write in Visual Basic. You can also write a procedure as a part of a global code module or at the General level of any form or report, and you can call that procedure any time you need to perform the action undertaken by the procedure. The Contact Management database uses this type

of procedure to handle the button clicks in the switchboard. At the General level in the Switchboard form is the procedure HandleButtonClick. This procedure receives the button number as its argument and takes action based on the button number received. One function can therefore service several buttons. The OnClick property of each button is set equal to the name of this procedure. When a click takes place, the procedure is called to perform its task.

You need to remember two things about event procedures. First, event procedures cannot return values for the rest of the program to use. You cannot set a variable equal to the value of the event procedure. Second, when a procedure is a part of a form or report, it has reference only to that form or report. Place your procedures in global code modules if you want to be able to use them anywhere in your database application and precede them with the keyword Public, rather than Private.

Visual Basic and Functions

Functions are procedures that return a value for use by other statements in the program. The return value is set using the function name as a variable, as shown in the following function generated by the Database Wizard for the Contact Management database:

```
Function IsLoaded(ByVal strFormName As String) As
Integer
 ' Returns True if the specified form is open in
 Form view
 ' or Datasheet view.

Const conObjStateClosed = 0
Const conDesignView = 0

If SysCmd(acSysCmdGetObjectState, acForm,
 strFormName) <> conObjStateClosed Then
 If Forms(strFormName).CurrentView <>
 conDesignView _
 Then
 IsLoaded = ' Set the return value
 ' using
 ' the function name as a
 ' variable.
 End If
End If

End Function
```

Don't worry about the statements that seem like Greek in this function right now. Wait until you have some experience with Visual Basic before you try to start interpreting them. The main thing to remember about building a function is to set the return value and to use the Function and End Function statements to frame it.

NOTE

Both functions and subs can take *arguments*, variables whose values are made available to the function for use. These variables are named in the parentheses following the function or subroutine name. When you name them, you use the As keyword to name their type, just as you do when declaring a variable. If you do not want the function or procedure to modify the value stored in the variable, precede the name of the variable with ByVal.

Building a Sample Procedure

Speaking of getting some experience with Visual Basic, here's your chance. In this section, we build a real VBA procedure! While this procedure will be simple, it will give you the basics. The next two chapters undertake some more complex actions. (Remember, we have three chapters to get you started programming with a language that people write whole books about. Be patient, give yourself some time, and study the Help files and sample code. You'll be an expert in no time!)

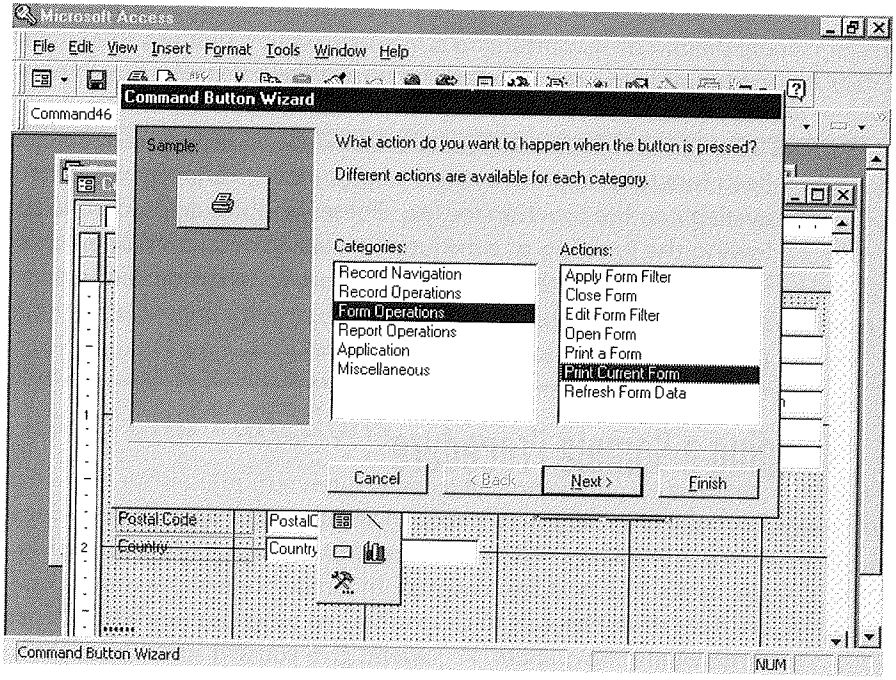
Having read the preceding sections, you have enough background to understand how a procedure is built. We have a surprise for you about how Access builds procedures. In many cases, you don't have a great deal of code to write because a Wizard writes it for you. (Convenient, no?) As an example, we are going to add a command button that prints a copy of the data entry form we built for the database used by our client who runs a nonprofit corporation. You can practice on any database form that you want. Visual Basic works the same way in all forms.

To add the button, take these steps:

1. Open the form in design view.
2. Make sure the Control Wizards button is "in."
3. Select the toolbox button that draws the command button.
4. Drag with your mouse to draw the button on the form. After a brief pause the Command Button Wizard appears (see Figure 25.4).
5. Select the class of action you want to take in the Categories list. Since we want to print a form, select Form Operations. Select the specific action in the Actions list, in this case, Print Current Form. Click on the Next button.

FIGURE 25.4

The first page of the Command Button Wizard allows you to select the type of action the event procedure performs.



6. Select whether you want a picture or text to appear on the button face using the option buttons provided. If you choose text, enter the text as you want it to appear. If you choose a picture, you can use the browse button to select a graphic. Click on the Next button.
7. Enter a meaningful name for your button in the text box on the last Wizard page. Make the name a mnemonic for the button's function. Then click on the Finish button.

When the Wizard finishes, it has built the click event procedure for you. You can see the procedure by right-clicking on the button and selecting Build Event from the menu that appears. The procedure created appears below:

```
Private Sub Print_Form_Click()
On Error GoTo Err_Print_Form_Click


 DoCmd.PrintOut
Exit_Print_Form_Click:
 Exit Sub
```

```
Err_Print_Form_Click:
 MsgBox Err.Description
 Resume Exit_Print_Form_Click

End Sub
```

Congratulations! You've just programmed your first procedure, and using excellent programming form we might add. This procedure is very straightforward. The first statement turns on error handling (more about that in the next chapter). The second statement uses the PrintOut method of the DoCmd object to print the form. (The DoCmd object contains all the actions you can invoke using macros.) The next program line is a *label*, which is a string of text that carries out no action but can serve as a jump destination. Labels end in colons. The line following the colon causes the procedure to end. The remaining three lines are a label and error-handling code. (Again, more about that in the next chapter.)

The greatest thing about writing procedures for common objects in Access is that you have a Wizard for each one. You do not have to be an expert programmer to get lots of work done in Visual Basic.

Converting Macros to Visual Basic

You can also convert macros you already have written into Visual Basic code. To convert macros associated with a form or report, open the object in design view and select Convert Macros to Visual Basic from the Macros submenu on the Tools menu.

To convert global macros, click on the Macros tab in the Database dialog box. Then click on the macro you want to convert. Open the File menu, select Save As/Export, and click on the Save As Visual Basic Module option button in the Save As dialog box. Then click on the OK button.

Learning More about Visual Basic for Applications

For more information about how to use the features of Visual Basic, take a look at these sources:

- Help on *Visual Basic* from the Office Assistant. You'll find several topics including instructions on how to obtain the *Microsoft Office 97/Visual Basic Programmer's Guide.*
- Any of the excellent books available on using Visual Basic for Applications in Access, including the *Microsoft Access 97 Developer's Handbook* from Sybex.

- The Visual Basic documentation that comes as a part of the Microsoft Developer Network CD-ROM product. All the latest documentation on Visual Basic comes on this CD every quarter.
- The code that the Database Wizard generates whenever it creates a database of a specific type. You can learn a lot about how Microsoft programmers use Visual Basic by studying these examples.
- Two magazines, the *Visual Basic Programmer's Journal* and the *VB Tech Journal*, both of which bring you monthly discussions of how to use Visual Basic to accomplish specific tasks.
- The *Access/Visual Basic Advisor*, a newsletter from Advisor Publications, or Pinnacle Publication's *Smart Access* and *Visual Basic Developer*.

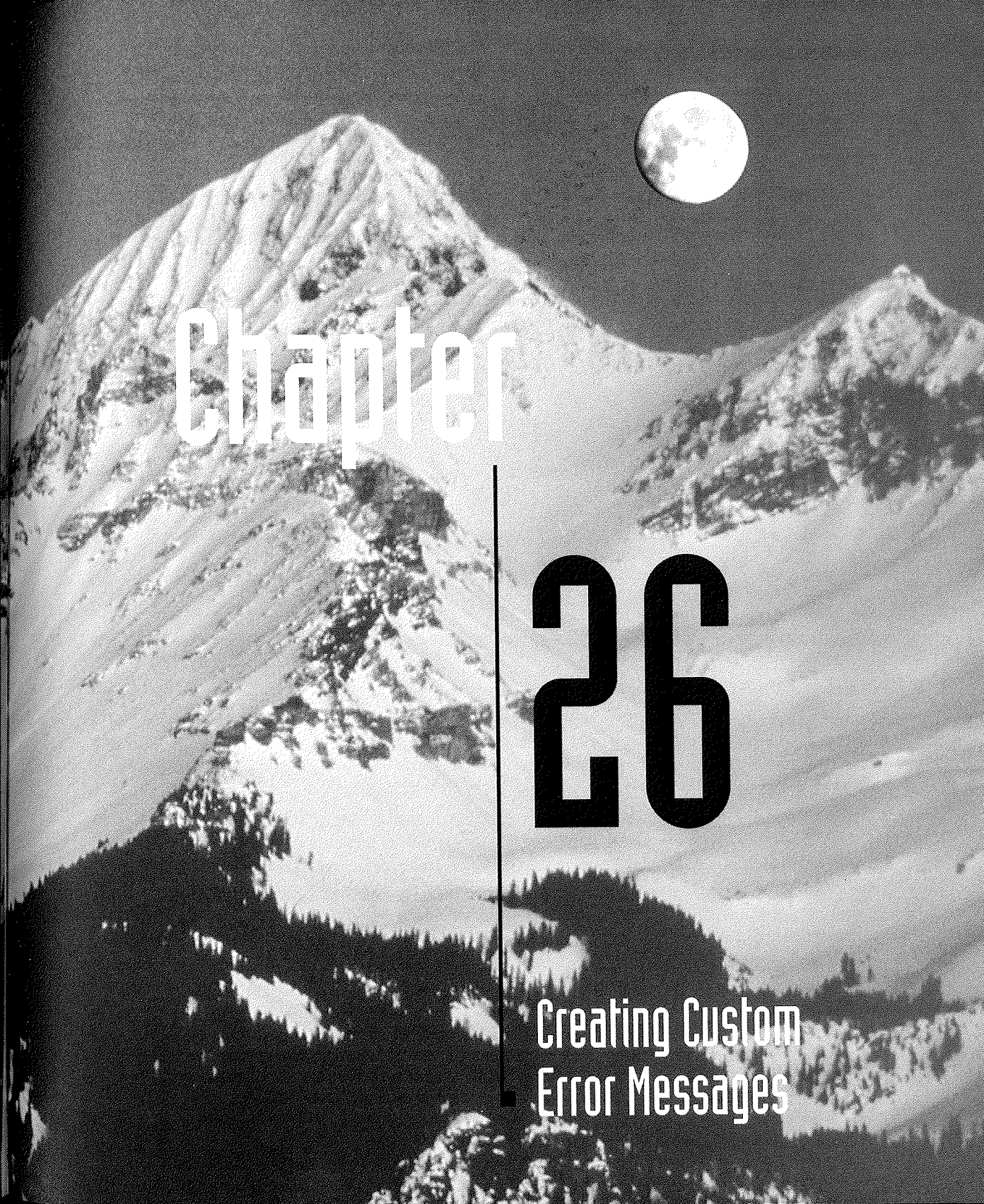
Where to Go from Here

Having introduced you to Visual Basic in this chapter, in the next two chapters we show you how to do some interesting work with this programming language. Chapter 26 explains how to create your own error messages, and Chapter 27 explains how to use OLE Automation to control other programs.

What's New in the Access Zoo?

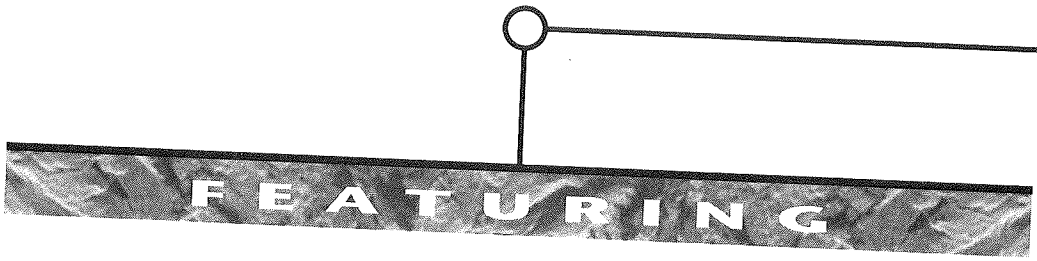
You've just taken a look at Visual Basic for Applications. The most exciting features of this part of the Access environment are the ability to

- Write database programs using a full-featured programming language
- Leverage objects provided by Access and other programs in writing your own code
- Attach your code directly to your database objects
- Make a suite of applications work in concert with your database to perform complex tasks



Chapter 26

Creating Custom Error Messages



Creating Custom Error Messages

Even the best planned application occasionally screws up. When that happens in Access, an error is generated. You have two options for handling these errors: let Access and Visual Basic do it for you or respond to the errors yourself. The first option is very convenient. You don't have to do anything. However, the result of an error is that your application stops executing. The user sees a dialog box and whatever was happening stops. Although using the cryptic error descriptions that Access provides by default for error handling is convenient for you, it is rather rude for users of your application.

The second option, handling errors yourself, is less convenient for you, but makes your application much more user friendly. When you respond to the errors, you can interpret the error for your user and provide a suggestion about what to do. In many cases, you do not even have to inform the user that an error has occurred. You write code to correct the situation and restart the execution of your code.

This chapter teaches you how to handle errors yourself, and in the process shows you a couple of ways to go beyond the Wizard-written code that you learned how to create in the last chapter. To learn how to handle errors, you need to learn about flow control in Visual Basic programs. But first, we need to focus on the two ways to create custom error messages: there's one way for macros and one way for Visual Basic.

There is one time when error trapping is absolutely necessary: when you use the Access Developer's Toolkit (ADT) to create a "run time" version of your application. In this case, any untrapped error causes your application to quit completely.

Building Custom Error Messages with a Macro

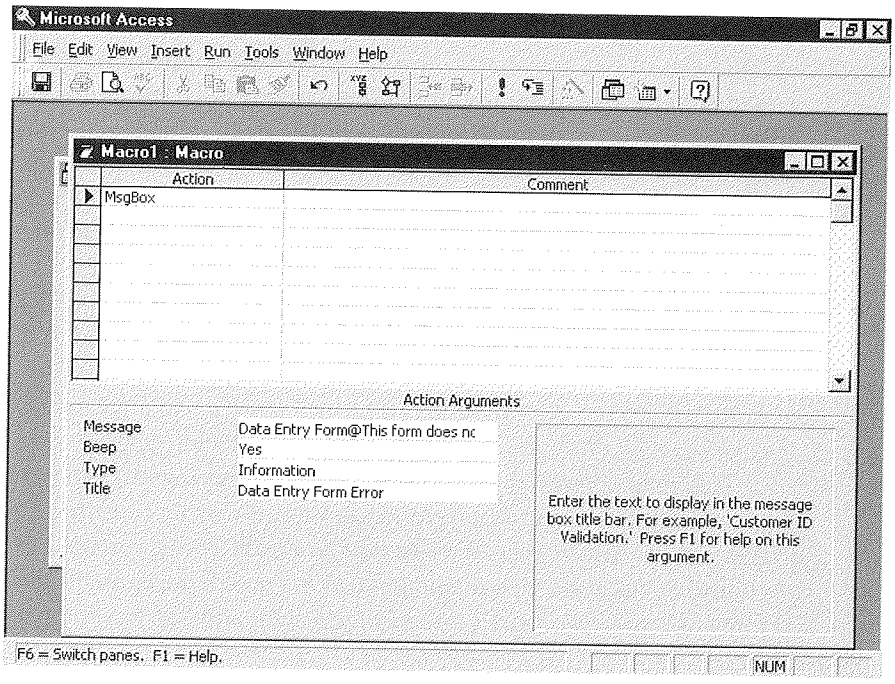
When you are using macros, you can create a custom error message using the MsgBox action. To do so, create a new macro by clicking on the New button in the Macros tab of the database window. On the macro sheet, select MsgBox as the action and fill in the properties. Figure 26.1 shows a completed error macro.

When you fill in the arguments, you can create a formatted message in the Message box using the @ character. Format your message in the following way:

```
Data Entry Form@This form does not support double-clicking. @Use the buttons at
   the bottom of the form to scroll to the record you want.
```

FIGURE 26.1

A macro designed to notify users that double-clicking has no effect in this context.

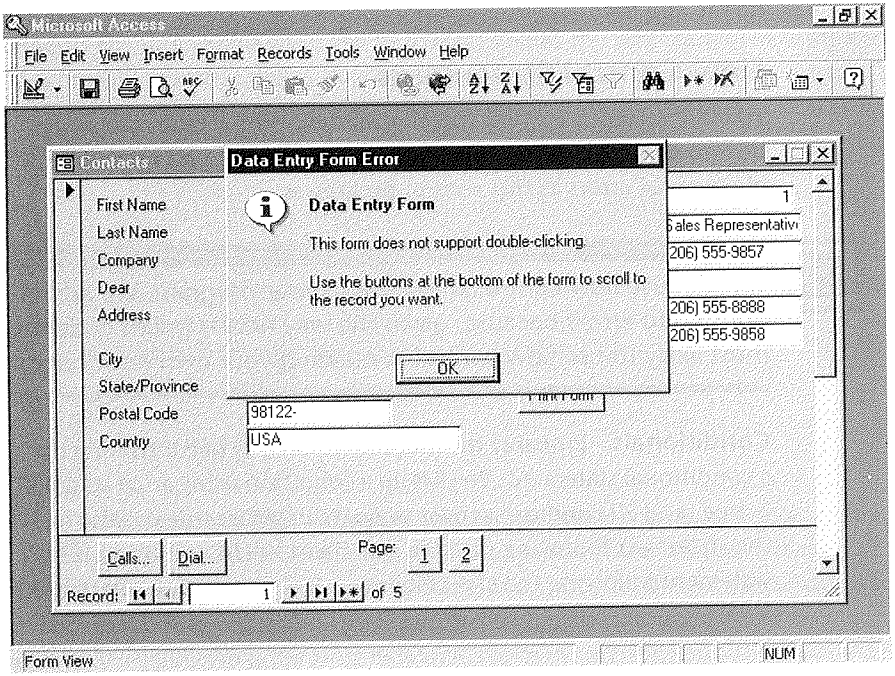


The first third of the message, to the left of the first @ symbol, appears in boldface at the top of the message box window. The second third of the message, between the two @ symbols, appears on the next line. The remainder of your message appears just below, as shown in Figure 26.2.

The remaining arguments for the MsgBox action govern additional features of the message box. The Beep argument determines whether a warning beep is played when the message box appears. The Type argument determines which icon is displayed in the message box: Information (as in the example in Figure 26.2), Critical, Warning?, Warning!, or none. The Title argument contains the text string displayed in the title bar of the message box window.

To display such a custom error message, you assign the macro to an appropriate event argument of the object to which it refers. In the case of a data entry form, we would assign this macro to the On Dbl Click event property for each object on the form. If your custom error message is intended to respond to an error generated by Access, assign it to the On Error event property.

FIGURE 26.2
A custom message displayed using the MsgBox action.



Building Custom Error Messages in Visual Basic

To create a custom error message in Visual Basic for Applications, you need to learn a little more about VBA programming. You need to learn how the flow of execution is controlled in a Visual Basic program, and you need to learn how to handle the flow of execution when an error occurs.

Flow Control in Visual Basic

Typically, Visual Basic executes a procedure by either copying the arguments into the variables made available to the procedure or (for **ByVal** arguments) giving the function access to the variable named in the argument. After arguments are made available, Visual Basic starts at the first statement in the procedure and executes each statement in line until it reaches the **End Sub** or **End Function** statement. If an error occurs, Visual Basic stops executing the code and displays an error message. It does not resume executing the code after the user dismisses the error message from the screen.

You can change the order in which statements are executed in several ways. For our purposes in explaining error handling, we will refer to these as using flow control statements and invoking flow control on error.

Flow Control Statements

To build effective custom error handling in Visual Basic procedures, you have to know how to control the flow of execution. We'll show you how to respond to several possible types of errors as this section progresses.

NOTE

You can use flow control anywhere in your program. We are discussing it in relationship to errors because, given the way Access builds code for you, you are most likely to use flow control in building your own error handlers.

Conditionals - Probably the most common way of controlling the flow of execution is a conditional statement. You set up a condition using an **If** statement. If the condition is true (and all conditions must evaluate to either true or false logically), you execute the statement following the **Then** statement. If the condition is false, you execute the statement following the **Else** statement. You can express alternate conditions using an

ElseIf statement. The whole conditional ends with an **EndIf** statement. Such a block of code looks like the following:

```
If ErrorNumber = 10 Then
 ' Code lines that execute if the ErrorNumber
 ' equals 10 go here.
ElseIf ErrorNumber = 100 Then
 ' Code lines that execute if the ErrorNumber
 ' equals 100 go here.
Else
 ' Code lines that execute if the ErrorNumber
 ' is equal to some other value go here.
End If
```

Loops - Loops allow you to repeat a set of statements over and over again. In handling errors, you will find loops most useful when you are correcting an error condition. For example, in response to a *File Not Found* error, you might want to loop through the list of controls on a form to make certain that those used to represent the file name have valid values. When you find the control that needs correction, you could prompt the user to enter a valid value.

You have four different ways of looping available to you. A **Do** loop executes the statements that appear between the **Do** statement and the **Loop** statement until a condition, expressed in either the **Do** or **Loop** lines, is false or while a condition is true. (You determine whether the loop executes until or while by using the **While** or **Until** keywords.) You can exit such a loop at any time by using an **Exit Do** statement. Such loops look like the following:

```
Counter = 0
Do While Counter < Me.Controls.Count
      If Me.Controls(Counter).Text = "" Or
IsNull(Me.Controls(Counter).Text) Then
       Me.Controls(Counter).Text = InputBox("Enter _
       a proper filename")
      End If
      Counter = Counter + 1
Loop
```

This example assumes that a form has several text controls on it, each holding a file name. The user has asked Access to open the files, and one of the controls is blank. An error occurs, and to handle it, you choose to check the text of each box to make certain none of them are blank. (You might also check for invalid characters as well.) Every form has associated with it a collection of controls. **Me** is an Access keyword referring to the form to which the code is directly attached. You can access each control using the notation shown. The

collection is named Controls, and each control can be identified by an index number. By using the variable Counter, you can start at the first control in the collection (Control(0)) and repeat the action for each control until the last one on the form (Control(9)). If we encounter a blank control, we use an **InputBox** statement, which displays the text in the argument and a text box in a dialog box, collects input from the user, and returns the text to be stored in the variable `Me.Controls(Counter).Text`, which is also the text property of the control. If you wanted to exit the loop when you found the first blank text control, you would place an **Exit Do** statement immediately before the **End If** statement.

TIP

You could accomplish the same task using the statement `Do Until Counter = 10`. The expression can also be placed on the last line, as in `Loop Until Counter = 10`.

You could accomplish the same error-handling task using a counted loop. These loops begin with the keyword **For** and execute a fixed number of iterations named in the first line. These loops have the following form:

```
For Counter = 0 To 9
      If Me.Controls(Counter).Text = "" Then
       Me.Controls(Counter).Text = InputBox("Enter _
       a proper filename")
      End If
Next Counter
```

In **For** loops the counter is incremented for you automatically. In this case it starts at the value 0 and increments to 9 automatically. The **Next** statement marks the end of the loop code. Although the name of the counter variable is optional on the last line, it's a good idea to use it. If you have loops within loops, you can tell which **Next** goes with which **For**.

Visual Basic provides a loop to use with collections of objects that simplifies the task of looping. These loops have the following form:

```
For Each Control In Me
      If Control.Text = "" Or IsNull(Control.Text)
       Then
       Controls.Text = InputBox("Enter _
       a proper filename")
      End If
Next
```

This form of the **For** loop allows you to avoid having to manage a counter. What if you aren't sure how many controls are on the form denoted by **Me**? Access knows, and

this loop simply makes use of Access's internal count to increment a counter that is hidden from you.

You can exit a For loop at any point by using the Exit For statement.

The last type of loop executes a set of statements while a condition remains true. It has this form:

```
Counter = 0
While Counter < 10
      If Me.Controls(Counter).Text = "" Or
IsNull(Me.Controls(Counter).Text) Then
       Me.Controls(Counter).Text = InputBox("Enter _
       a proper filename")
      End If
      Counter = Counter + 1
Wend
```

This form of loop is very like a **Do** loop with some simplification of form. It is bounded by **While** and **Wend** statements, and once again you must increment your own counter.

To learn more about any statement used in Visual Basic, load the Visual Basic Reference Help file from your Access disks or CD-ROM. You may need to use Add/Remove Programs to do so because this file is not included as part of a Typical install. Once it is installed, it appears in the list of Access help topics. For instructions on loading the language reference, search for *Visual Basic* with the Office Assistant and click on *About Getting Help on Visual Basic.*

Branches - Branches are critical to error-handling code, as you will see in the next section. Understanding them is therefore essential to learning how to make Visual Basic handle errors elegantly.

The concept of branching goes back to the days when each line in your program was numbered. If you knew the flow of execution had to skip from one line to another distant line, you could use a **GoTo** statement to jump to the appropriate line number.

Although you can still use line numbers in Visual Basic, the practice is not recommended. (Eventually you have to renumber all the lines because of additions or changes to the program, and then all your line-number-based jumps have to be adjusted by hand.) Instead, you jump to labels, which are strings of text used as bookmarks in the code. A label

is any string of text ending in a colon. Visual Basic keeps track of where they are and can move to them when you use a **GoTo** statement to request a jump.

The **GoTo** statement has the following form:

```
GoTo Label
```

And the label would appear in your program code in a line as

```
Label:
```

When Visual Basic encounters the **GoTo** statement, it moves to the text string Label: and resumes execution with the first line following.

A variant of the **GoTo** statement allows you to return the flow of execution to the line after the statement that caused the jump to take place. The variant form looks like this:

```
GoSub Label
'Other program lines go here
Label:
      If Me.Controls(Counter).Text = "" Then
       Me.Controls(Counter).Text = InputBox("Enter _
       a proper filename")
      End If
Return
```

The **GoSub** statement works exactly like **GoTo** in that the jump occurs to the text string **Label:** and the **If** statement is executed. However, when Visual Basic encounters the **Return** statement, execution returns to the line following the **GoSub** statement. You can jump out, execute some code, and return to where you were.

Another variation of both these statements begins with the keyword **On**, as in the following lines:

```
On ErrorCode GoTo Label1, Label2, Label3
On ErrorCode GoSub Label1, Label2, Label3
```

In each of these statements, **ErrorCode** can be a variable or an expression that evaluates to a value ranging from 0 to 255. If the value is 0, execution resumes with the next statement. If the value is 1, execution jumps to Label1. If the value is 2, control jumps to Label2. If the value exceeds the number of items on the list of labels, execution resumes with the next statement. If the value is negative or greater than 255, an error occurs. These variations are just ways to have a single jump statement branch to several labels.

NOTE

You must remember that Visual Basic ignores any label it encounters in the normal flow of execution. As a result, if you have several labels at the end of a procedure for handling errors, you need to have a means of stopping your program before it gets to those labels in the event no error occurred. We'll show you how in the next section.

The most effective branching statement, however, is the **Select Case** statement. It allows multiple branches, and it allows you to respond to discontinuous values of an expression or variable. As a result, you can collect an error number (patience, that information is coming!) and branch to the statements that handle the error appropriately. You can even specify what to do in case there is no match for the values you specify. The **Select Case** statement looks like this:

```
Select Case ErrorNumber 'Evaluating the error number
Case 1, 2, 3 'Error numbers 1 through 3
      Debug.Print "I have mapped these error numbers
       to the same message"
Case 5 To 8 'Error numbers 5, 6, 7, and 8
      Debug.Print "Error numbers 5, 6, 7, and 8
       receive this message through Debug.Print"
Case Is > 8 And ErrorNumber < 20 'Error numbers
 between 8 and 11
      Debug.Print "I have demonstrated how to trap
       error numbers within a range, specifically
       between 8 and 20"
Case Else 'What to do when there is no match
      Debug.Print "All other error numbers get this
       message"
End Select
```

This example shows that you can use various expressions to form the matching statements. Each **Case** statement represents one possible match. The **Case Else** statement marks the code that will execute if no match is found. When a **Case** or **Case Else** statement executes, the code lines following it execute and the flow of execution jumps to the statement following the **End Select** statement.

Flow Control On Error

We're sure that seems like a lot about flow control, but error handling in Visual Basic depends on flow control. You use an **On Error** statement to determine the flow control when an error occurs. If you don't have an **On Error** statement in your procedure, execution stops when an error occurs. If you want the opportunity for your user to be able to correct the error or if you want execution to continue at all for any reason, you need to use one of these three variations of the statement:

- **On Error GoTo Label** This version causes control to jump to the specified label when an error occurs. At the end of the error handler at that label, a **Resume** statement causes execution to resume with the line after the one that caused the error.
- **On Error Resume Next** This version causes the error to be ignored and execution to continue normally.

is any string of text ending in a colon. Visual Basic keeps track of where they are and can move to them when you use a **GoTo** statement to request a jump.

The **GoTo** statement has the following form:

```
GoTo Label
```

And the label would appear in your program code in a line as

```
Label:
```

When Visual Basic encounters the **GoTo** statement, it moves to the text string Label: and resumes execution with the first line following.

A variant of the **GoTo** statement allows you to return the flow of execution to the line after the statement that caused the jump to take place. The variant form looks like this:

```
GoSub Label
'Other program lines go here
Label:
     If Me.Controls(Counter).Text = "" Then
      Me.Controls(Counter).Text = InputBox("Enter _
      a proper filename")
     End If
Return
```

The **GoSub** statement works exactly like **GoTo** in that the jump occurs to the text string **Label:** and the **If** statement is executed. However, when Visual Basic encounters the **Return** statement, execution returns to the line following the **GoSub** statement. You can jump out, execute some code, and return to where you were.

Another variation of both these statements begins with the keyword **On**, as in the following lines:

```
On ErrorCode GoTo Label1, Label2, Label3
On ErrorCode GoSub Label1, Label2, Label3
```

In each of these statements, **ErrorCode** can be a variable or an expression that evaluates to a value ranging from 0 to 255. If the value is 0, execution resumes with the next statement. If the value is 1, execution jumps to Label1. If the value is 2, control jumps to Label2. If the value exceeds the number of items on the list of labels, execution resumes with the next statement. If the value is negative or greater than 255, an error occurs. These variations are just ways to have a single jump statement branch to several labels.

NOTE

You must remember that Visual Basic ignores any label it encounters in the normal flow of execution. As a result, if you have several labels at the end of a procedure for handling errors, you need to have a means of stopping your program before it gets to those labels in the event no error occurred. We'll show you how in the next section.

The most effective branching statement, however, is the **Select Case** statement. It allows multiple branches, and it allows you to respond to discontinuous values of an expression or variable. As a result, you can collect an error number (patience, that information is coming!) and branch to the statements that handle the error appropriately. You can even specify what to do in case there is no match for the values you specify. The **Select Case** statement looks like this:

```
Select Case ErrorNumber 'Evaluating the error number
Case 1, 2, 3 'Error numbers 1 through 3
     Debug.Print "I have mapped these error numbers
      to the same message"
Case 5 To 8 'Error numbers 5, 6, 7, and 8
     Debug.Print "Error numbers 5, 6, 7, and 8
      receive this message through Debug.Print"
Case Is > 8 And ErrorNumber < 20 'Error numbers
 between 8 and 11
     Debug.Print "I have demonstrated how to trap
      error numbers within a range, specifically
      between 8 and 20"
Case Else 'What to do when there is no match
     Debug.Print "All other error numbers get this
      message"
End Select
```

This example shows that you can use various expressions to form the matching statements. Each **Case** statement represents one possible match. The **Case Else** statement marks the code that will execute if no match is found. When a **Case** or **Case Else** statement executes, the code lines following it execute and the flow of execution jumps to the statement following the **End Select** statement.

Flow Control On Error

We're sure that seems like a lot about flow control, but error handling in Visual Basic depends on flow control. You use an **On Error** statement to determine the flow control when an error occurs. If you don't have an **On Error** statement in your procedure, execution stops when an error occurs. If you want the opportunity for your user to be able to correct the error or if you want execution to continue at all for any reason, you need to use one of these three variations of the statement:

- **On Error GoTo Label** This version causes control to jump to the specified label when an error occurs. At the end of the error handler at that label, a **Resume** statement causes execution to resume with the line after the one that caused the error.
- **On Error Resume Next** This version causes the error to be ignored and execution to continue normally.

- **On Error GoTo 0** This version causes the current error handling to be canceled. You can then use another form of `On Error` to redirect flow control in a different way.

It's time to look again at the click event procedure you created for the Print button in the last chapter. Here is the procedure:

```
Sub Print_Form_Click()
On Error GoTo Err_Print_Form_Click
 DoCmd.PrintOut
Exit_Print_Form_Click:
 Exit Sub
Err_Print_Form_Click:
 MsgBox Err.Description
 Resume Exit_Print_Form_Click
End Sub
```

Notice the **`On Error`** statement at the beginning of the procedure. When an error occurs anywhere in the procedure, it redirects execution to the label `Err_Print_Form_Click`. This label leads to code that handles the error. The **`Resume`** statement includes the label `Exit_Print_Form_Click` as an argument, which causes the resumption of execution at that label. The first statement following this label causes the event procedure to exit.

Notice the logic of building the procedure. If no error occurs, the statements execute sequentially until the **`Exit Sub`** statement causes the procedure to end. If an error occurs, execution jumps beyond the **`Exit Sub`** statement to the error-handling code. After the errors are handled, execution jumps back to the **`Exit Sub`** statement. This procedure provides an excellent template for developing error-handling routines.

You might recognize the MsgBox function in the error-handling code and guess that it performs the same function as the macro of the same name. It does. But what is that Err.Description argument? Visual Basic provides an Err object. When an error occurs, it makes the Err object, with its properties and methods, available to you. The Description property is the text string that describes the error. The MsgBox function builds a message box that announces that an error of this type has occurred.

The Err object also has a Number property. This property can be used with branching statements to allow you to handle any of several errors that might occur in your procedure. (The **`Select Case`** statement is especially useful for this purpose.)

For more information about the Err object, search Help for *Err*.

Building the Error Message

So, having been drilled in the basics of Visual Basic error handling, you might have guessed already how to provide a custom message to your users: Use the MsgBox function and use

your own text string rather than the one stored in Err.Description. Congratulations! You've mastered some of the mysteries of Visual Basic coding.

You might wonder why we say *MsgBox function* rather than *MsgBox statement*. The reason is that MsgBox returns a value. It has some subtleties, therefore, that you should know about.

First, MsgBox has three critical arguments, which must occur in the order specified. (Although you can omit the Buttons and Title arguments, you will almost always wish to supply them, since they provide information to your users.)

- **Prompt** The text string to display as the message.
- **Buttons** A number that determines which buttons are displayed in the message box. These numbers are represented by constant expressions maintained by Visual Basic for your use. As a result, you should use the constant names, as shown here.
- **Title** The text string to display in the title bar of the message box.

You can use the @ symbol to format the message string exactly as you do when you use the MsgBox macro to display a custom message.

If you want to learn more about the arguments for MsgBox, search the Help file for the function name.

The Buttons argument can take the following values:

- **vbOKOnly** Show OK button only.
- **vbOKCancel** Show OK and Cancel buttons.
- **vbAbortRetryIgnore** Show Abort, Retry, and Ignore buttons.
- **vbYesNoCancel** Show Yes, No, and Cancel buttons.
- **vbYesNo** Show Yes and No buttons.
- **vbRetryCancel** Show Retry and Cancel buttons.
- **vbCritical** Show Critical Message icon.
- **vbQuestion** Show Warning Query icon.
- **vbExclamation** Show Warning Message icon.
- **vbInformation** Show Information Message icon.
- **vbDefaultButton1** First button is default.
- **vbDefaultButton2** Second button is default.
- **vbDefaultButton3** Third button is default.
- **vbApplicationModal** Only the Access application is suspended until the user responds to the message box.
- **vbSystemModal** Suspend all applications until the user responds.

To combine any of these attributes to determine what is displayed and how, add these constants together using the plus sign.

The following code fragment shows how to build the same custom message that you built using the macro earlier in this chapter:

```
strMsg = "This form does not support " _
& "double-clicking.@Use the buttons at the bottom of "
& "the form to scroll to the record you want."
intResult = MsgBox (strMsg, vbOkOnly +
 vbInformation, "Data Entry Form")
```

The variable intResult collects the return value of the function, which allows you to determine which button in the message box the user clicked on. The possible return values are the following:

- **vbOK** OK button clicked.
- **vbCancel** Cancel button clicked.
- **vbAbort** Abort button clicked.
- **vbRetry** Retry button clicked.
- **vbIgnore** Ignore button clicked.
- **vbYes** Yes button clicked.
- **vbNo** No button clicked.

Obviously, you can use flow control statements to test the return value and take appropriate action in response to the user's click.

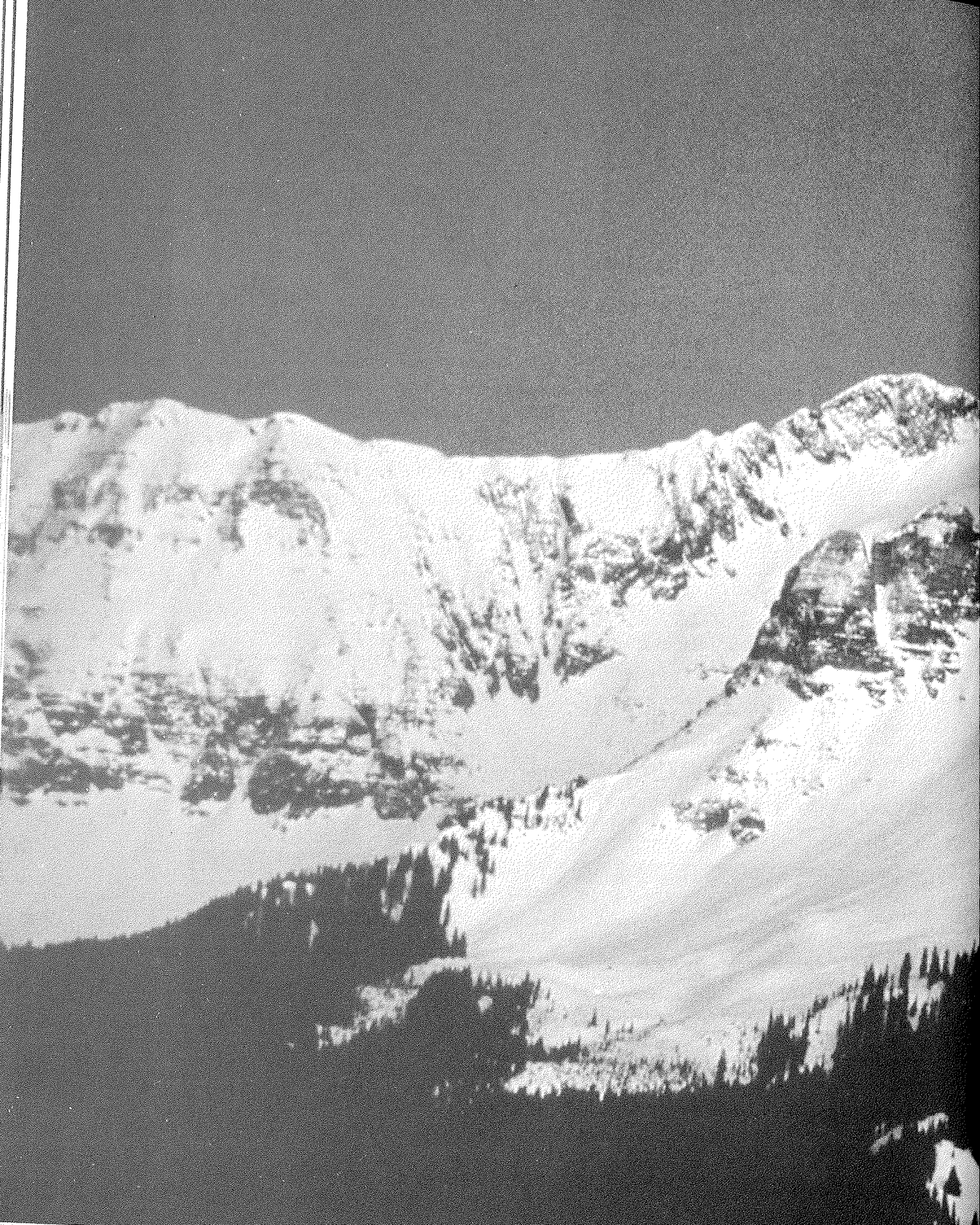
Where to Go from Here

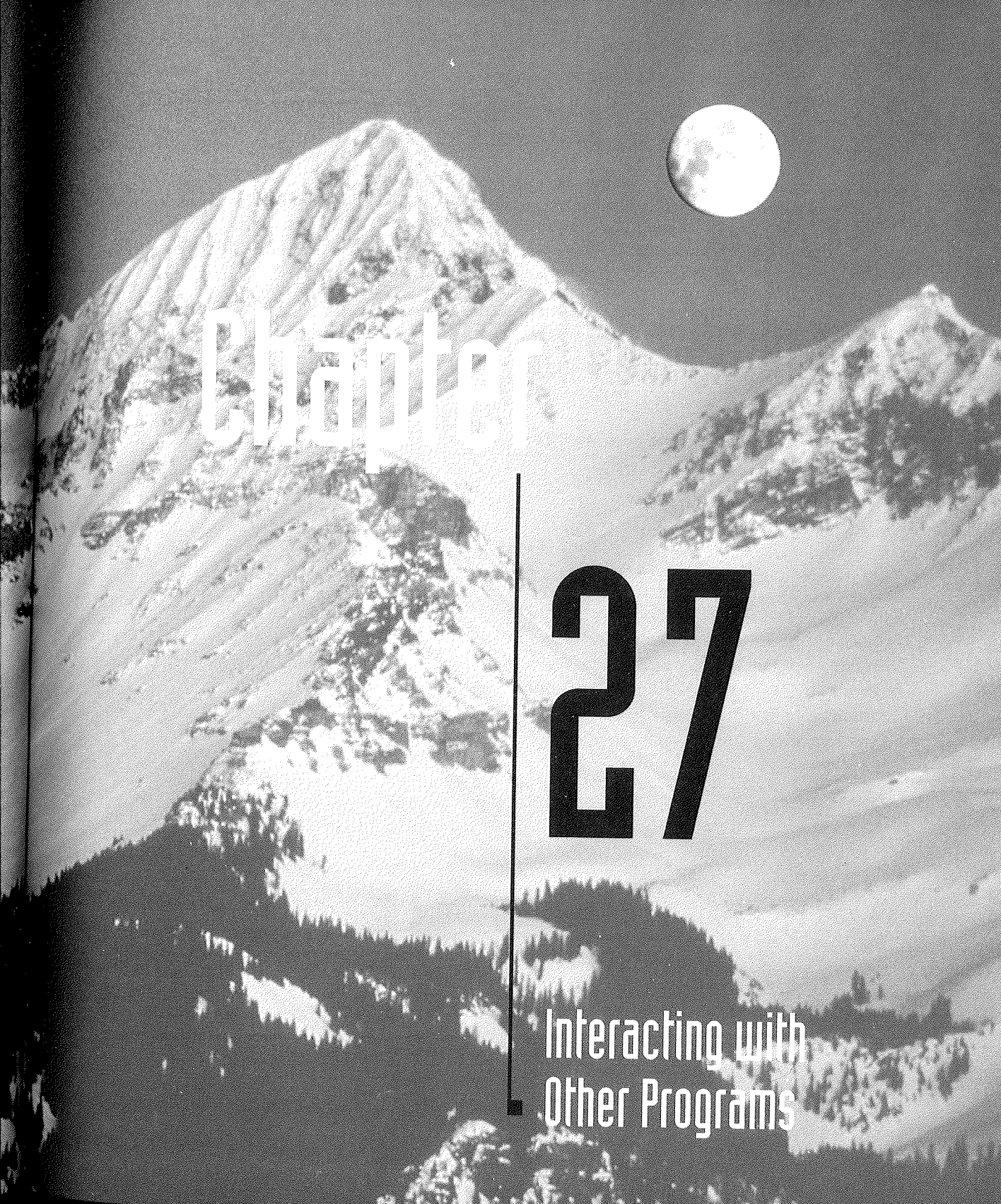
Now that you have a grasp of flow control and error handling, in the next chapter we'll show you how to control other applications from inside your Access application. This is a slick trick, so be prepared to see some real magic.

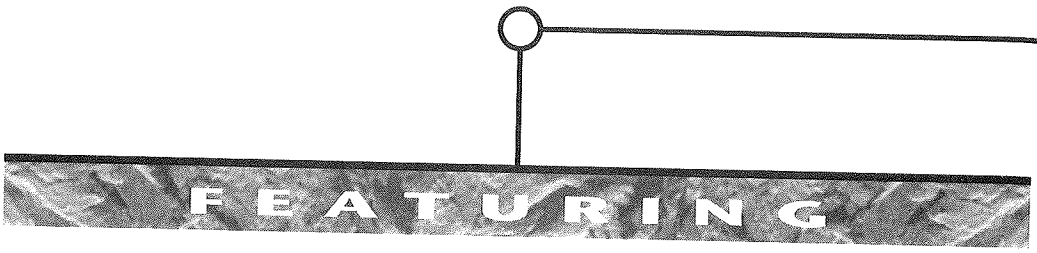
What's New in the Access Zoo?

Microsoft Access 97 for Windows 95 has features for creating your own error messages. As with Access 95, you can correct error conditions by using either

- Visual Basic On Error statements
- Visual Basic flow control statements




Chapter
27
Interacting with
Other Programs



Interacting with Other Programs

One of the greatest features of Access 97 for Windows 95 is its support for OLE automation, a technology that allows you to control other programs from within your database applications. To use this feature, you need to be able to program in Visual Basic for Applications code. You also need to be aware of some quirks associated with OLE objects and their automation. In this chapter we will explain those quirks and provide you with a basic template for using automation in your own applications.

What Is OLE Automation?

OLE automation is a way of treating any application on your system as an object that belongs to Access. Imagine being able to treat Word or Excel as though they behaved exactly like a command button that you draw on a form. You could carry out any command in either application using the *object.method* notation. And you would have access to every command. You could reliably start the application from within an Access application and direct the application to do exactly what you wanted it to do.

OLE automation gives you this kind of flexibility, but there are a couple of hitches:

- The application you want to control has to be an OLE automation server. In other words, it has to be designed to allow itself to be automated.
- When you automate an object, you get access only to the commands to which the designers allow you access. The set of commands available may change depending on whether the data object you intend to use is embedded in your database.
- You have to know something about the command structure of the program you are automating. In automating Word and Excel, for example, you use Visual Basic for Applications. However, you have to master program-specific extensions to Visual Basic.

You can work your way around most of the limitations you might encounter. Quite obviously, you would not try to automate a program that cannot be automated. And you can find ways to "workaround" limited functionality. In addition, Access provides an object browser to allow you to see the automation objects available and the methods you can use. This chapter uses workarounds and the object browser.

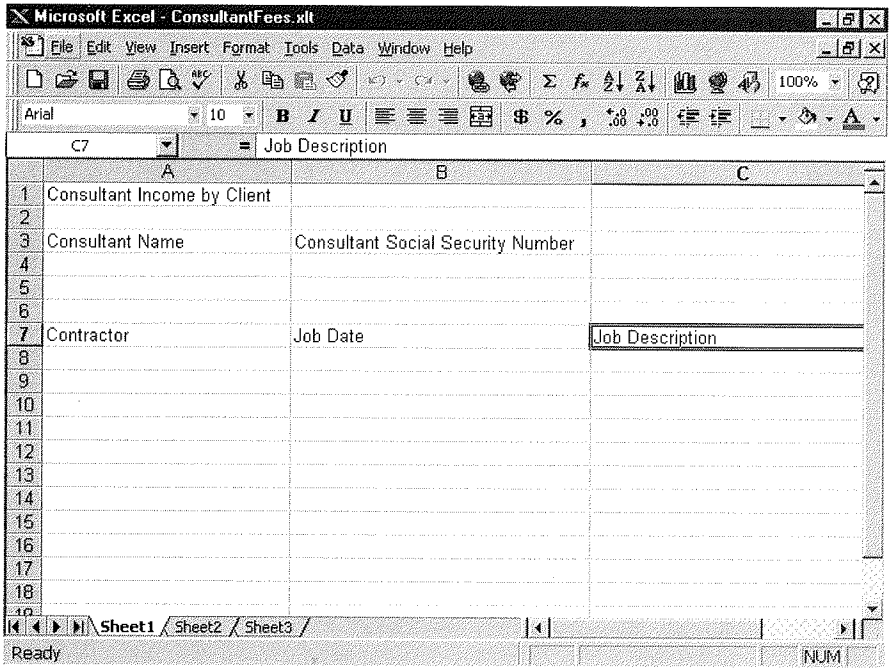
Creating the Basic Object

To use automation, first you have to make an object out of the application you intend to automate. In this example, we will automate the creation of an Excel worksheet. The worksheet will be based on a template we'll make ahead of time. The name and social security number from the Access record we are viewing will be filled into the worksheet, the worksheet will be saved, and Excel will be closed.

You must do two things as preliminaries to prepare to automate these actions. First, you must create an Excel template for the worksheet. Assume that the members of our nonprofit corporation, whose database you have seen from time to time in examples, want to build an Excel worksheet to track the fees their educational consultants earn. Open Excel, create a worksheet containing the boilerplate text, and save it as a template using File ➤ Save As. (Be sure to set the *Save as Type* to Template.) Figure 27.1 shows a simple worksheet of this sort.

The second step is to open a data entry form that includes controls for the name and social security number in design view and to add a button to the form. When you draw the button, select Application and Run MS Excel in the Wizard, as shown in Figure 27.2. Name the button AutomateExcel and finish the Wizard. You won't be using the actual code generated, but the Wizard will generate a useful event procedure template for you.

FIGURE 27.1
A sample Excel template for tracking consultant fees.



Now you are ready to create an object for Excel. Right-click on the button and follow these steps:

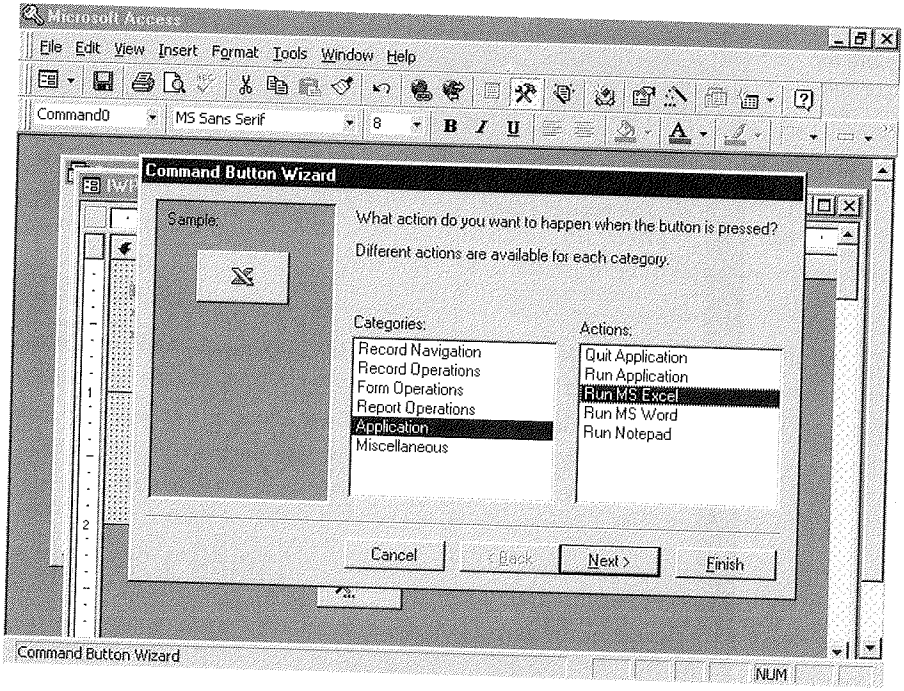
1. Select Build Event from the context menu.
2. Open the drop-down list in the upper-left corner in the code window and select (General).
3. Open the drop-down list in the upper-right corner and select (declarations).
4. Enter the following lines in the code window after the Option Compare Database line:

```
Dim xlObject As Object ' Declare variable to
' hold the
' reference to the
' Excel Object.
```

5. Open the Object drop-down list in the upper-left corner and select AutomateExcel. You will be taken to the Click event procedure that Access generated for you automatically.

FIGURE 27.2

Entering the correct information in the Command Button Wizard to get ready to automate Excel.



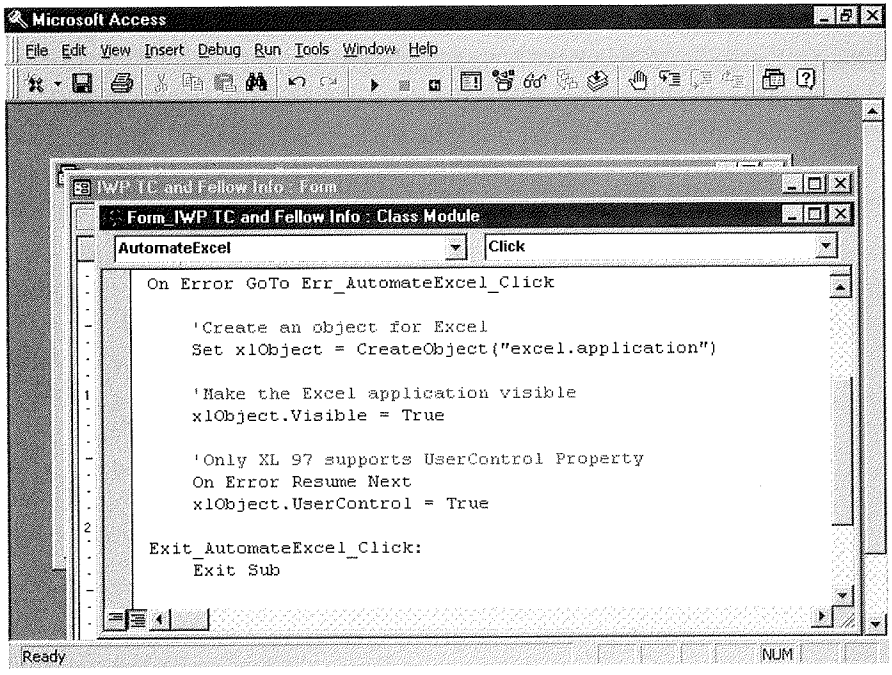
6. Revise the lines of code after the On Error statement so they look like the following lines of code:

```
' Create an object for Excel
Set xlObject = CreateObject("excel.application")

' Make the Excel application visible
xlObject.Visible = True

'Only XL 97 supports UserControl Property
On Error Resume Next
xlObject.UserControl = True
```

Your Click event procedure should look like Figure 27.3. At this point close the code window, take the form to form view, and try out the button. Excel should start and become visible.

What have you done so far? A pretty neat trick, actually. The Dim statement you added declared a variable that can hold a reference to an object. Whenever you want to use the object whose reference is stored there, you can use the variable name as the

FIGURE 27.3
The click event so far.



object name. You created the object variable at the form level so that it would remain available while the form was in use. If you created it in the event procedure itself, as soon as the event procedure terminated, Access would deallocate the variable's memory, effectively removing it from use until the event procedure was run again, when the variable would be recreated.

The Set statement in the Click procedure sets the variable equal to the return value of the CreateObject function. The CreateObject function—you guessed it—creates objects out of whatever item you provide as its argument. In this case we asked it to create an Excel application object. OLE automation-enabled applications provide strings that you can use as arguments to CreateObject to make objects out of them. CreateObject creates the object and returns a reference to the object. The reference is now stored in xlObject for later use.

Immediately after creating the object, you used it. Excel was started automatically because you created an object out of it. But Excel does not become visible until you decide to show it. The next line in the Click procedure sets the xlObject's Visible property to true, causing Excel to show itself to you on the screen.

Working with the Basic Object

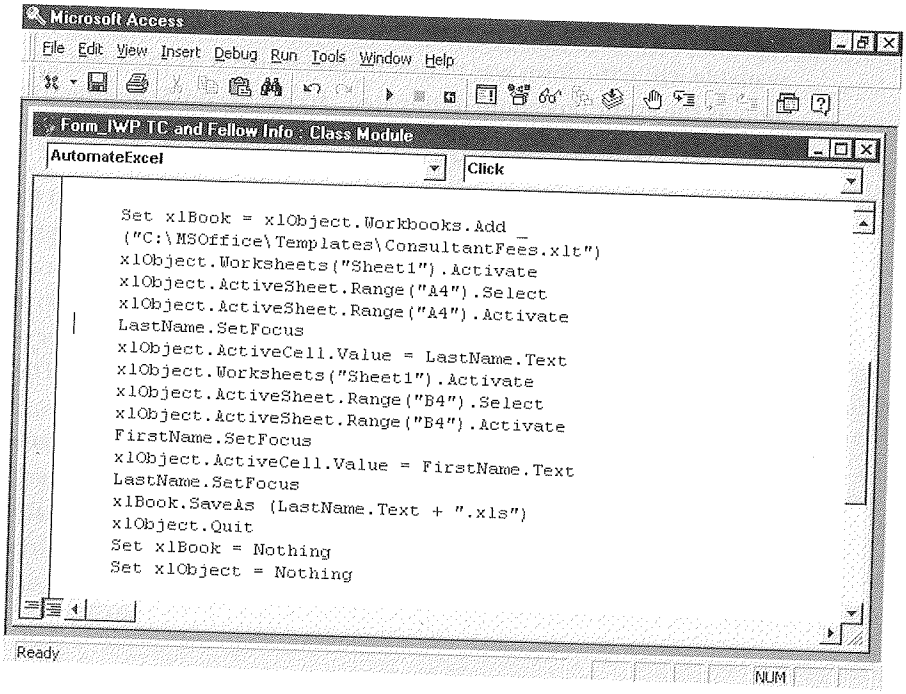
Now that you have created an Excel object, you might like to do something with it. First, having started Excel, you need to create a workbook. Return the form to design view and open the code window. Place the cursor in the code window at the beginning of the blank line after `xlObject.Visible = True` and add the line below. (Be sure to substitute the correct path name for `ConsultantFees.xlt` if you stored it in a directory other than `C:\MSOffice\Templates`.)

```
Set xlBook = xlObject.Workbooks.Add _
("C:\MSOffice\Templates\ConsultantFees.xlt")
```

Having created the workbook using the last few lines of code, we want to insert the appropriate information from the database into the Excel worksheet. To do so, you have to activate the worksheet, select the cell using Excel's Range function, and activate the cell. To accomplish these actions, add these lines of code:

```
' Activate the worksheet, select the range, activate
' a cell in the range.
xlObject.Worksheets("Sheet1").Activate
xlObject.ActiveSheet.Range("A4").Select
xlObject.ActiveSheet.Range("A4").Activate
```

FIGURE 27.4
The Click event with a VBA command to create a workbook.



In these lines, you use the Worksheets object to activate the worksheet and the ActiveSheet object to work with the cells on the worksheet you activated.

You can use the Object Browser to add most of these lines. Press F2 to open the Object Browser. If you don't see Excel on the top drop-down list, choose Tools ➤ References, check off Microsoft Excel 8.0 Object Library, and click on OK.

To actually insert information from Access into the Excel worksheet, you have to set the focus to the control containing the information on the Access form. You do so using the SetFocus method for the control. You can then set the value for the ActiveCell object in Excel. Insert these lines of code:

```
' Set the focus to the Last_Name control on the
' Access form.
LastName.SetFocus
' Place the last name in the active cell.
xlObject.ActiveCell.Value = LastName.Text
```

Next, you repeat the same process for the social security number cell using these lines of code:

```
' Repeat the process for the social security
' number cell.
xlObject.Worksheets("Sheet1").Activate
xlObject.ActiveSheet.Range("B4").Select
xlObject.ActiveSheet.Range("B4").Activate
SocialSecurityNumber.SetFocus
xlObject.ActiveCell.Value =
 SocialSecurityNumber.Text
```

To complete the operation of creating the spreadsheet, set the focus to the LastName control once again, since the last name of the consultant will be the file name for storing the workbook. Then use the Workbook's SaveAs method to save the spreadsheet:

```
' Set the focus to the Access Last_Name control
 ' and save the workbook.
LastName.SetFocus
xlBook.SaveAs (LastName.Text + ".xls")
```

Now take the form to Form View and click on the button. Watch as Excel opens and builds the worksheet completely under your control from Access.

Closing Your Automation Session

Having learned how to automate Excel and do some work with it, you might want to know how to stop Excel when you want to. Otherwise you could have multiple copies of Excel starting, and eventually your system would crash from lack of memory, even if you have a lot of memory. To automate the close routine, you do two things. Use the Excel object's Quit method and set the value of any variables you used to store object references to the built-in value Nothing. The following lines of code do the trick:

```
' Quit Excel and clear the object variables.
xlObject.Quit
Set xlBook = Nothing
Set xlObject = Nothing
```

Now when you click on the button, Excel creates the spreadsheet and then closes itself. (The assumption is that you create these worksheets infrequently; otherwise, it would make more sense to keep the workbook open until you were finished and to set the name of each sheet to the last name of the consultant.)

Where to Go from Here

OLE Automation offers you exceptional possibilities. You can conduct complete spreadsheet analyses, build documents, chart graphs, print presentations, etc., all from within Access. If you so desire, you can work from within Word or Excel as well.

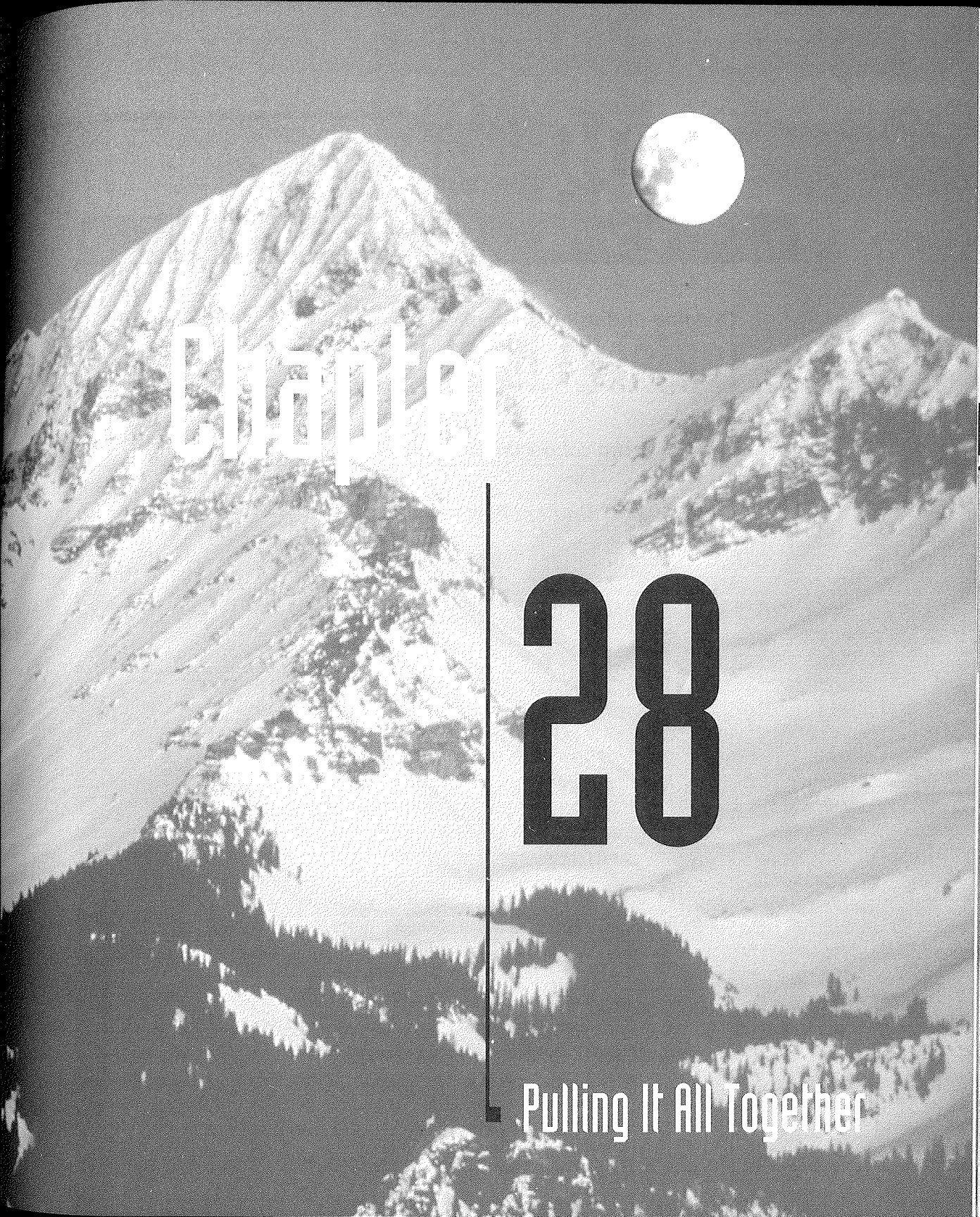
The main question you might have is how you can learn more about using these Access features. Take these steps:

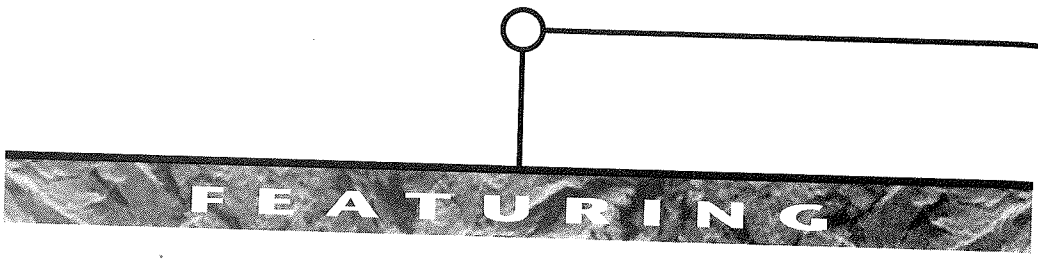
- Use the Object Browser to explore different applications to see which objects, properties, and methods are available to you.
- Check the documentation for each of your applications to learn the fundamentals of their macro languages. In Excel and Word, you can use VBA as well.

What's New in the Access Zoo?

Interacting with other applications has always been possible under Access. You also have these features to make such interaction easier:

- The Visual Basic for Applications programming language
- The ability to use objects provided by other applications
- The ability to allow other applications to use Access's own objects


Chapter
28
Pulling It All Together



Pulling It All Together

Chapters 1 through 27 have, for the most part, been about creating individual objects in Microsoft Access. Needless to say, many tools are also available for creating tables, queries, forms, reports, macros, and Visual Basic code.

But as with any endeavor, knowing how to create a finished product requires more than just knowing how to use the individual tools. For example, you might well know how to use a hammer, nails, saw, and drill, but do you know how to build a house? Maybe you can pick up a guitar and play any note or chord with ease, but how do you go about creating a song? Likewise, perhaps you know every tool and technique your word processor has to offer. But then, how do you go about creating a novel?

You might think of this latter transition as going from "toolmaster" to "artist." And for most people that transition is sort of the second brick wall in the learning process. That is to say, there's the first brick wall of learning what tools are available and how to use them; then there's the second brick wall of learning how to apply those tools to actually create something useful.

Many people get past the second brick wall by doing some "reverse engineering" from other peoples' work. In the fine arts, we take this approach for granted, even though we use a different terminology. We talk about the artist's "influences." That is

to say, the artist has mastered the tools of his trade. But in learning to apply those tools to create, the artist has no doubt observed and borrowed from other artists. The artist has reverse engineered the existing work to see what makes it tick.

We don't mean to imply that all works are plagiarism. To the contrary, anything the artist creates is uniquely his or her own creation. But in virtually every work of art or product that a human being creates, you can find some influences from the artists that preceded him or her.

So what does any of this have to do with Microsoft Access? Well, granted, Access is a tool for managing data. But it is also a tool for creativity. You can't create songs, novels, or houses from Access's tools. But you certainly can create Windows 95 applications.

A good way to make the transition from being an Access toolmaster to an Access applications artist is to reverse engineer other peoples' applications to see what makes them tick—to see how more experienced Access developers have applied Access's tools to create their own unique applications. For the rest of this chapter (and book) we want to give you some tips on how to learn by example.

What's Available to You

Microsoft has provided three complete applications with Microsoft Access for you to explore. They are named `Northwind.mdb`, `Orders.mdb`, and `Solutions.mdb`. You can typically find them in the `c:\msoffice\access\samples` folder after you've installed Microsoft Access.

NOTE

If any of the sample databases are missing from your Samples folder, you can install them from your original Office 97 CD-ROM or Access 97 floppy disks. From the Windows 95 desktop, click on the Start button and choose Settings ➤ Control Panel ➤ Add/Remove programs. Click on Microsoft Office 97 or Access 8.0 and then click on Add/Remove. Follow the instructions on the screen to add just the Sample Databases.

We've also included a sample application named Fulfill 95 on the CD-ROM that comes with this book, and we refer to that application throughout this chapter. To install and learn about Fulfill 95, see Appendix C.

Opening and Using an Application

Keep in mind that an Access database application is an Access database (.mdb file) with macros and code to make it perform like a stand-alone product. So you can open any

Access application by using the standard File ➤ Open Database commands within Access. You can also double-click the name of the .mdb file in the Windows 95 Explorer, Find, or My Computer icon to launch Access and load the application.

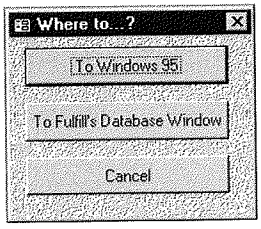
To use the application, you just interact with the command buttons, menus, and toolbars that appear on the screen. If you want to go spelunking behind the scenes to see what makes the application tick, you need to get to the application's objects. The general term we use for all the behind-the-scenes stuff is *source code*. That term is sort of a leftover from the days when the only thing you would find "behind the scenes" was program code. Nowadays, especially in an Access application, you'll see lots of objects—tables, queries, forms, reports, and macros, as well as code—behind the scenes. But Access developers use the term *source code* anyway.

Getting to the Source Code

Before we describe how to get into source code, we need to tell you that it isn't always possible to get to the source code! If an Access developer sells his or her product for a fee, he or she might have secured the database to prevent you from getting to the source code, using the techniques we covered in Chapter 18. We respect a developer's right to do that and have no complaints about it. You might, however, be able to purchase the source code from the developer.

Many applications, like the four we mentioned at the start of this section, are wide open to exploration. All you need to do is get past the forms and get to the database window. In the Fulfill 95 application, the process is quite simple:

1. Choose Exit from Fulfill's main switchboard. You'll come to this dialog box:



2. Choose To Fulfill's Database Window.

You're taken to the database window for all of Fulfill's objects. All the standard Access menus and toolbars are intact, so you can explore to your heart's content.

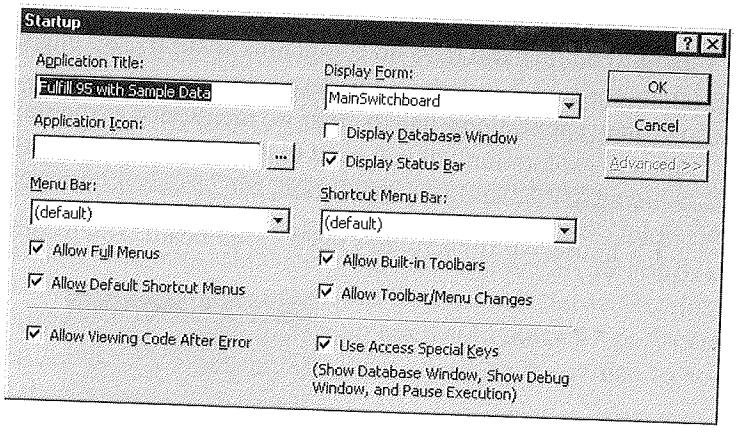
To go behind the scenes with the Northwind, Orders, or Solutions sample databases, you may have to close any custom form that appears by clicking on its Close (×) button. If you don't see the database window, try pressing the F11 key to bring it into view.

If an application gives you access to its database window, chances are you can also modify its startup options. Doing so will allow you to get right to the application's database window, as well as the standard built-in menu bars and toolbars, as soon as you open the application. To change the startup options:

1. Choose Tools ➤ Startup with the application's database window showing.
2. Select (check) the various Display and Allow options from the Startup dialog box to give yourself access to all the normal Access tools, as in Figure 28.1.
3. Choose OK.
4. Reopen the database to activate the new settings. That is, choose File ➤ Close. Then choose File and click on the database's name in the File menu to reopen with the new startup settings.

FIGURE 28.1

The Startup options determine how much access you'll have to built-in Access tools when first opening a database.



Finding Out What Makes It Tick

Once you get to an application's database window and have all the standard menus and toolbars in place, you're free to explore to your heart's content. Just click on the type of object you want to explore (e.g., Tables, Queries, Forms), click on the name of the object you want to explore, and then click on the Design button.

The bulk of the "meat" in most applications is in its forms. You can discover a lot about what makes an application tick by exploring its forms. Here's an example. Suppose you open, in design view, the form named AddressBook in Fulfill 95. Your screen might initially look something like Figure 28.2.

Why No Custom Menus in Fulfill 95?

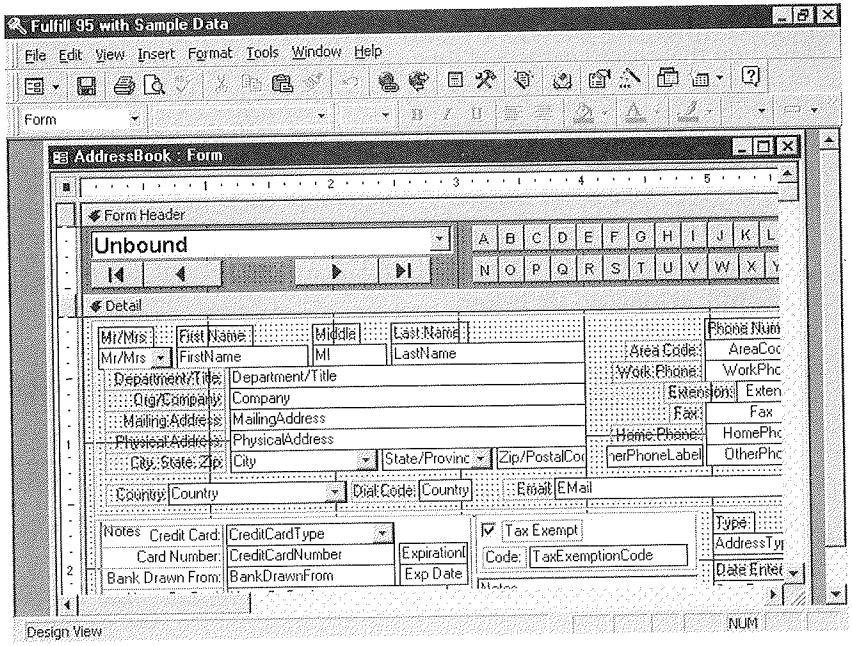
We've left custom menus and toolbars out of Fulfill 95 on the CD-ROM because its main purpose in life is to act as an example for aspiring application developers to explore. If we were to turn Fulfill 95 into a "real product," we'd have to do some more work.

For starters, we would create all custom menus and toolbars. Then we would use the Access Developer's Toolkit (ADT) to round out the application even further. We'd use the ADT (or some other commercially available tool) to create a standard Windows 95 help system for Fulfill. And then we'd use the ADT to generate a *runtime version* of Fulfill 95 so that we could distribute the application to people who don't own Microsoft Access. (The Microsoft Access Developer's Toolkit (ADT) is a separate product that must be purchased separately.)

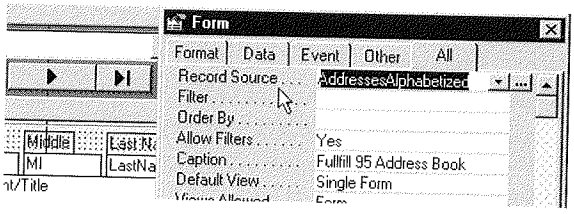
Then we'd probably give away a few copies to interested parties to get some feedback and to find any weaknesses and bugs we didn't catch the first time through. We'd fix any remaining problems before actually selling the product.

FIGURE 28.2

Fulfill's AddressBook form open in Design view.



To find out where this form gets its data, you need to look at the form's property sheet, so you choose Edit ➤ Select Form. Then you open the property sheet, click on the All tab, and take a look at the Record Source property. There you'll see that the form is based on a query named AddressesAlphabetized, as below.



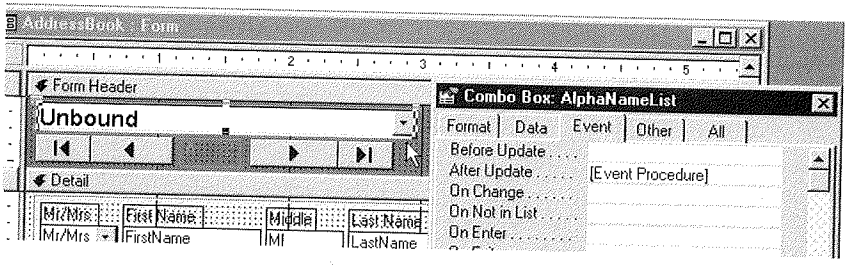
To see what makes that query tick, you just need to click on the Build button next to the query's name.

In virtually all applications, the elements that control most of the application's behavior are the *event procedures* assigned to the form and individual controls. To see the events assigned to a form, as a whole, you choose Edit ➤ Select Form, open the property sheet, and click on the Event tab. In this example we've created four custom event procedures for the AddressBook form, as you can see in Figure 28.3.

Many of the individual controls on a form will also have one or more custom event procedures assigned to them. To see a control's event procedure, click on the control you're interested in and take a look at the Event tab in the property sheet.

In some cases you may see the name of a macro in an event property. You can click on that macro name and then click on the Build button to explore the macro. In most cases you'll probably see [Event Procedure] as an event's property. The Event Procedure is Visual Basic code that's stored right along with the form and is accessible only through the form's design view.

Let's take a "for instance." Suppose we click on the big Unbound control in the AddressBook form. The property sheet informs us that (1) this control is a combo box, (2) its name is AlphaNameList, and (3) a custom event procedure is assigned to this control, as you can see below.



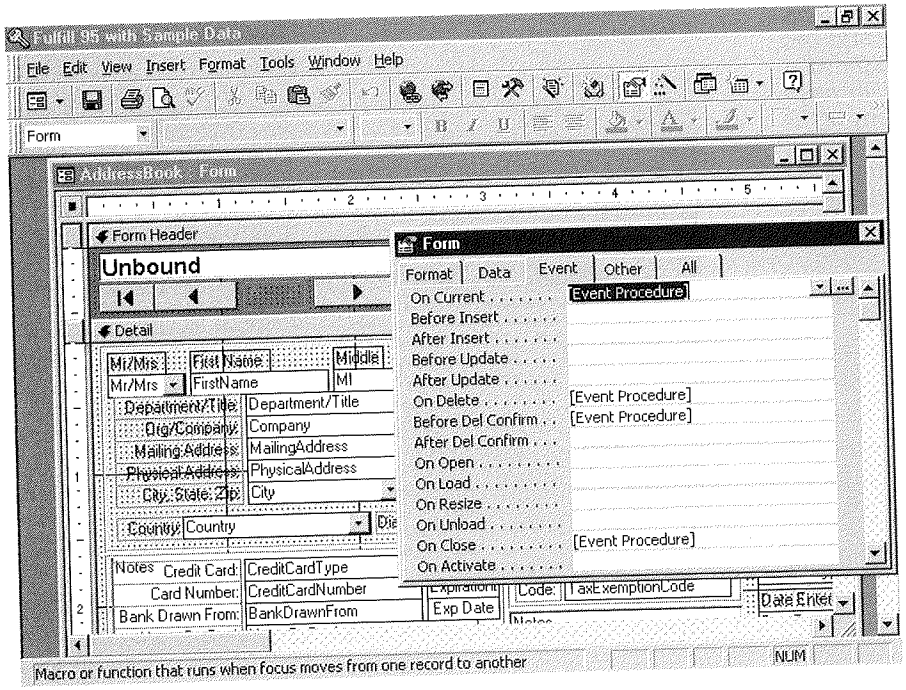
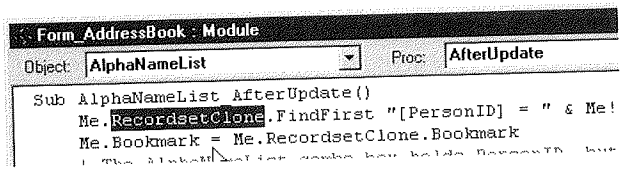


FIGURE 28.3
The Address-Book form's behavior is controlled by four event procedures.

To go behind the scenes and see the Visual Basic code, click on [Event Procedure] in the property sheet and then click on the Build button. The underlying code appears in a module window, as shown in Figure 28.4.

Unless you happen to be a Visual Basic expert, your reply to this finding might be, "Great, but I have no idea what any of that code means." One of the cool things about Microsoft Access is that the online Help is linked to keywords in the code. You can start learning what all the various Visual Basic commands do just by exploring Help right from this screen.

For example, to find out what *recordsetclone* is all about, you could drag the cursor through that command to select it.



When you press Help (F1), a help screen appears telling you about RecordsetClose, as in Figure 28.5.

FIGURE 28.4

Visual Basic code attached to the After Update property of the AlphaNameList control in the AddressBook form.

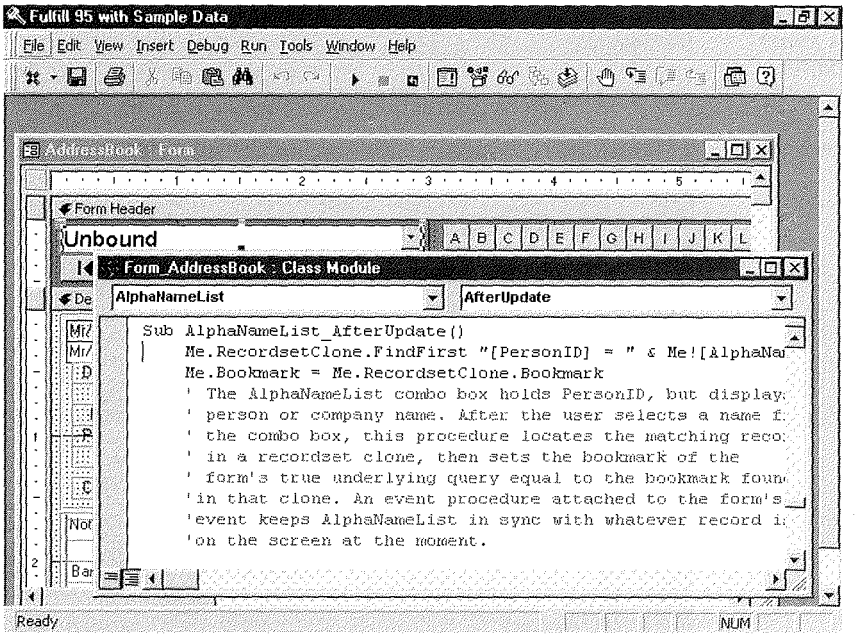
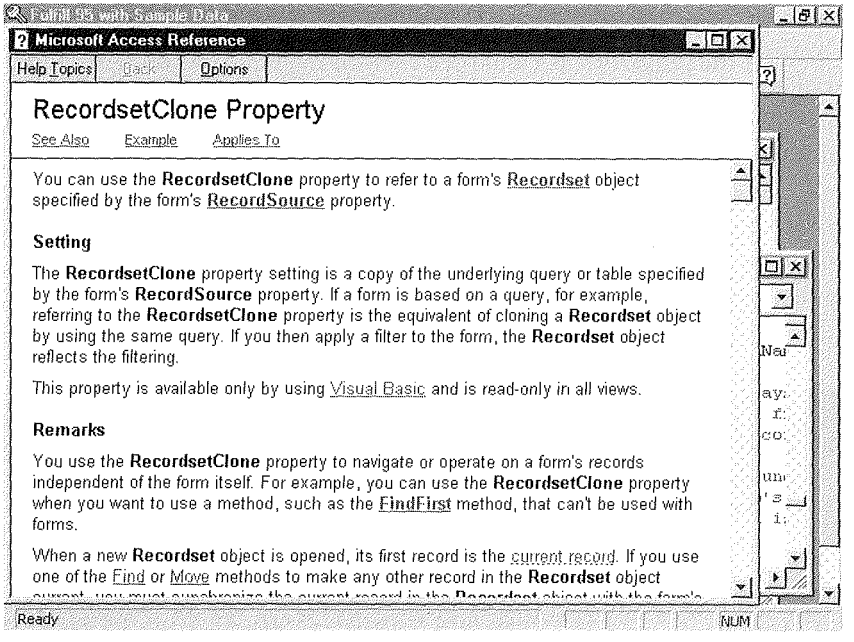


FIGURE 28.5

Information about RecordsetClone on the screen.



What Good Is This Information?

Now you might look at that so-called help screen back in Figure 28.5 and think, "But it's so technical, it really doesn't help me at all." If that's the case, you're reading all of this much too early in your application-development apprenticeship.

The reason this information appears in Chapter 28 is that it won't do you any good until you know how to do all the stuff in Chapters 1 through 27! This kind of exploration is useful only when you're ready to go from toolmaster to artist. You have to know the "mechanics" of creating tables, queries, forms, controls, reports, macros, and Visual Basic code before the nitty-gritty technical details of individual Visual Basic commands have any useful meaning to you.

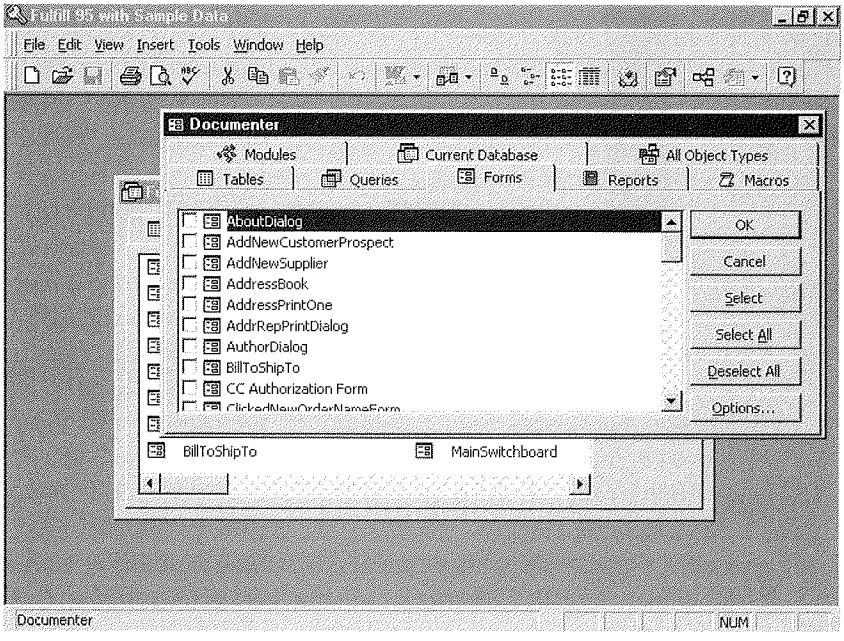
Printing Technical Documentation

Pointing and clicking on individual objects is just one way to explore existing Access applications. You can also print all the technical behind-the-scenes stuff. Here's how:

1. Open the application you want to explore and get to its database window.
2. Choose Tools ➤ Analyze ➤ Documenter to get to the dialog box shown in Figure 28.6.

FIGURE 28.6

The Database Documenter dialog box lets you view and print technical behind-the-scenes information on any object in a database.

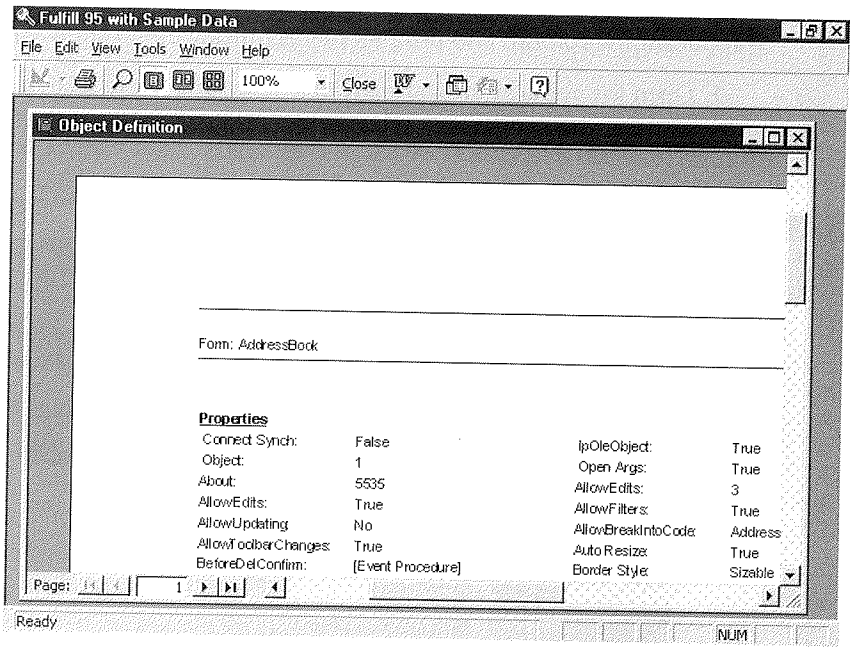


3. Click the tab for the type of object you want to explore and then click the checkboxes for the objects you want to document. Be aware that any given object can produce a lot of documentation, so you may want to document just one or two objects at a time.
4. (Optional) Click on the Options button and specify exactly what you want to print. Then choose OK after setting your options.
5. Choose OK and give Access a few minutes to prepare the documentation. When the documentation is ready you'll be taken to the report preview screen showing a window named Object Definition (Figure 28.7).
6. Click on the Print button in the toolbar to print the documentation.

When you've finished you can just click on the Close button in the Object Definition window to return to the database window.

FIGURE 28.7

The Object Definition window displays technical information about one or more objects in a database.



Modifying Existing Applications

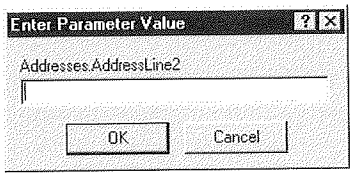
Once you get to the source code (i.e., database window) of an Access application, the temptation to start tweaking it to better meet your own needs will be strong. In fact, if

you can get your hands on the source code for an application, the implication is that you want to modify the application. We have some advice to give you on modifying existing applications.

Modifying an existing application is a terrific way to help ease the transition from toolmaster to artist. However, if you try to customize an existing application before you've become an Access toolmaster, then you're very likely to find yourself in deep waters very quickly. Just the simple act of deleting or renaming a single object—even one little field in a table—can wreak havoc throughout the rest of the application. Here's why.

Let's suppose we have a field named AddressLine2 in a table in an application. You decide you don't need that field, so you delete it from the table or you change its name to PhysicalAddress. You save the change and all seems fine.

But then as you start using the application, you keep coming across little *#Name?#* and *#Error#* messages, or you keep seeing a little window like this popping up on the screen:



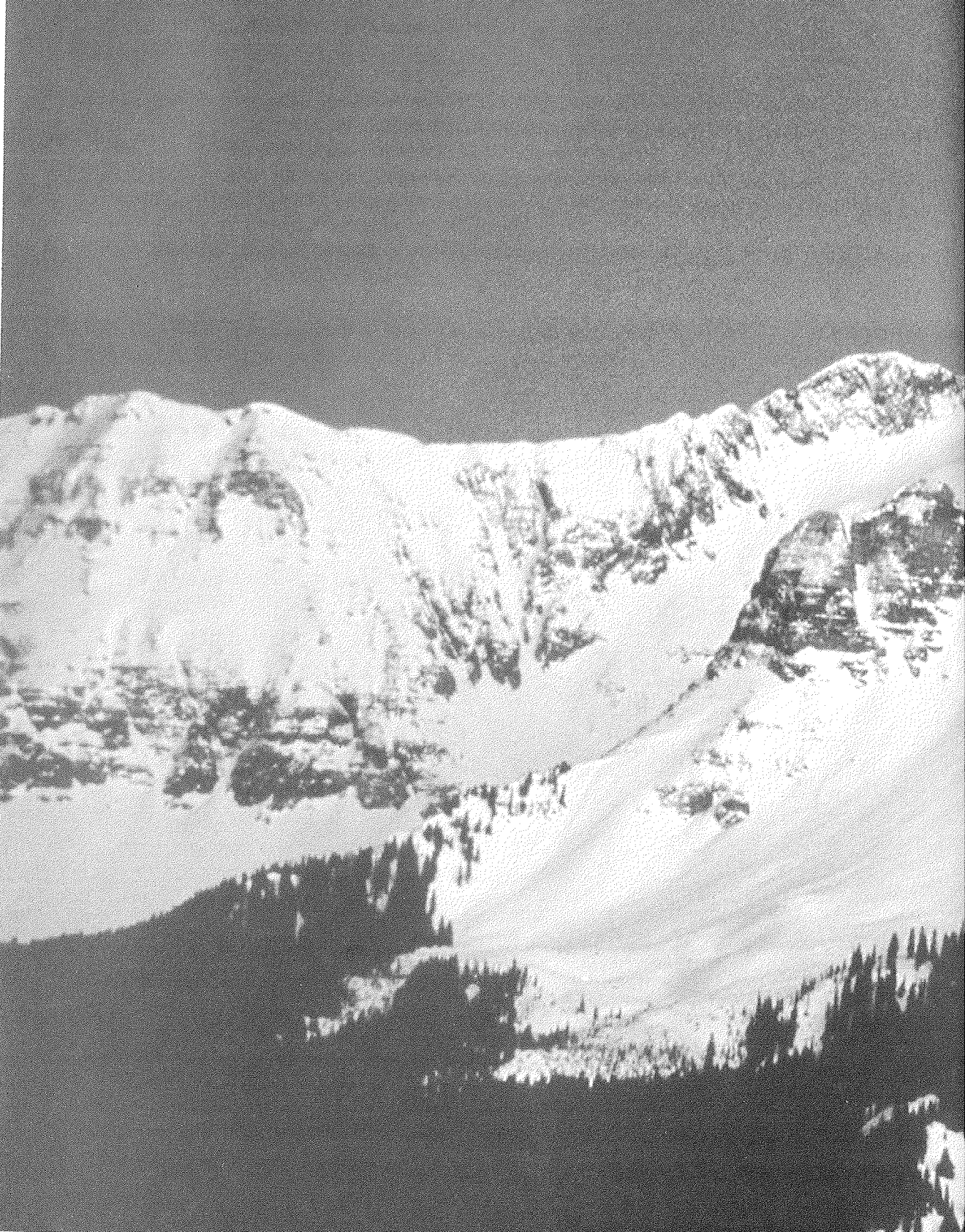
The problem is that other objects—queries, forms, reports, macros, and modules—might "expect" to find a field named AddressLine2 in that table. When you delete or rename that field in Access's table design view, *that change does not carry over to any other objects in the database.* So when you use one of those objects, it just flat-out doesn't work.

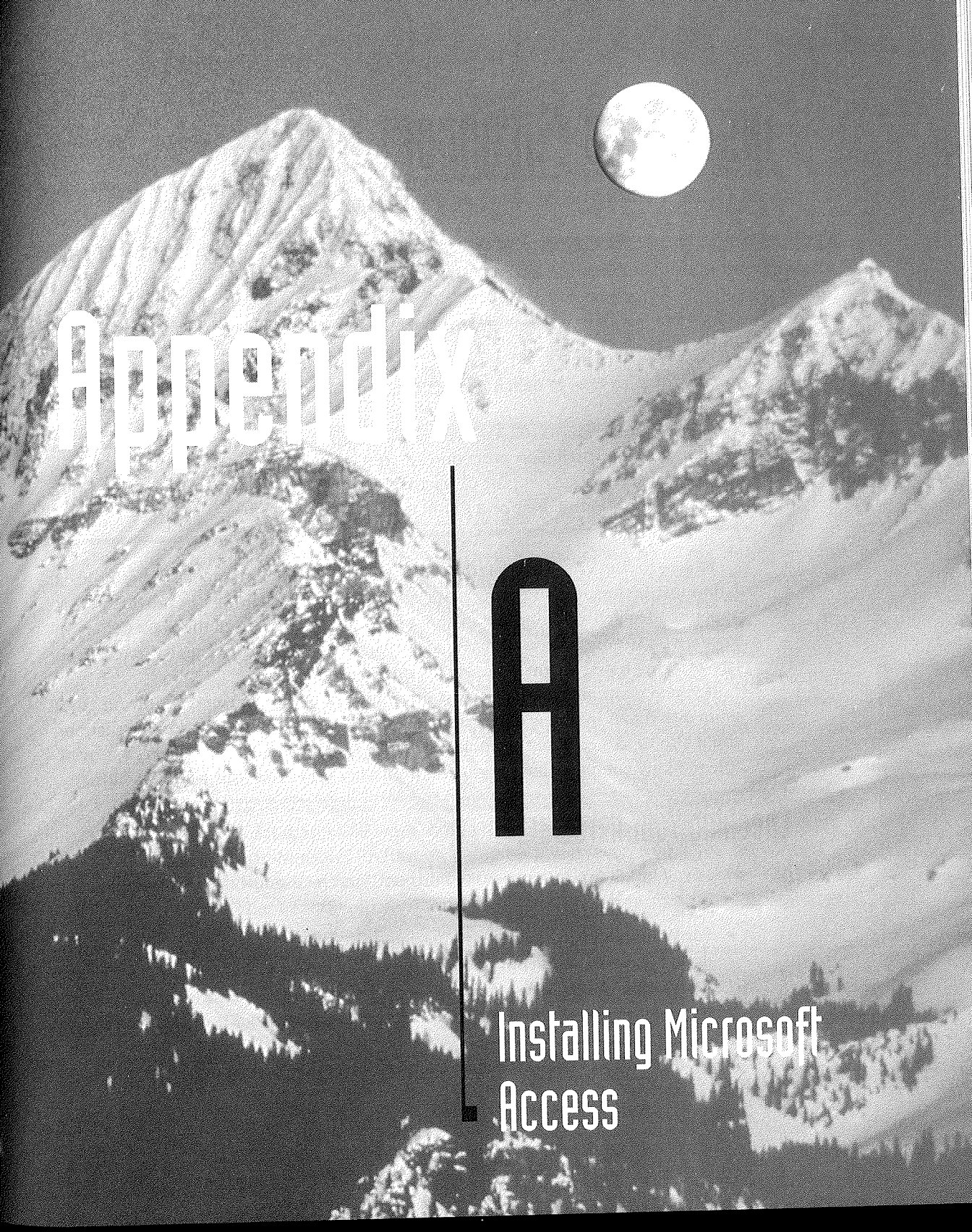
Where to Go from Here

Exploring existing applications is just one way to make the transition from Access toolmaster to Access artist (aka *application developer*). You can also find plenty of books to help you that pick up where this one leaves off. An excellent book on macros is *Access 97 Macro and VBA Handbook* by Susann Novalis, published by Sybex.

Another great resource for information, add-on products, training, and more is the *Access/Visual Basic Advisor* magazine. For information and pricing contact *Access/Visual Basic Advisor* at P.O. Box 469030, Escondido, CA 92046-9918. (Phone (619) 483-6400 or (800) 336-6060 or fax (619) 483-9851.) Or you can reach the publisher at CompuServe 70007,1614 or `70007.1614@compuserve.com` on the Internet.

As for us, the authors of *Mastering Access 97 for Windows 95*, we thank you for reading and wish you all the best in your endeavor to become a full-fledged Windows 95 application developer. We know from experience that mastering this complex tool takes a heck of a lot of time, patience, and brain cells. But the creative prowess that mastery buys you is well worth the suffering.




Appendix
A
Installing Microsoft Access

Installing Microsoft Access

You must install Microsoft Access 97 for Windows 95 before you can use it to manage your data. This appendix briefly explains how to install Access on a single-user computer.

NOTE

The single-user installation procedures might differ slightly, depending on whether you're installing Access as part of Microsoft Office Professional and whether you're installing from floppy disks or a CD-ROM.

Be sure to check the materials that come with your Access program for specific installation instructions, late-breaking news, and more details than we can provide in this appendix. For general information about installing Access, check the book *Getting Results with Microsoft Access 97 for Windows 95,* version 8.0, that comes with Access. For late-breaking news, see the `Acreadme` file on your setup disk.

If you're planning to install Access on a network, see the `network.txt` file on your setup disk for an overview of the steps. For detailed network installation instructions, you'll also need a copy of the book *Microsoft Office For Windows 95 Resource Kit,* available in many computer bookstores and from Microsoft Press (call 1-800-MS PRESS).

Checking Your Hardware

Before you install Access, you need to find out if your hardware meets these requirements:

- A PC with at least a 486 DX2 or DX4 processor.
- For Windows 95, we strongly recommend at least 16MB memory (RAM). For Windows NT, we recommend 32MB memory.
- About 42MB of hard disk space available for a full installation (about 33MB for a typical install and about 14.8MB for a compact install). Of course, you'll need extra space for your databases.
- Windows 95 or Windows NT 3.51 (either Server or Workstation).

- A VGA or higher monitor.
- A Microsoft mouse or compatible pointing device.

Yes, we know the box states lower memory and processor requirements, but those more modest recommendations may result in sluggish performance when you use Access.

NOTE

If you're installing Microsoft Access on Windows NT, you must be the Administrator or have Administrator privileges.

Preparing Your 1. *X*, 2. *X*, and Access 95 Databases

Please read this section if you have old Access version 1.0 or 1.1 (1. *x*), version 2.0, or version 7 (Access 95) databases. If you're installing Access for the first time and do not have any old databases, feel free to skip ahead to the next section.

To use your old databases in Microsoft Access 97 for Windows 95, you can either convert them to the new format or use them in their original format.

- **Convert the old databases to the new format.** Doing so allows you to take full advantage of all the new features in Access. However, you should not convert your databases if others must use them with Access 1. *x*, 2.0, or 7 (Access 95).
- **Use the old databases *without* converting them to the new format.** This method, called *enabling* the databases, allows people to view and update the databases with older versions of Access. However, it does not let you change database objects or create new ones in Microsoft Access 97 for Windows 95. In secure databases (see Chapter 18), you also can't change or add permissions unless you convert the old database to the new format.

WARNING

In general, databases will work more efficiently if you convert them to Microsoft Access 97 for Windows 95 format. Remember, however, that if you do convert your databases, they can't be used with Access 1.*x*, 2.0, or 7, and they can't be converted back to the old versions. Of course, you can restore the old databases from backup copies you made before you converted.

Regardless of whether you decide to convert or enable your old databases, you must do these quick preparation steps *before* removing your previous version of Access.

1. Back up any databases you plan to convert. *Do not skip this step!*

2. If you're using Access 2.0 or 7 (Access 95), skip to step 3.

   Or

   If you're using Access 1.*x*, you should be aware of and compensate for the following:

   - **Backquote character (`) in object names.** Backquotes aren't allowed in Microsoft Access 97 for Windows 95 object names, and they'll prevent you from converting your database or opening the object in Access 97 for Windows 95. You must use Access 1.*x* to rename such objects before converting or using them in Access for Windows 95.
   - **Indexes and relationships.** Microsoft Access 97 for Windows 95 tables are limited to 32 indexes each. You won't be able to convert your databases if any tables exceed that limit. If necessary, delete some relationships or indexes in complex tables before converting the database.
   - **Modules named DAO, VBA, or Access.** Modules with these names will prevent a database from converting because they are names of "typelibs" that Access references automatically. You'll need to change these module names before converting any databases that use them.

3. Use Access 1.*x*, 2.0, or 7 (Access 95) (as appropriate) to open your database and then open all of your forms and reports in design view. Click on the Module tab of the database window.
4. Open any module in design view and then choose Run ➤ Compile Loaded Modules. If your database doesn't contain any modules, click on the New button to create a blank module and then choose Run ➤ Compile Loaded Modules. This step will ensure that all of your modules are fully compiled.
5. Close and save all of your forms, reports, and modules and then close your database.
6. Repeat steps 2 through 5 for each database you want to convert.
7. Back up your databases again to preserve your work.

Now you're ready to install Microsoft Access 97 for Windows 95. After installing Access, you can convert or enable your databases, as explained later in this appendix under "Enabling and Converting Access Databases."

For more information about conversion issues, please see Appendix A of the *Building Applications with Microsoft Access 97 for Windows 95* manual that comes with Access and look up topics under *Converting* in the Microsoft Access Answer Wizard. You also may want to search the Microsoft Knowledge Base for late-breaking information on conversion issues. The Knowledge Base is available on CompuServe (GO MSKB), the Microsoft Network (use the Find tools to search for Knowledge Base), and the Microsoft Internet servers at `www.microsoft.com` and `ftp.microsoft.com`. Chapter 4 offers more details about the Microsoft Knowledge Base and other sources of technical information about Access and other Microsoft products.

Installing Access on a Single-User Computer

Installing Access for use on a single computer is quite easy. Here are the basic steps:

1. Start Windows 95 and make sure other programs are not running.

WARNING

When installing Access, be sure to do so on a "vanilla" system. Exit any nonstandard Windows shells or memory managers, any terminate-and-stay-resident (TSR) programs, and any virus-detection utilities. Check the Windows Taskbar and close any programs that are running. If any programs are started from your `autoexec.bat` and `config.sys` files, restart your computer by choosing Start ➤ Shut Down ➤ Restart the Computer ➤ Yes. When you see the message *Starting Windows 95*, press F8 and choose Step-By-Step confirmation. You'll be asked whether to run each command in `config.sys` and `autoexec.bat`. Press Y to run the command or N to bypass it.

2. Insert the Microsoft Access Setup Disk 1 into floppy drive A or B of your computer. Or insert the Microsoft Office Professional CD-ROM into your computer's CD-ROM drive.
3. Click on the Start button on the Windows Taskbar and then choose Run.
4. In the Command Line text box, type the name of the drive you put the setup disk in (for example, **a:** or **b:** or **d:**), followed by **setup** (or use the Browse button to search for the Setup program). For example, if you put Setup Disk 1 in drive A, type

```
a:setup
```

5. Click on OK or press Enter.

TIP

As an alternative to steps 2 through 5, you can choose Start ➤ Settings ➤ Control Panel, double-click on Add/Remove Programs, and then click on the Install button to start the Install Program from Floppy Disk or CD-ROM Wizard. When the Wizard starts, follow the instructions on the screen.

6. Follow the initial instructions that appear on the screen. Among other things, you'll be asked to supply your name and serial number and given a chance to choose an installation folder for Microsoft Office. In most cases, you can fill in a dialog box (if necessary) and then press Enter or click on OK to go to the next dialog box.

To learn more about which components an installation option will set up or what an installation button will do, click on the ? button at the upper-right corner of any Microsoft Access 97 For Windows 95 Setup dialog box and then click on the place you're interested in.

7. Click on one of these buttons when asked to choose the type of installation you want:

   Typical Installs Access with the most common options. This option is the default choice. Requires about 33MB of available disk space.

   Compact Installs Access with the bare-bones components needed to run the program. This option is useful if you're using a laptop computer or if your hard disk is very short of space. Requires about 14.8MB of disk space.

   Custom Allows you to select the installation options you want to use. Choose this option if you want to install all of Access or if you're short on disk space and want to install only certain components of Access. A full Custom install requires about 42MB of available disk space. After choosing the Custom option, you can click on the Select All button to check all components, or you can select (check) or deselect (clear) components as needed. When you're finished choosing components, click on Continue. Figure A.1 shows a Microsoft Office 97 Microsoft Access dialog box that's similar to the one you'll see. Table A.1 briefly describes the main components you can select.

It's best to select (check) *all* the components listed in Table A.1 if you have room for them on your hard disk. The check mark options are toggles: simply click on a checkbox to select or deselect it. If necessary, you can install omitted components or remove unwanted components later, as described under "Running Setup to Add or Remove Components."

   Run From CD or **Run From Network** Installs only those files needed to run Access from the CD-ROM or from the network.

8. If you're installing from floppy disks, feed the appropriate disk into the floppy disk drive whenever Access Setup asks you to, and then press Enter or click on OK to continue with the installation.

FIGURE A.1

The Custom installation option lets you choose which Access components to install. If you have enough disk space, it's easiest to click on the Select All button.

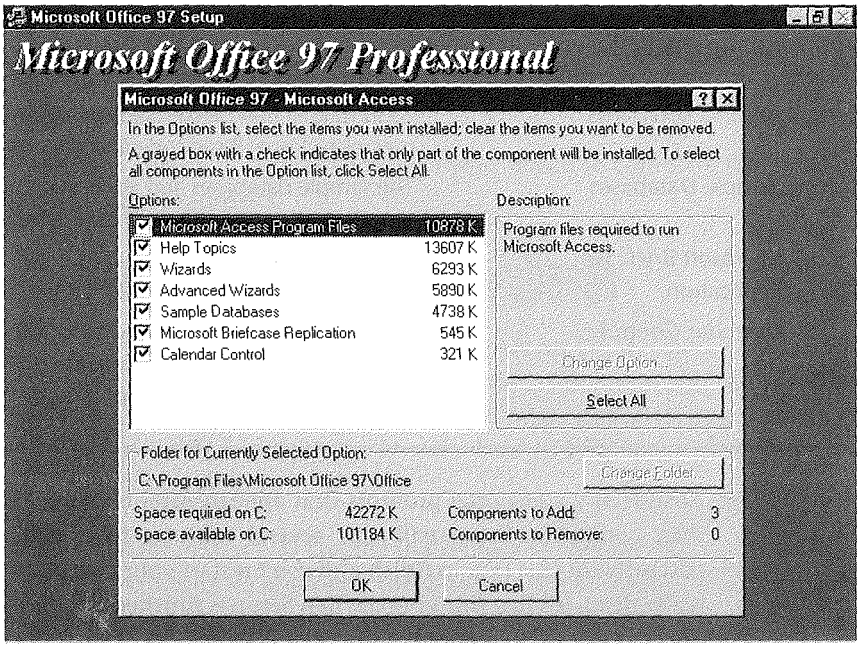


TABLE A.1: MICROSOFT ACCESS COMPONENTS THAT YOU CAN INSTALL

COMPONENT	DESCRIPTION
Microsoft Access	The Access program and minimum utilities needed to run it. This option is required for an initial installation of Access.
Help Topics	Online Help and language reference files. To get the most out of Access, be sure this option is selected.
Wizards	The easy to use question-and-answer tools that help you create tables, forms, reports, queries, and so on. It's best to leave this option checked so you're sure to get all the built-in Wizards. You can use the Add-In Manager (described in Chapter 15) to install or uninstall Wizards that you've purchased separately from Access.

Continued

TABLE A.1: MICROSOFT ACCESS COMPONENTS THAT YOU CAN INSTALL (CONTINUED)

COMPONENT	DESCRIPTION
Advanced Wizards	Tools that let you perform advanced tasks like converting macros to Visual Basic code and documenting databases.
Sample Databases	The sample databases and applications that Microsoft supplies with Access, including Northwind and Solutions.
Microsoft Briefcase Replication	The feature that lets you keep different copies of databases in sync (see Chapter 17).
Calendar Control	The OLE custom control and associated help files that let you create fully programmable calendars in your forms and reports (see Chapter 13).

9. Answer any additional prompts that appear. After copying all the needed files to your computer, Setup will take several minutes to update your system. Wait patiently for this task to finish and *don't* restart your computer (the computer isn't really dead, it's just thinking). When Setup is finished, you'll see a message indicating that the setup completed successfully.
10. Click on the OK (or Online Registration) button to finish the job.

After installing all the files you need, Setup will create a Microsoft Access option on the Start ➤ Programs menu. When you've completed the installation, go to Chapter 1 of this book, where you'll learn how to start Microsoft Access and get around in it.

Running Setup to Add or Remove Components

If necessary, you can run the Setup program again to add Microsoft Access components that you chose not to install initially or to delete components that you no longer need. To add or remove components, follow these steps:

1. Start the Setup program from the Microsoft Access Setup Disk 1 or Microsoft Office 97 Professional CD-ROM, as explained earlier.
2. Click on one of these buttons in the next dialog box:

   Add/Remove Takes you to a screen similar to the one shown earlier in Figure A.1. From here you can select components you want to add or deselect components you want to remove.

   Reinstall Repeats the last installation, replacing any accidentally deleted files and restoring initial settings.

Remove All Removes all previously installed Access components from your hard disk.

Online Registration Lets you use your modem to register your software with Microsoft.

Exit Setup Lets you exit the Setup program without making any changes.

3. Follow any instructions that appear on your screen to complete the installation or removal of components.

Enabling and Converting Old Access Databases

As mentioned earlier, you can convert your old Access 1.*x*, 2.0, and 7 (Access 95) databases to the new Access 97 for Windows 95 database format, or you can just *enable* the databases for use with Access 97 for Windows 95 without converting them.

Enabling a Database

To enable a database that was created in an older version of Access, start Microsoft Access 97 for Windows 95 and open the database (see Chapter 1 if you need details). You'll see a Convert/Open Database dialog box that explains your options, as shown in Figure A.2.

1. Choose Convert Database to convert the database to the new format or choose Open Database to enable the database (leaving it in the old format).
2. Click on OK and follow any instructions that appear.

Your database will be either converted or enabled, depending on your choice in step 1.

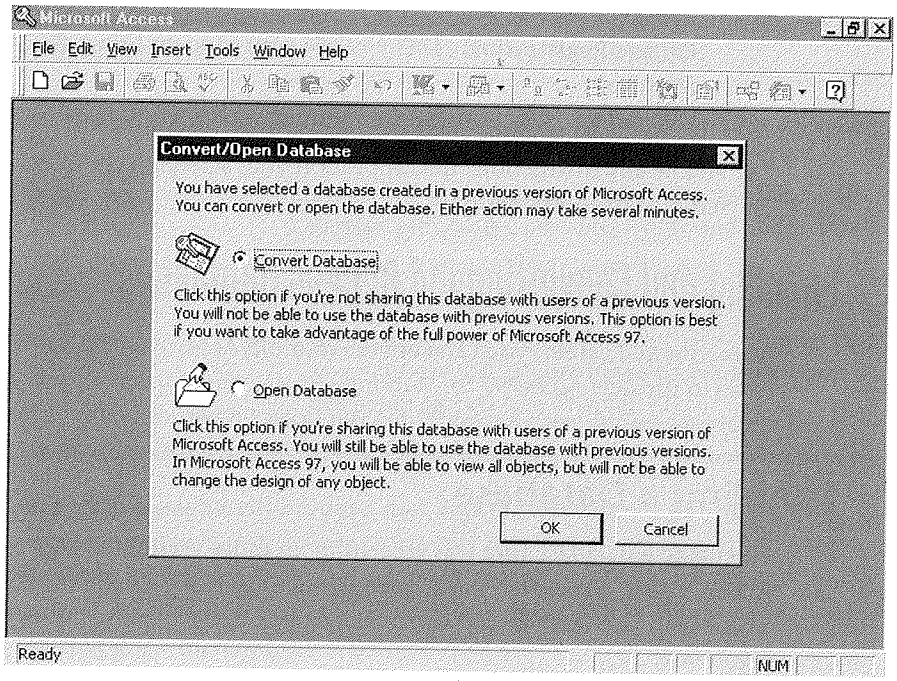
Converting a Database

To convert a database, you can open it and choose the Convert Database option described above, or you can follow these steps:

1. Close all open databases. If you're using a network, make sure that no one else has opened the database you want to convert.
2. Choose Tools ➤ Database Utilities ➤ Convert Database.
3. Locate and click on the database you want to convert in the Database To Convert From dialog box. If you also want to convert any old-style OLE objects (such as Graph 3.0 objects), select (check) the Convert OLE option below the Advanced button. Click on the Convert button.

FIGURE A.2

The Convert/Open Database dialog box lets you choose whether to convert a database to the new Access format or open (enable) it in its old format.



4. Type a new name for the database in the Database To Convert Into dialog box or just select a different directory location if you want to keep the same name. Click on Save.
5. Respond to any prompts that appear while Access converts your database. When conversion is complete, you can open the new database and use it normally.

After converting (or enabling) your old database, you may discover some changes in the way database objects behave; you may need to tweak your applications to make them work more smoothly.

WARNING

Once you convert a 1.*x*, 2.0, or 7 (Access 95) database to Access 97 for Windows 95 format, you can't open that database in version 1.0, 1.1, or 2.0 and you can't convert it back.

If your database is very large, the conversion steps given above may fail (it's rare but possible). If that happens, create a new blank database and then try using the Import procedures discussed in Chapter 7 to import just a few objects (about 20 to 30) at a time. The Import process will convert the objects as needed.

The steps given above apply to the simple case of converting an unsecured database without linked tables on a single-user computer. For more information about conversion issues, see Appendix A of the *Building Applications with Microsoft Access 97 for Windows 95* manual that comes with Access and look up *Converting* with the Office Assistant.

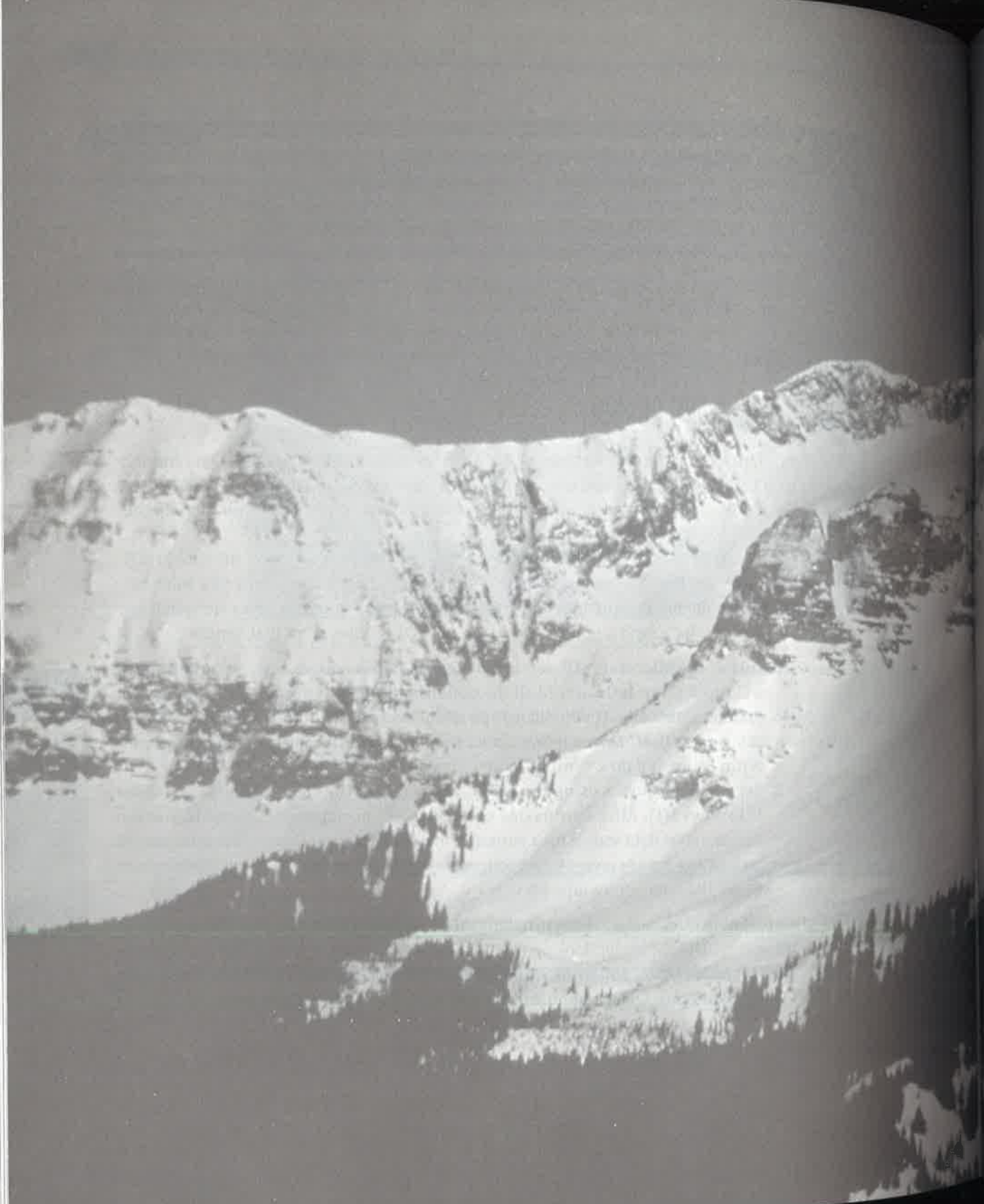
Setting Up ODBC Support

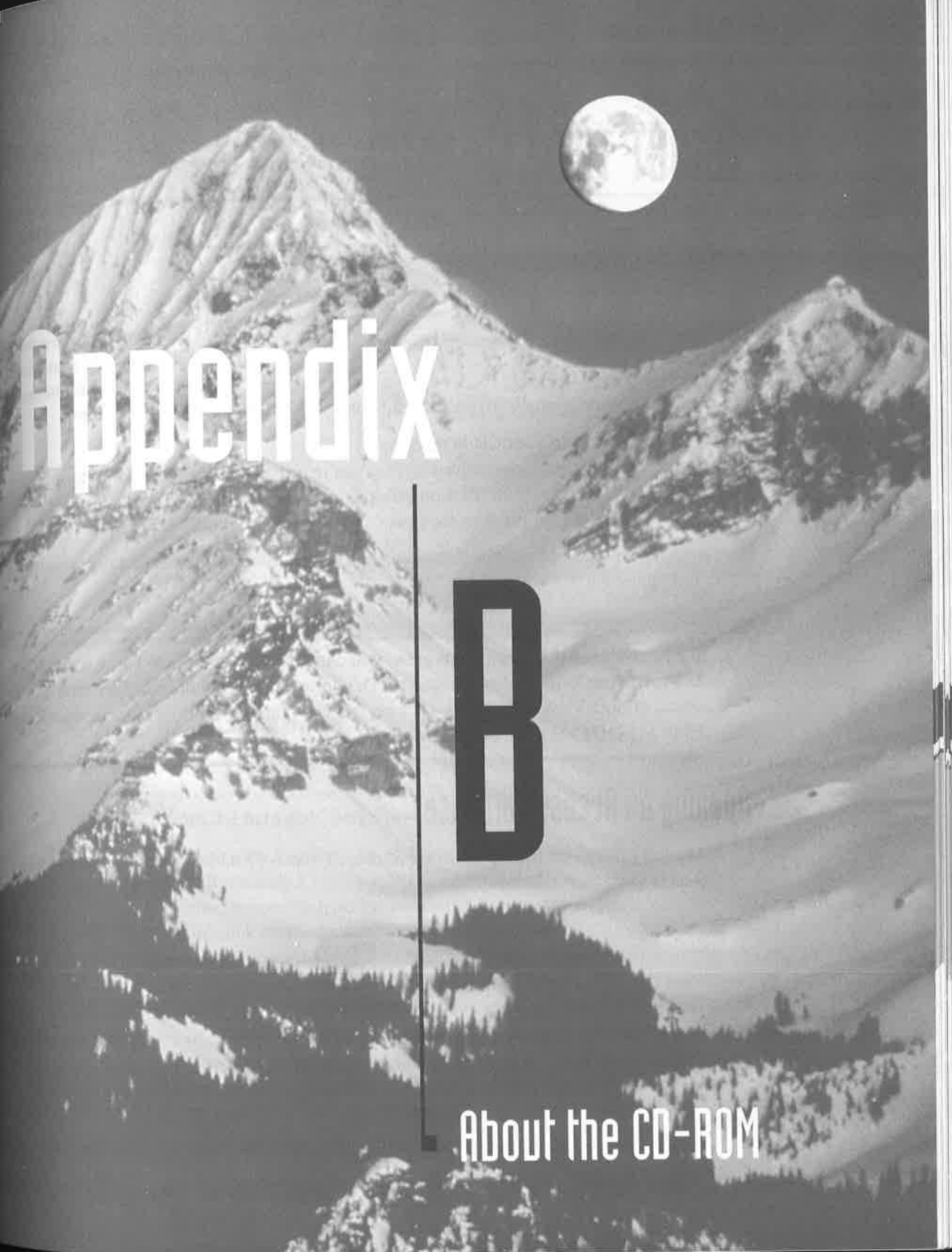
The Open Database Connectivity (ODBC) drivers that come with Access let you connect to SQL database servers and use SQL databases from Access. (If you're not interested in using SQL databases, you needn't bother to set up ODBC support and you can skip the rest of this appendix.)

Getting your machine ready to use SQL database servers and to use the data in SQL databases involves the two main steps given below. These steps assume you want to work with databases stored on a Microsoft SQL server; if your databases are stored on another vendor's server, you'll need to use an ODBC driver from that vendor.

1. **Install the Microsoft SQL server driver that comes with Access.** If you performed a Custom setup and checked all the options, the ODBC drivers will be installed for you automatically. If you did a Typical or Compact setup or cleared the *Microsoft SQL Server ODBC Driver* option under *Data Access* in the Custom setup, you must rerun Setup and do a Custom setup to install those drivers.
2. **Set up data sources using the ODBC Manager (or the ODBC Administrator for Windows NT).** After starting one of these ODBC management programs, you can define a new data source for a currently installed driver or change the definition of an existing data source. (A data source is a set of instructions that tell ODBC where to get the data you're interested in.)

For more information about installing ODBC support and setting up data sources using the ODBC Manager, look up *ODBC* in the Access Answer Wizard and then double-click on *Install ODBC Drivers And Setup ODBC Data Sources* under *How Do I,* or start the ODBC Manager or ODBC Administrator and then click on its Help button.





Appendix B

About the CD-ROM

About the CD-ROM

The CD-ROM that comes with this book contains sample databases, shareware, freeware, demos, and a multimedia catalog for you to enjoy. To see the contents of the CD-ROM, follow these steps:

1. Put the CD in your CD-ROM drive.
2. Close any open applications so that you're just at the Windows 95 desktop.
3. Double-click on your My Computer icon.
4. Double-click on the icon for your CD-ROM drive (typically D:).

The contents of the CD are displayed in a My Computer window, looking something like Figure B.1.

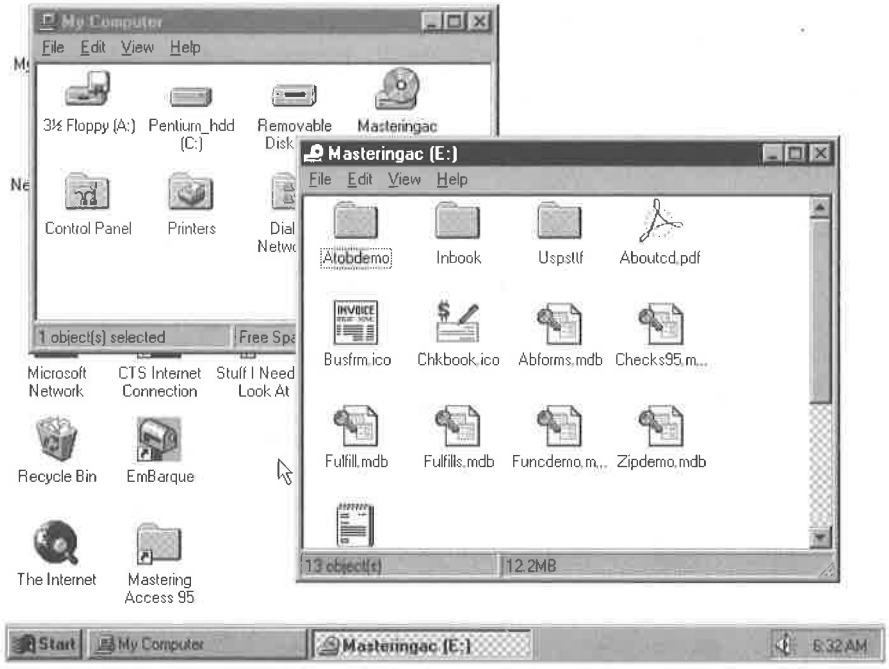
If you don't see the Large Icons view, you can just choose View ➤ Large Icons from the menu bar in the "Masteringac" or D: window. To show or hide file name extensions, choose View ➤ Options, click on the View tab, then either clear or check the Hide MS-DOS file extensions option.

Opening an Access Database

Most of the files on the CD-ROM are Microsoft Access 97 databases, as indicated by the Access icon, shown below.

If you try to open a database directly from the CD, you'll get a read-only error message. You will have to copy the databases to your hard drive and take off the read-only attribute before you can use them. Step-by-step instructions for doing the copy and changing the read-only attribute follow.

URE B.1
contents of
e CD-ROM
k displayed
a My Com-
er window,
with Large
Icons view.



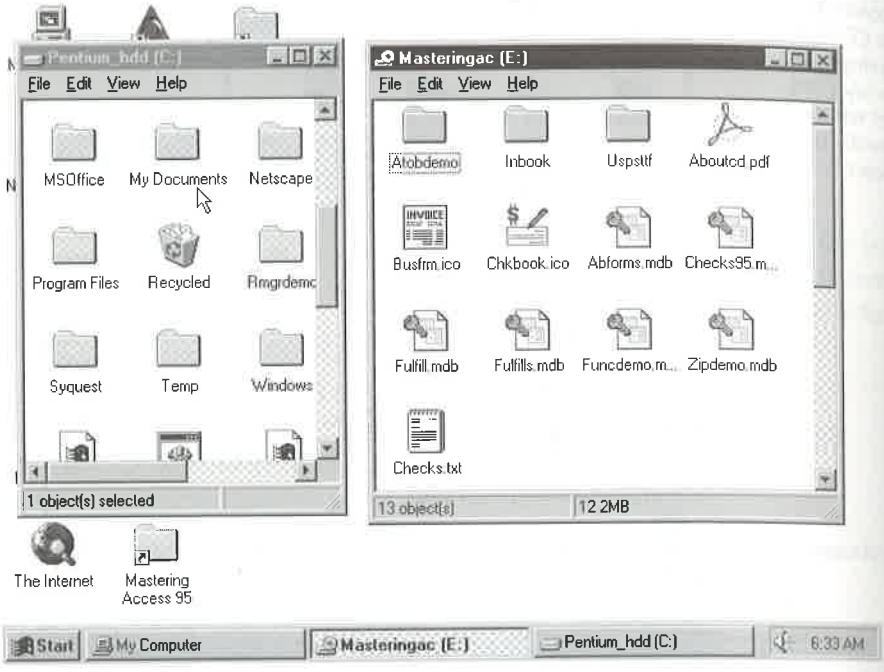
Copying a Database to Your Hard Disk

If you want to be able to add, change, and delete data in one of the sample databases, you must copy the database to your hard disk, change its read-only attribute, and then run it from the hard disk. Here's how:

1. Exit out to the Windows 95 desktop if you're in Microsoft Access.
2. Open (double-click on) your My Computer icon.
3. Double-click on the icon for your hard disk (typically C:).
4. Double-click on the icon for your CD-ROM drive.
5. Size and position the windows for the hard drive and CD-ROM so that you can see both the destination folder (the folder you want to copy *to*) and the object you want to copy. For example, in Figure B.2 you can see the folder named My Document on drive C: (where you want to copy to) and several databases from the CD-ROM.
6. Drag and drop the icon for the database you want to copy from the CD-ROM drive's window over to the destination folder's icon.

FIGURE B.2

The destination (My Documents in drive C: in this example) and databases from the CD-ROM visible on the Windows desktop.



7. Double-click the destination folder's icon to open a window for it. Right-click the database name and choose Properties. Then uncheck the read-only attribute and click OK.

That's all there is to it. To run the database from your hard disk:

1. Open (double-click on) the folder on your hard drive that contains the database (My Documents in this example).
2. Double-click on the icon for the database you want to open.

You won't get the read-only message this time because you've opened the database on a "normal" read-write disk.

NOTE

Remember that the ROM in CD-ROM stands for "read-only memory." You cannot change anything that you open directly from the CD-ROM. You can only change items that you copy to, and run from, your hard disk.

In the sections that follow we'll describe the contents of each database.

Abforms and Checks95

The Abforms and Checks95 databases are freebies from Cary Prague Books and Software. If you want to actually use the Checks95 database, copy both the Checks95 (or `Checks95.mdb`) file and the Chkbook (or `Chkbook.ico`) file, using the basic technique described under "Copying a Database to Your Hard Disk" earlier in this appendix.

For more information, see "Cary Prague Books and Software" later in this appendix or the file named Checks (`Checks.txt`) on the CD.

Fulfill and FulfillS

The databases named Fulfill and FulfillS contain a sample data-entry database application that you're free to use, abuse, ignore, explore, and tinker with at will. The difference between the two is as follows:

FulfillS (or `FulfillS.mdb`) The Fulfill database application with some fake data already entered. Explore from the CD or copy it to your hard disk if you want to add, change, or delete data.

Fulfill (or `Fulfill.mdb`) Same as FulfillS but without the fake names and addresses, products, and orders. To use this version, first copy it to your hard disk.

Appendix C contains more information about using Fulfill 95 with its many special bells and whistles. Chapter 28 explains how to explore Fulfill 95 "behind the scenes" to see what makes it tick and to learn some tricks for your own custom applications.

ZipDemo and FuncDemo

The ZipDemo (or `ZipDemo.mdb`) database illustrates a Microsoft Access *event procedure* that does something very useful: after you type in a zip code, it autofills the city, state, and area code fields on the same form. The Fulfill databases use the same technique to do the same job.

The FuncDemo (or `FuncDemo.mdb`) database illustrates sample *function procedures* (also called *user-defined functions*) written in Visual Basic:

NumWord( ) Converts a number, such as *123.45*, to words, such as *One hundred twenty three and 45/100*. Cary Prague's Check Writer application uses this function to print checks.

Proper( ) A function that Access forgot. Converts any case (i.e., *alan simpson, ALAN SIMPSON, aLaN SiMpsOn*) to proper noun case (i.e., *Alan Simpson*).

You can open and explore the ZipDemo and FuncDemo databases right from the CD: there's no need to copy either to the hard disk.

The Inbook Folder

The `Inbook` folder contains some small sample databases used in examples throughout the book:

- Lessons (or `Lessons.mdb`) used in the hands-on lessons at the start of this book.
- OrdEntry (or `OrdEntry.mdb`) used heavily throughout the first few chapters.
- StarSrch (or `StarSrch.mdb`) illustrates the use of photos in a database.
- Chap 22 (or `Chap22.mdb`) shows a small custom dialog box.

Access to Business Contacts Unlimited Demos

Access to Business offers great products for Access users and is a valuable resource for everyone from beginner to seasoned pro. We've included their Contacts Unlimited demos on the CD.

Using the Access to Business Demos

To view the demos, do the following:

1. Close any open applications so that you're at the Windows desktop.
2. Put the CD-ROM that came with this book into your CD-ROM drive.
3. Click on the Start button and choose Run.
4. Type ***d*:\atobdemo\cueval7\Cudemo7.exe** in the text box shown below, but replace ***d*:** with the drive letter of your CD-ROM drive.

5. Choose OK.

It might take a minute for the catalog to load from the CD. But when it does, you'll see the opening screen. Work your way through to the Contents screen shown in Figure B.3.

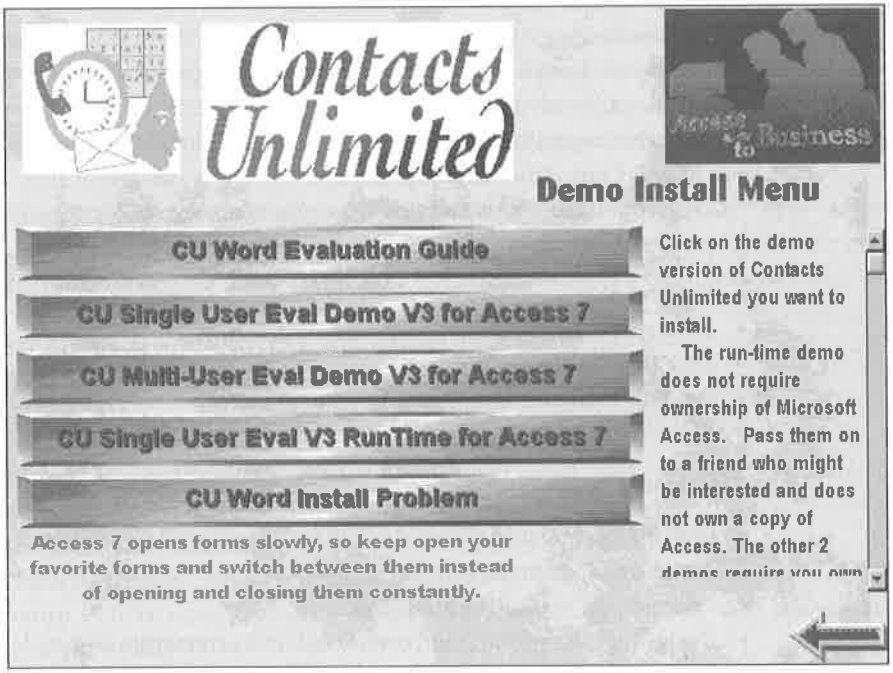


FIGURE B.3 *The opening screen for the Access to Business multimedia catalog of Microsoft Access-related products.*

You can install any of the three demos of Contacts Unlimited and a Word evaluation guide as well.

Should you have any questions or comments, please direct them to Access to Business at:

Access to Business
16903 SE 39th St.
Bellevue, WA 98008-5825, USA
Telephone: (206) 644-5977
Fax: (206) 641-9271
E-mail: `atob@a-to-b.com`
World Wide Web: `http://www.a-to-b.com`

The remainder of this section was contributed by Access to Business.

About Access to Business

Access to Business markets and develops Microsoft Office and Microsoft Access–related products for all business types. Our main product is Contacts Unlimited, a full-featured contact manager written completely in MS Access.

We offer a Super Demo CD for $29.95 that contains more than 25 demos. We accept Visa, MasterCard, American Express, and company checks.

Access to Business is also a full-service graphic design company offering these services:

- Company office stationary designed and printed
- Product and company logo design
- Product package design and production
- Disk label design and production
- Product manuals, brochures, newsletters
- Multimedia product demos

Marketing support includes:

- Possible partnership with us or others to bring your product to market
- Assisting in producing online help
- Advising on marketing strategies
- Designing and producing PowerPoint and interactive demos

We already have an international partner in Australia and Canada and are negotiating with several other countries. We are looking for national and international partners to sell some or all of our product line. Our products are available to qualified resellers, consultants, and developers.

Some of our products are available for developer vertical licenses. You can use these products as an engine to develop an entirely new product with a new name and purpose. Contact us for more details and pricing.

The Access to Business name and logo are trademarked. Call for complete current pricing.

Cary Prague Books and Software

Cary Prague Books and Software is another valuable resource for Access users, and the company has graciously given us the rights to distribute its Check Writer database application free of charge. Also provided are some examples of the company's predesigned business forms.

Using the Check Writer Database

To use Check Writer, copy Checks95 (or `Checks95.mdb`) and Chkbook (or `ChkBook.ico`) to the My Documents folder, or to some other folder of your choosing, on your hard disk, and make sure its read-only attribute is unchecked. (See "Copying a Database to Your Hard Drive" earlier in this chapter for step-by-step instructions.) Then run Checks95 from your hard disk.

When Check Writer starts up, just follow the instructions on the screen to get around. Be sure to choose Convert Database when you are given the opportunity so you can try out the program. Figure B.4 shows one of the sample screens from Check Writer.

From the Check Writer Readme File

Check Writer for Microsoft Access 95 © 1995 Cary Prague Books and Software.

The main Check Writer switchboard contains four basic functions:

- **Check Writer/Register** Add, change, delete, or print checks, deposits, and adjustments, or display the check register.
- **Check Reconciliation** Display the Check Reconciliation system.
- **Bank Accounts** Add, change, or delete Bank Account information.
- **Setup** Enter your company information, recurring payees, and default bank account number.

The Check Writer comes with two sample bank accounts and a selection of transactions to get you started. You can practice with those and then create your own accounts.

FIGURE B.4

Cary Prague Books and Software's Check Writer database application.

You start by creating your own bank account (Bank Accounts icon) and then make that bank account the default bank account (Setup icon). Once you have done that, you can open up the Check Writer and enter transactions.

Use the buttons at the bottom of the form to navigate from record to record or between functions. Click on the Register icon to display the check register. In the register, click on the Check icon to return to the check writer form to enter or edit checks.

You can change the account being viewed at any time using the combo box at the top of the Check Writer form. You can only change the transaction type for new checks and can choose between checks, deposits, and a number of adjustments that you can add to by using the Setup icon.

You can enter the payee for a check or select from the combo box in the payee line of the check. You can add more recurring payees in the Setup screen.

When you mark a check as void, a void stamp will appear on the check and it will not be counted in the check register or check reconciliation.

The area below the check is the voucher stub and anything you enter is printed when you select certain types of checks.

Press the Find button to display a dialog box showing five ways to find a check, deposit, or adjustment.

Press the Print button to display a dialog box allowing you to print the current check/deposit/adjustment, marked checks, a range of checks by date, or a range of checks by check number. You can modify the reports that print the checks for custom paper.

The Check Writer is normally $79.95. Mention *Mastering Microsoft Access 97 for Windows 95* and you can buy the documentation for only $49.95. To order full documentation or purchase tech support call us at (860) 644-5891, fax us at (860) 648-0710, or e-mail us at `71700,2126` or send mail to `CARYP` on the Microsoft Network or to CaryP@msn.com from the Internet.

Sample Business Forms

Cary Prague Books and Software has also provided some samples of the company's predesigned business forms. To view them:

1. Copy `abforms.mdb` to your hard drive and make sure its read-only attribute is unchecked, as discussed near the start of this appendix.
2. Open the database and click on the Forms tab in the database window.
3. View a sample form by double-clicking on its name in the database window.

The purpose of the abforms database is simply to show you a small sampling of ready-to-use business forms that you can purchase from the vendor. The section that follows was contributed by the vendor, Cary Prague Books and Software.

More from Cary Prague Books and Software

We also have fully customizable Payroll for Microsoft Access for only $129.95. You can also purchase our fully customizable business accounting product named *Yes! I Can Run My Business* at $50 to $200 off of our regular prices:

Single User Edition: $299.95 Only $249.95
Unlimited Multiuser: $399.95 Only $349.95
Royalty Free Developer: $999.95 Only $799.95

You can also request a free catalog of all our products and services including books, videos, and add-on software for Windows 95, Microsoft Access 2.0 and 95, Visual FoxPro, and Visual Basic 4.0.

Cary Prague Books and Software
60 Krawski Drive
S. Windsor, CT 06074
Telephone: (860) 644-5891
Fax: (860) 648-0710
CIS: 71700-2126
Internet: `71700,2126@compusrv.com`
Microsoft Network: `CARYP`

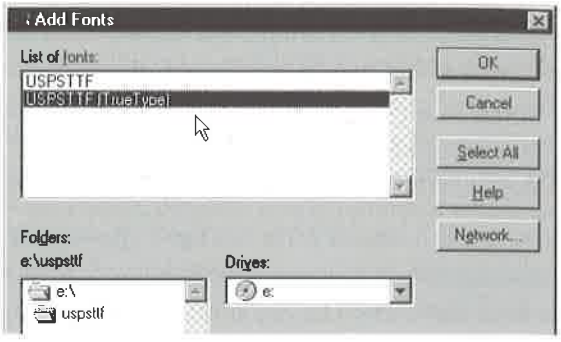
United States Postal Service Barcodes

TAZ Information Services has generously donated their shareware TrueType font for printing PostNet barcodes. (TrueType© Font copyright © 1993 TAZ Information Services, Inc. All Rights Reserved.)

The shareware TrueType font for printing PostNet barcodes is contained in the file `Uspsttfs.ttf`. This font must be installed using the Windows 95 Control Panel. (Do not install the `Uspsttfs.fon` file.) To install the font:

1. Insert the CD-ROM that comes with this book into your CD-ROM drive.
2. Click on the Start button at the Windows 95 desktop and point to Settings.
3. Click on Control Panel.
4. Double-click on the Fonts folder.
5. Choose File ➤ Install New Font from the Font window's menu bar.
6. Make sure the Copy fonts to Font Folder check box is checked.
7. Choose the drive letter for your CD-ROM drive from Drives.
8. Double-click on the USPSTTF folder.

9. Click on the USPSTTF (TrueType) fonts, as shown below:



10. Choose OK and follow the instructions on the screen.

Once it is installed, you can use the font as you would any other TrueType font. For example, when creating a report in Microsoft Access's design view, you can choose any control on the report. Then use the Formatting (Form/Report Design) toolbar to choose the USPSTTF font for that control.

From TAZ Information Services

The remainder of this section was supplied by TAZ Information Services. If you have any questions about their products, please contact them directly at

TAZ Information Services, Inc.
PO Box 452
Linthicum, MD 21090-0452
Telephone: (800) 279-7579
America Online: `TAZ INFO`

Specifications

To meet the U.S. Postal Service (USPS) specifications, the barcodes must be set in 16 point and the FIM pattern must be set in 72 point.

The frame bar that begins and terminates the POSTNET is generated by the keyboard character "s" or "S." The bar coding of the Zip Code itself follows USPS formats for 5-, 9-, or 11-digit Zip Code information by simply typing the numbers. The FIM patterns are obtained by typing the character of the desired pattern (either "a" "A," "b" "B," or "c" "C" for FIM A, FIM B, or FIM C, respectively).

Following are the three formats defined and an example of the keyboard input necessary to generate the proper POSTNET. In these examples, the address P.O. Box 452 and Zip Code 21090-0452 are used.

Five-Digit Zip Code (A Field)

Frame Bar	5-digit ZipCode	Correction Character	Frame Bar
S	**21090**	**8**	**S**

Keyboard = s210908s

Zip + 4 Code (C Field)

Frame Bar	5-digit ZipCode	+4 Code	Correction Character	Frame Bar
S	**21090**	**0452**	**7**	**S**

Keyboard = s2109004527s

Delivery Point Barcode (DPBC) (C Prime Field)

Frame Bar	5-digit ZipCode	+4 Code	Delivery Point	Correction Character	Frame Bar
S	**21090**	**0452**	**52**	**0**	**S**

Keyboard = s210900452520s

The generation of the Correction Character is accomplished by adding the digits in the Zip Code to be used (either 5, 9, or 11 digits). The correction character is the number that must be added to this sum to produce a total that is a multiple of 10. In other words,

Correction Character = 10 - (X(mod 10)),

where X is the sum of the digits.

For example 2 + 1 + 0 + 9 + 0 + 0 + 4 + 5 + 2 = 23. The next highest multiple of 10 is 30. Thus, the Correction Character is 7, 30 - 23 = 7.

The generation of a Delivery Point is, generally, to append the last two digits of the address line (street address, P.O. Box, rural route, etc.) to the zip +4 zip code number. The Correction Character is then generated on this number, i.e., sum of digits (21090045252) = 30. Thus the Correction Character is zero (0).

However, because of the wide variation of address line numbers, such as fractional streets, letter suffix, etc., no complete rule can be given here. A publication titled *Letter Mail Barcode Update* and dated May 1992 is available through your postmaster and contains all the rules for the generation of the delivery point.

CASS Certification

Preprinting the POSTNET barcode on your mail can improve delivery through the USPS. By obtaining a CASS Certification of your mailing list, you can substantially reduce your postage costs. TAZ can perform the CASS Certification of your mailing list required for reduced postage rates via disk or online file transfer. See the file `Cass.txt` in the USPSTTF folder for further information.

Disclaimer

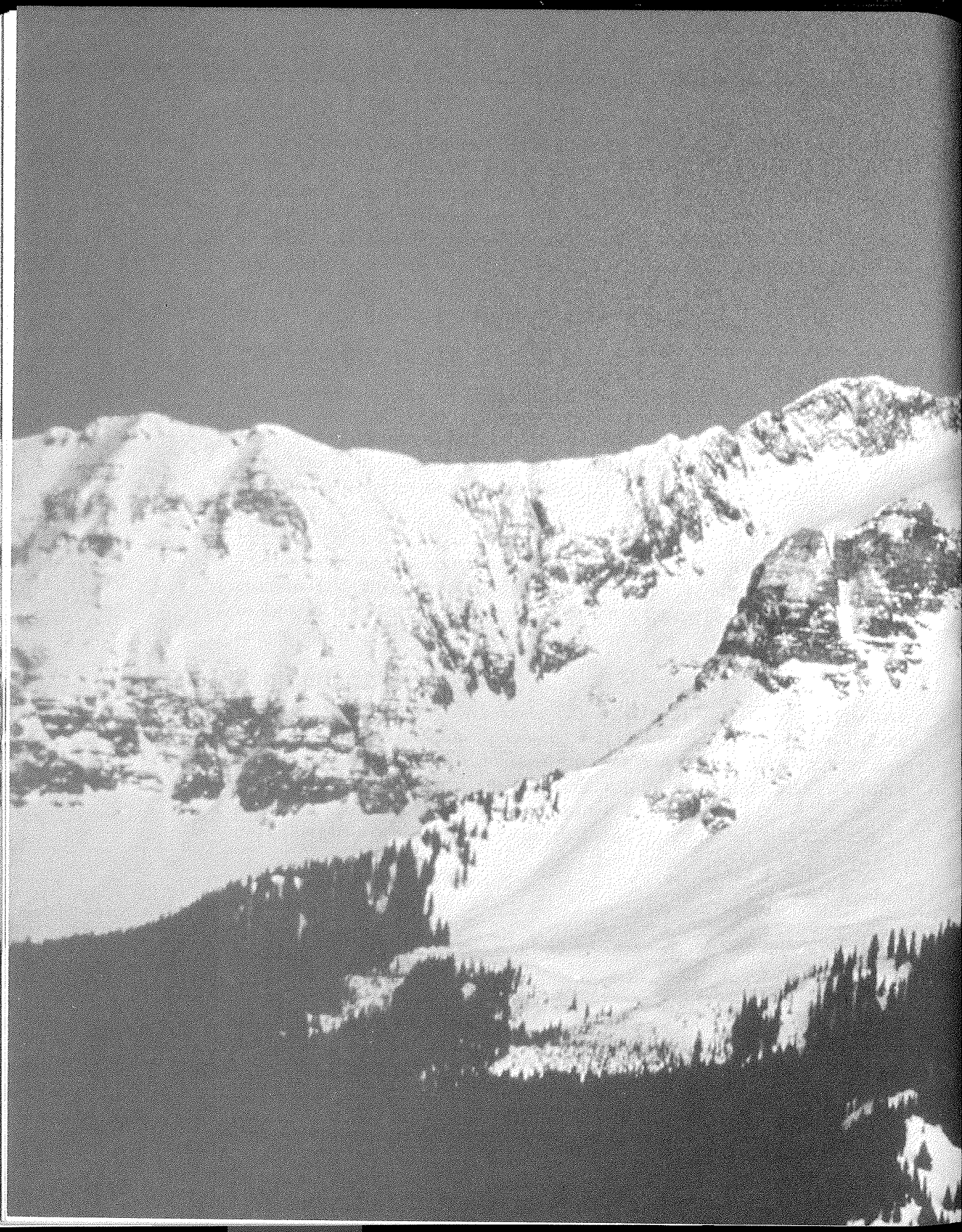
TAZ Information Services, Inc. makes no warranties on this trial copy of the USPS Barcodes and FIM Patterns TrueType font. In no event shall TAZ Information Services, Inc. be liable for any damages whatsoever arising out of the use of or inability to use this sample product.

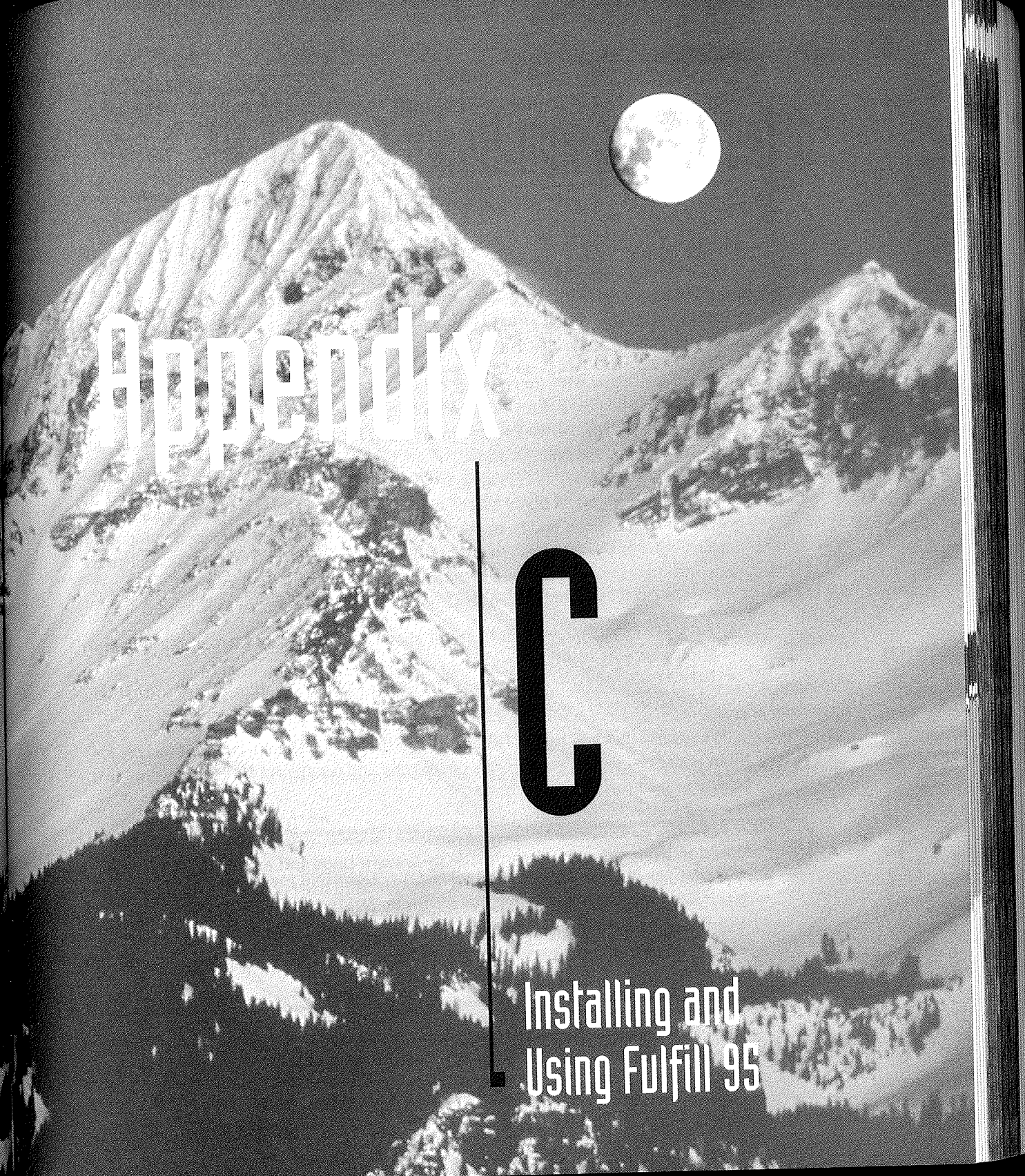
Registering Your Fonts

Thank you for trying the USPS Barcodes and FIM Patterns TrueType Font. If you have any questions about these fonts, please contact TAZ Information Services, Inc. TAZ will attempt to answer all questions; however, support is guaranteed only for registered users.

To register your fonts, please print and complete the form titled `REGISTER.TXT` in the `USPSTTF` folder on the CD-ROM that came with this book.

To check with Better Business Bureau of Greater Maryland, call (900) 225-5222.





Appendix C

Installing and Using Fulfill 95

Installing and Using Fulfill 95

Fulfill 95 is a sample order-entry database application that you can use and explore freely. It's especially designed for "SHMOOP" users (where SHMOOP stands for small office, home office, mobile office, and home PC), who are, as you may know, the same users that Windows 95 itself is geared to. You can use Fulfill to manage any kind of business (e.g., a mail-order business) that takes orders over the phone.

Fulfill 95 is a completely custom Access application. We didn't use Database Wizards to create it. Fulfill offers many more bells and whistles than the Wizard-generated order-entry application and is, in our opinion, much easier to use. Even if you don't need an order-entry database, you might still want to try Fulfill. You'll probably discover some neat tricks that you can incorporate into your own custom Access databases.

The CD-ROM that comes with this book contains two versions of Fulfill 95:

- **`FulfillS.mdb`** This version of Fulfill already contains some hypothetical data to show you how Fulfill works.
- **`Fulfill.mdb`** This version comes with most of Fulfill's tables empty so that you can add your own data from scratch.

We suggest that you try the FulfillS version first. If you then decide you can use Fulfill in your "real work," you're welcome to copy and use the `Fulfill.mdb` version. Just be sure to read this disclaimer first.

WARNING

Although every effort has been made to prevent bugs and errors, Fulfill 95 has never actually been tested in the field. If you plan to use Fulfill 95 to manage "real data," we strongly suggest that you run it in parallel with your existing manual or automated order-entry system for a while to see if it performs as you expect. Address any problems or errors to the author, whose address appears in Fulfill's Help ➤ About ➤ Author screen.

The focus in this appendix is on how to *use* Fulfill 95. If tinkering with Fulfill makes you curious about what makes it tick, you're more than welcome to peek under the

hood. Chapter 28 discusses techniques for exploring Fulfill 95 and similar open applications behind the scenes.

Copying Fulfill 95 to Your Hard Disk

Fulfill 95 is an Access database and therefore will work only on a computer that has Microsoft Access for Windows 95 installed. All you need to do is copy `FulfillS.mdb` (or Fulfill.mdb) from the CD-ROM onto your hard disk. You can use any copying technique you want. Then make sure its read-only attribute is not checked. You can copy Fulfill to any folder you want. Here are some step-by-step instructions to copy `FulfillS.mdb` to the My Documents folder on drive C:

1. Close any open applications to get to the Windows 95 desktop.
2. Insert the CD-ROM that came with this book into your CD-ROM drive.
3. Double-click on the My Computer icon at the Windows 95 desktop.
4. Double-click on the drive C: icon and then move and size the window so you can see the icon for the destination folder (the folder you'll be copying *to*), My Documents in this example.
5. Go back to My Computer and double-click on the icon for your CD-ROM drive (typically drive D:).
6. Size and position the window so you can see both the destination folder and the `FulfillS.mdb` file in Figure C.1.
7. Drag the `FulfillS.mdb` file (or `Fulfill.mdb` file) from the CD-ROM drive to the My Documents folder and then release the mouse button.
8. Double-click on the My Documents folder and right-click on the icon for FullfillS.mdb (or Fullfill.mdb). Select Properties and make sure the read-only setting is unchecked. Click on OK.

You can close the windows for MyDocuments drive C: and your CD-ROM drive.

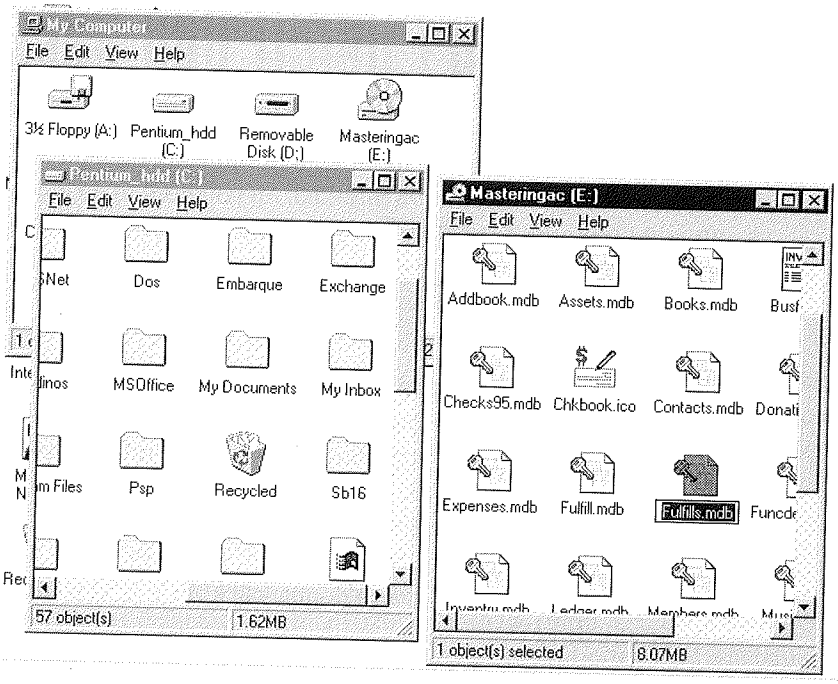
Starting Fulfill 95

Once you've copied `FulfillS.mdb` (or `Fulfill.mdb`) to your hard disk, you can use either of these techniques to start Access and load the Fulfill 95 database:

- Open the folder in which you've stored the database and double-click on the `FulfillS.mdb` file icon, as below.

FIGURE C.1

Ready to drag `FulfillS.mdb` from the CD-ROM to the My Documents folder on drive C:.



Or

- Start Microsoft Access and use the File ➤ Open Database commands to browse to and open `FulfillS.mdb` (or `Fulfill.mdb`).

Microsoft Access 97 will start up with the main switchboard for Fulfill 95 displayed, as in Figure C.2.

NOTE

When Fulfill 95 starts, you will have access to the standard Access menus and toolbars. Normally, we would have created custom menus and toolbars for an application such as this. But we want you to be able to explore "behind the scenes" easily, so we left all the standard Access tools in place.

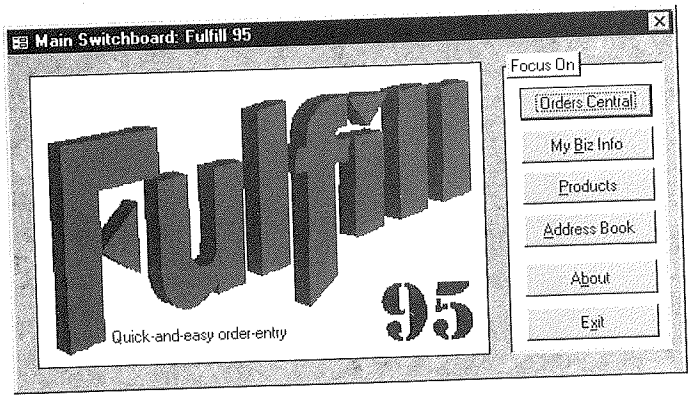


FIGURE C.2
The main switchboard for Fulfill 95.

Entering Information about Your Business

The first time you use Fulfill 95, you need to tell it about your business. You need to provide this information only once, not every time you start Fulfill. To enter information about your business:

1. Click on the My Biz Info button in the main switchboard.
2. Fill in the blanks as instructed on the screen and then click on the Click Here When Done button. You're returned to Fulfill's main switchboard.

If you're using FulfillS 95, you'll see some data already filled in, as in Figure C.3. You can leave the information as it appears or change it if you prefer.

Using Fulfill's Address Book

An important part of any order-entry system is the ability to manage names and addresses of customers, prospects, and suppliers. Fulfill 95 comes with a handy address book that makes list management very easy. To get to Fulfill's Address Book (see Figure C.4), just click on the Address Book button on Fulfill's main switchboard. If you're using FulfillS, you'll see the address for a hypothetical company on the screen.

FIGURE C.3

The form for filling in information about your own business in Fulfill 95.

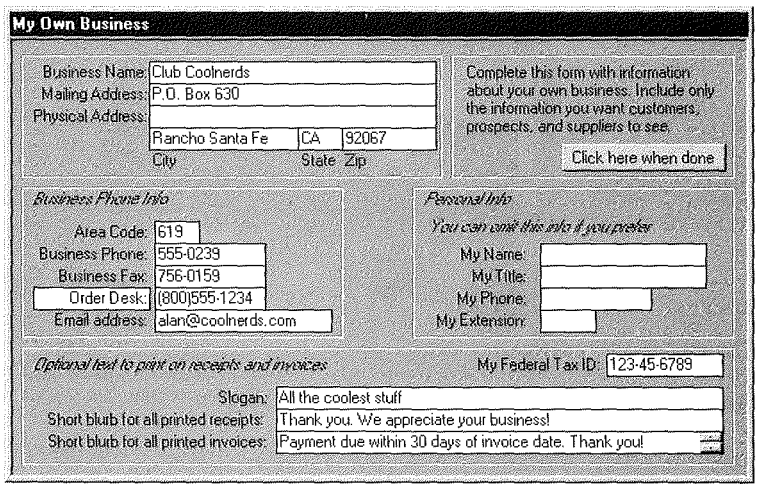
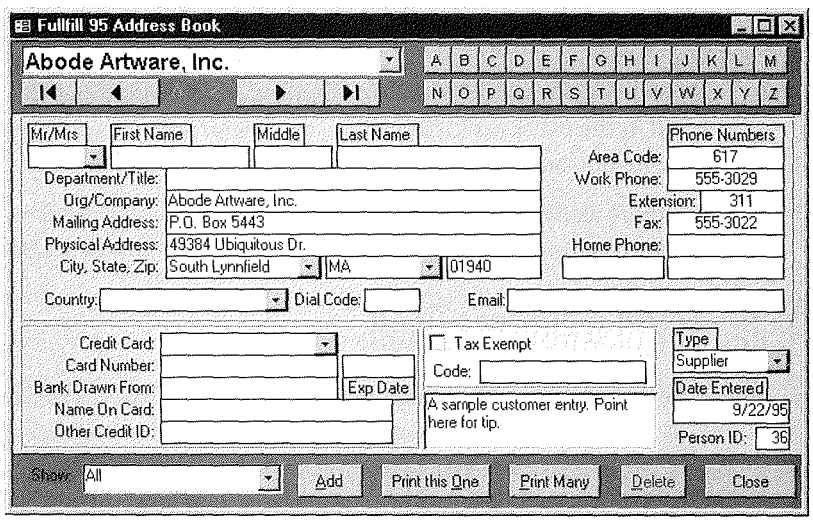


FIGURE C.4

Fulfill's Address Book form with a sample company address displayed.



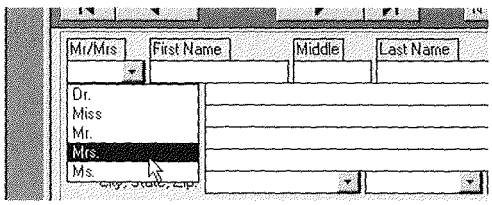
Adding a Name and Address

Adding a new name and address to Fulfill's address book is simply a matter of clicking on a button and filling in the blanks. (But if you don't read the sections that follow, you might miss lots of little time-savers and shortcuts.) To get started, click on the Add button near the bottom of the Address Book form.

Entering the Name and Company

The first few blanks in the Address book are for typing a person's name and affiliation. If you don't have the name of a particular person for this address (just a business name), you can leave these fields blank and skip right down to Department/Title or Org/Company. If you do have a person's name:

- Type an honorific (Ms., Mr. etc.) or choose one from the drop-down list under Mr/Mrs as shown below. You can leave this field blank if you want.



If you want to open a drop-down list without taking your hands off the keyboard, just press Alt+↓. Then you can use the ↑and ↓ keys to point to a selection and press Enter to select it. Also, if you want to change the Mr/Mrs drop-down list, double-click on the Mr/Mrs blank and follow the instructions on the screen.

- Press Tab to move to the next field or just click on the next field you want to fill.
- Type a middle name or initial. If you enter a single character, such as *C*, Fulfill will automatically add the period for you (*C.*).
- Type a Last Name.

If you're not sure how to fill in a blank on a form, try right-clicking and choosing Info from the shortcut menu that appears.

- Type any information for department, title, organization, or company name affiliated with the address you're entering into the Department/Title and/or Org/Company blanks. You may also leave either or both fields empty.

TIP

Tips for aspiring developers: The custom shortcut menu is assigned to the Address Book form's Shortcut Menu Bar property. Macros named AddressBookShortcut and AddressBookShortcut_Shortcut display the menu. The Info help is handled by a module named ShortcutMenuInfo.

Entering the Street Address

Now you need to enter the mailing address:

- If the blinking cursor isn't in the Mailing Address field yet, click on the field or press Tab until the cursor gets there.
- Type in the mailing address (required) and then press Tab or Enter. The cursor jumps to the Zip Code field, for reasons we'll describe in a moment.
- If you need to type in a second address line, do so in the Physical Address line. You can just backtrack to that field at any time by clicking on it or by pressing Shift+Tab.

NOTE

Pressing Tab or Enter *always* moves the blinking cursor forward to the next field in the form. Pressing Shift+Tab always moves the cursor back to the previous field. You can move the cursor to any field on the form simply by clicking on that field.

Entering the City, State, Zip, and Area Code

After typing in the mailing address and pressing Tab or Enter, the cursor lands in the Zip Code field. Fulfill will try to fill in the City, State, and Area Code fields automatically after you enter a zip code. Note that when entering a U.S. zip+4 code, you can omit the hyphen and Fulfill will add it for you. For example, if you type 920671234 as a zip code, Fulfill will automatically change that to 92067-1234. Type a zip code and press Tab or Enter to see how Fulfill works:

- If you *don't* hear a beep, Fulfill has made a "best guess" on the City, State, and Area Code and filled in those fields. *But you still need to check, and perhaps correct, Fulfill's guess because the Fulfill doesn't always get it right!* The cursor lands on the Work Phone field.
- If you *do* hear a beep, Fulfill doesn't know that zip code. You'll have to fill in the City, State, and Area Code fields yourself.

Fulfill will not change the City, State, or Area Code fields if you've already filled them in. Fulfill assumes that if you've already filled in the field with some information, it shouldn't try to second-guess you with its own information!

Fulfill is pretty accurate when filling in the City, State, and Area Code fields for a particular zip code. But it's not 100 percent accurate because some zip codes cover multiple townships and area codes. (Not to mention the fact that zip and area codes get reconfigured all the time!)

TIP

Tips for aspiring application developers: The zip codes are stored in a table named ZipLookup. The event procedure that looks up the zip code and fills in the City, State, and Area Code fields is attached to the After Update property of the Zip/PostalCode field in the AddressBook form.

Making Fulfill 100 percent accurate on filling in the city, state, and area code from the zip code would require storing a huge collection of data that would eat up disk space and slow down your work. As it stands, Fulfill contains a list of about 44,000 zip, city, state, and area code combinations that make it accurate enough to be worth including.

If Fulfill doesn't find a zip code you type in often, you can add that zip code to the list. Just double-click on the field where you type in the zip code and then follow the instructions on the screen.

Optionally, you can add the city, state, zip, and area code combination to the City drop-down list, as we'll discuss next.

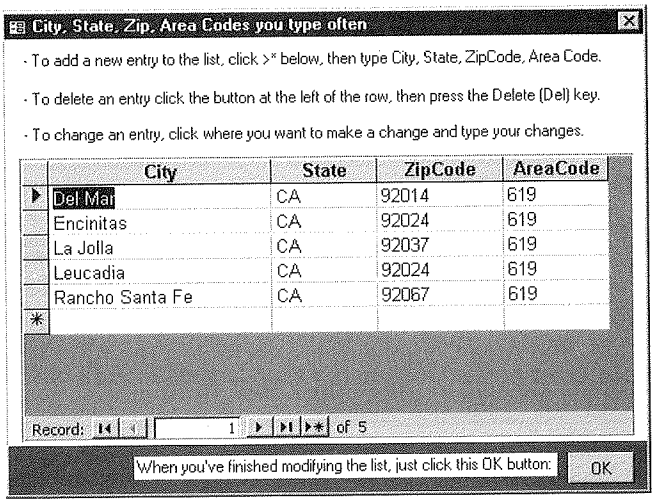
Using the City Drop-Down List - If most of the addresses that you type in are within your own vicinity, you can pre-enter the city, state, zip, and area code so that you don't have to type in *any* of that information in the future. To do so, double-click on the City field, and you'll see the dialog box shown in Figure C.5. (If you're using FulfillS you'll see some examples already typed in.)

Follow the on-screen instructions to add, change, or delete any city, state, area code, and zip code combination(s) that you would otherwise need to type in often. Click on OK to return to the Address Book form.

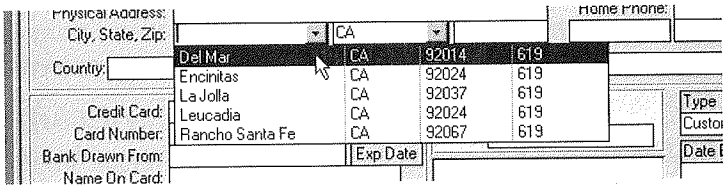
In the future you'll be able to choose any city, state, zip, area code from your custom drop-down list. After typing in the mailing address for an address in your own vicinity, click on the drop-down list button in the City field. Your custom list will appear, as in

FIGURE C.5

Double-clicking on the City field takes you to a dialog box for modifying the City drop-down list.



the example below. Just click on the one you want, and Fulfill will fill in the City, State, Zip, and Area Code fields for you.



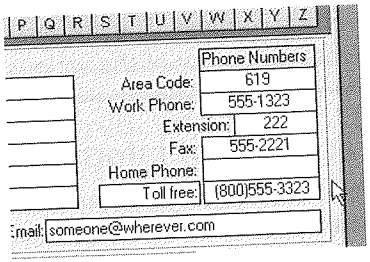
Note that Fulfill also has a drop-down list button for the State field, which you can use to fill in the two-letter abbreviation for any state in the United States.

Entering the Country

If you are entering an address that's within the United States, leave the Country and Dial Code fields empty. Blanks in those fields mean USA. If you are entering an address from a foreign country, you can type in the country name and dialing code or choose a country from the Country field's drop-down list. You can also customize the Country drop-down list to better suit your own needs, using the standard Fulfill technique: Just double-click on the Country field and follow the instructions that appear on the screen.

Entering the Area Code, Phone Numbers, and E-Mail Address

When filling in the phone numbers, enter the area code only once at the top of the list unless, however, one of the phone numbers uses a different area code, such as 800, in which case you *do* want to type in the area code, as in the Toll Free example below:



You can leave any field blank if you don't have the appropriate information. Also, note that you can add a label and phone number for the last number in the list. For example, we typed in a Toll Free: label next to the phone number in the example shown.

When typing the phone number, you can omit the hyphen. For example, if you type 5551323, Fulfill will convert that to 555-1323 after you move to another field. If you do need to type in an area code with a particular phone number, you can omit the parentheses. For example, if you type 8005553323, Fulfill will convert your entry to (800)555-3323.

There are no shortcuts for the E-mail field. If the name you're entering has an e-mail address, just go ahead and type it into the E-mail field.

Entering Credit Card Information

If you're entering the name and address for a customer who regularly pays by credit card, you can record that credit card information in the Address Book. Choose a credit card from the Credit Card drop-down list. If you accept a credit card that isn't in the list, you can add it to the list. Just double-click on the Credit Card field and follow the on-screen instructions.

WARNING

Fulfill 95 does not have any built-in security features to prevent unauthorized users from seeing credit card information. If security is a concern, you should either omit the credit card information or, at the least, omit the expiration date.

You can leave any of the Credit Card fields blank if you prefer. When typing in the expiration date, you should use the standard mm-yy format. But you can omit the slash

and leading zero. For example, if you type 1296 Fulfill will convert your entry to 12/96. If you type 196 Fulfill will convert that to 01/96 (after you move to another field).

More tips for application developers: Most of the underlying fields in the AddressBook table are of the text data type, which makes it easier to create these little custom conversion routines. The conversion routines themselves are attached to the After Update properties of the various Phone fields and ExpirationDate field.

Entering Tax Exempt Information

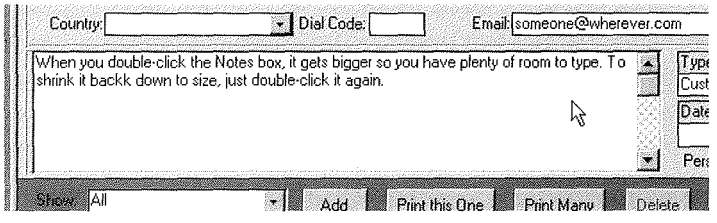
The Tax Exempt fields are for those rare customers who reside in a state where you normally charge sales tax, but who are, for whatever reason, exempt from sales tax. If you acquire such a customer, you can select (check) the Tax Exempt option. Optionally, you can record the account number or whatever information that customer offers to justify their tax-exempt status. When you enter orders for this customer later, Fulfill will not charge sales tax.

Fulfill charges sales tax only to customers in areas where you are required to charge sales tax. If a particular customer is in a region where you don't charge sales tax, you don't have to mark that person as tax exempt. Fulfill won't charge them sales tax anyway. You set up your sales tax rates in the Orders form, as we'll discuss later in this appendix. You don't need to concern yourself with that problem right now.

Entering Customer Notes

The empty white space under the Tax Exempt fields is for entering optional notes. The notes can be as long as you want them to be. Now you might be thinking, "Yeah, but the box is so small." Here's a little trick for you:

- Double-click on the Notes field to make it larger, as below:

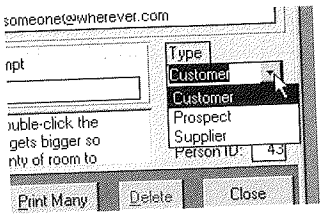


- Double-click on it again to shrink it back down to size.

Tip for aspiring application developers: The AddressBook form contains two controls bound to the Notes field: one named Notes and an invisible one named BigNotes. The event procedures that make BigNotes visible/invisible are attached to the On Dbl Click property of the Notes and BigNotes controls.

Address Type, Date Entered, and Person ID

The final step in entering a person's name and address is to identify the addressee as a customer, prospect, or supplier. You can use the drop-down list button under Type to do so:



You can't change the Type drop-down list, so double-clicking on it does nothing. Fulfill uses the information in the Type field when printing information about customers and *must* be able to identify the address as belonging to either a customer, prospect, or supplier.

The Date Entered and Person ID fields in Fulfill's Address Book are filled in automatically and cannot be changed. (In general, a yellow-tinted field on a form is for information only and cannot be changed.)

When you've finished typing in all the information for one name and address, click on the Add button to enter another address or click on Close to return to the Main Switchboard.

It's not necessary to enter every name and address through the Address Book form. As you'll discover, you can enter customer, prospect, and supplier addresses on the fly while entering orders and information about your products. But if you're not using FulfillS and you want to try out some of the navigation and printing techniques discussed in the sections that follow, you should add at least a few names and addresses so you have some data to work with.

Navigating with the Address Book

The tools across the top of the address book are for *navigation*—finding a particular address to change or print. The address book is automatically kept in alphabetical order, based on the following rules:

- If an entry appears in the Last Name field, then the address is listed in alphabetical order by name.
- If a particular address has no entry in its Last Name field, then the entry is listed in alphabetical order by Company name.

So for example, suppose you have the addresses in three records of the Address Book, as shown in Table C.1.

TABLE C.1: SAMPLE DATA IN FIRSTNAME, LASTNAME, AND COMPANY FIELDS

FIRST NAME	LAST NAME	COMPANY
		Sybex, Inc.
Sylvia	Levanthal	A & B Furniture
Andy	Adams	

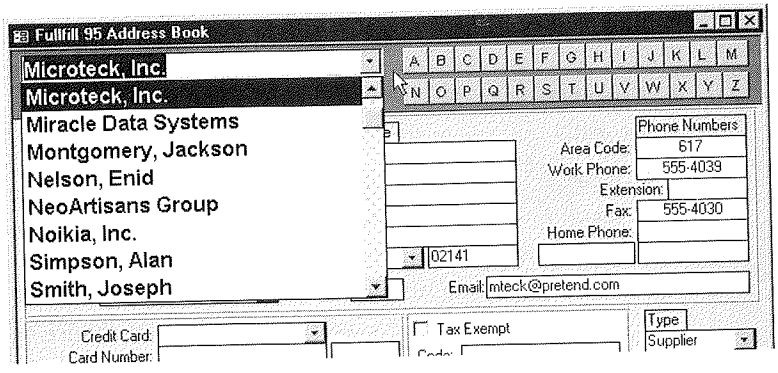
Because the last two rows have Last Name entries, they are listed in alphabetical order by name. Since the first entry has no Last Name contents, it will always be listed in alphabetical order by company. Thus, the three entries would be alphabetized as

Adams, Andy
Levanthal, Sylvia
Sybex, Inc.

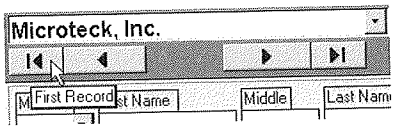
Tip for aspiring developers: The tricky alphabetization scheme is handled by a calculated field in the query named AddressesAlphabetized.

Navigating around in Fulfill's Address Book is pretty easy once you understand those rules. You just use the tools at the top of the form to zero in on the name you're looking for. Here are the exact steps:

- Click on one of the letters in the Rolodex button area to jump to that part of the alphabet. (If nothing in the alphabetical list starts with the letter you clicked, you'll just hear a beep and nothing will change.)
  The drop-down list box to the left of the Rolodex buttons shows the first address entry that starts with that letter.
- Use the drop-down list button to view nearby names, as shown below. To jump to one of those names, just click on it in the drop-down list.



You can also use the navigation buttons that surround the A...Z buttons to move from record to record. To see where one of those buttons will take you, rest the mouse pointer on the button for a couple of seconds and wait for the ScreenTip to appear:



Searching the Address Book

If you're sure you've entered a particular name and address, but can't seem to find it using the navigation buttons, perhaps you've just forgotten the exact spelling of the person's last name or the company name. You can search for any text in any part of an address.

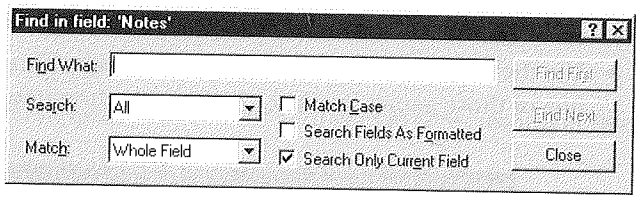
NOTE

If you use the Show option to limit the display of address to Customers, Prospects, or Suppliers, and you want to include *all* addresses in your search, set the Show option back to All before you begin the search.

1. Right-click on the specific field you want to search.

   Or

   Right-click on any field on the form if you want to search all the fields in all the records.
2. Choose Find from the shortcut menu to display the Find dialog box:



3. Type the text you're looking for in the Find What box.
4. Set other options in the Find dialog box as follows:

   - **Search** Choose Up to search up from your current position, Down to search down from your current position, or All to search the whole address book (broadest search, most likely to find a match).
   - **Match** Choose Whole Field, Start of Field, or Any Part of Field. The last option provides the broadest search and has the best chance of finding a match.
   - **Match Case** Select this option only if you want to locate an exact uppercase/lowercase match to the text you typed. Leaving this option cleared provides a better chance of finding a match.
   - **Search Fields As Formatted** Looks for a specific format match rather than just an informational match. Leave this option cleared to increase your chances of finding a match.
   - **Search Only Current Field** If you select (check) this box, only the field that you right-clicked on will be searched. The name of that field appears in the Find dialog box's title bar. To broaden the search and increase your chances of success, you can clear that checkbox.

5. Click on the Find First button to begin the search.

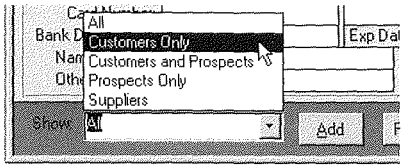
If Fulfill finds a match, the first matching record will be displayed. If you can't see the information behind the Find dialog box, just drag the dialog box out of the way

(by its title bar). If the found record is not the one you're looking for, you can click on the Find Next button to locate the next record that matches your request. When you find the record you're looking for (or are ready to give up), close the Find dialog box (click on its Close button).

If you can't find the record you're looking for, perhaps you misspelled something in the entry. You might want to print a directory of all the names and addresses you've entered to see how everything is spelled. We'll talk about printing in a moment.

Limiting the Address Book Display

If you have lots of addresses in your address book and want to limit the display to Customers, Prospects, or Suppliers, choose an option from the Show drop-down list near the bottom of the form (see below).



Any addresses that aren't of the type(s) you specified will be virtually invisible until you reset the Show option to All.

Printing from the Address Book

You can print information from the Address Book in a variety of formats at any time. If you want to print only the address currently showing on your screen, click on the Print This One button to open the Quick Print One Address dialog box shown in Figure C.6.

Click on the format you want to print. Then click on the Preview button if you want to see how the address will look when printed. To actually print the address, prepare the printer and click on the Print button. When you've finished printing, click on the Close button in the Quick Print One Address dialog box to return to the Address Book form.

NOTE

The various label and card formats available in the Print dialog boxes are standard Avery labels for laser and ink-jet printers. You can purchase those labels at most office-supply and computer stores. The mini-labels (Avery 2x6x) are for very small print runs (usually one or two labels) and work properly only with certain printers.

If you want to print several addresses, click on the Print Many button to open the standard Print Names and Addresses dialog box shown in Figure C.7.

FIGURE C.6

Options for printing one name and address from Fulfill's Address Book.

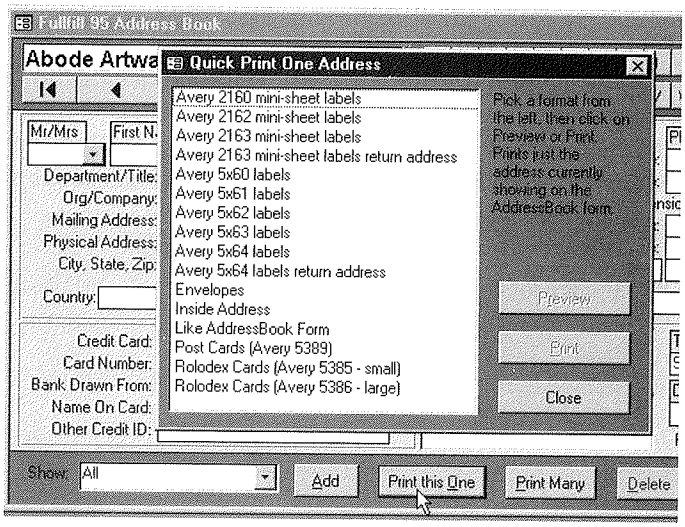
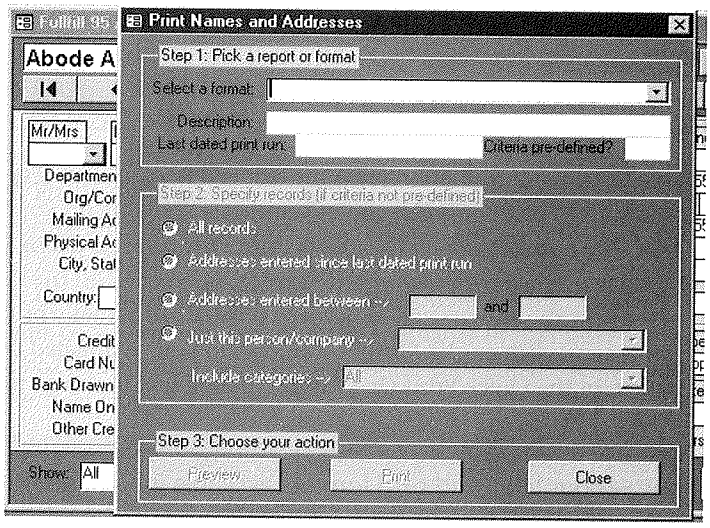


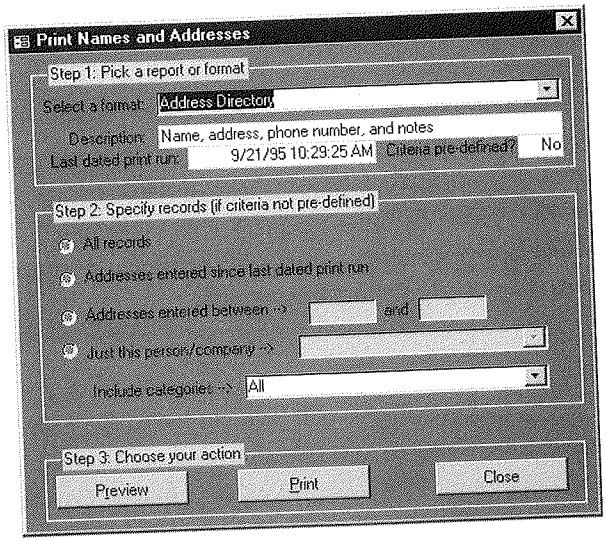
FIGURE C.7

Options for printing several names and addresses from Fulfill's Address Book.



To use the Print Names and Addresses dialog box, select a format from the drop-down list box. Depending on the format you select, the dialog box will display additional options. For example, suppose you choose the Address Directory format, as in Figure C.8.

FIGURE C.8
Address Directory format options for printing several names and addresses.



The yellow-tinted fields tell you some things about the format you've selected:

- **Description** Describes the format you've selected.
- **Last Dated Print Run** The date and time of the last time you printed in this format using the second option under Step 2 (described in a moment).
- **Criteria Pre-defined?** If it is set to Yes, you cannot choose Criteria under Step 2 because the criteria are already defined for this format. (The Categorized Address Summary and Possible Duplicates reports are the only two with predefined criteria.)

After selecting a format that allows you to specify your own criteria, you can choose options under Step 2 in the Print Names and Addresses dialog box to specify which records you want to include in the printout:

- **All Records** All names and addresses within the selected category will be printed.
- **Addresses Entered Since Last Dated Print Run** Only new entries—those addresses entered since the date and time shown in the yellow Last Dated Print Run box will be printed. For example, if you want to print Rolodex cards only for addresses that you've never printed before, you would choose this option.
- **Addresses Entered Between** If you choose this option, you can specify a range of dates to include. Initially the option will suggest "today" by setting both the start date and end date to today's date. But you can type in any range of dates you wish.
- **Just This Person/Company** You can pick a single person or company to include in the printout.

The Include Categories option lets you limit the printout even further. For example, if you choose Customers and Prospects from the Include Categories drop-down list, only addresses for those two categories will be printed. Suppliers will not be included in the print run.

Tips for aspiring developers: The Print dialog boxes are forms named AddressPrintOne and AddrRepPrintDialog in Fulfill. Visual Basic code attached to the Print and Preview buttons handles much of the work of setting up an appropriate query and previewing/printing the reports.

If you select All next to Include Categories, then all types of addresses—Customers, Prospects, and Suppliers—will be included in the print run.

Once you've made your selections, you can click on the Preview button to see how the printed report will look. Once you're in the Preview mode you can click on the document or use the various Zoom tools on the toolbar to change magnification. When you're done previewing, click on the Close button in the toolbar.

To print the addresses, click on the Print button. If you requested labels or cards, you'll hear a beep and see a message reminding you to load the proper stock into the printer. Load the paper and then choose OK to start printing.

Managing Your Product List

Before you start filling orders, you should type in information about some (or all) of the products you sell. You can add products on the fly while entering orders, but you need at least a few products in the list to get the benefits of Fulfill's order form.

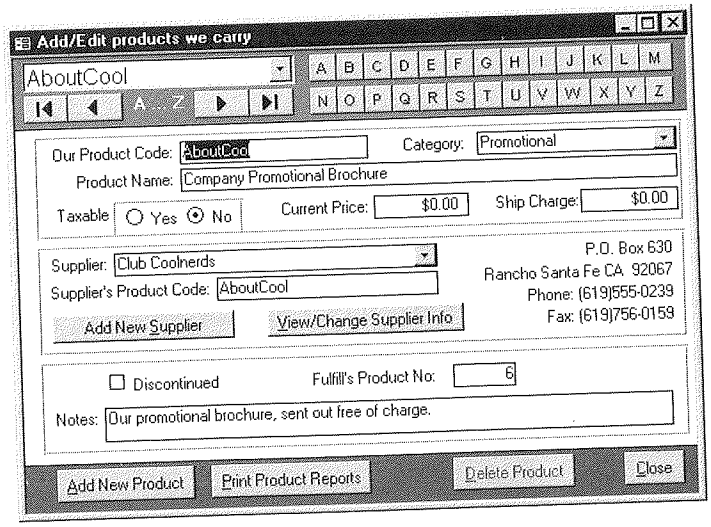
If you're using FulfillS, several products and product categories are already entered for you. But you may want to follow along in the sections that follow, just to get a feel for the Products component of Fulfill.

Adding New Products

When you want to add a product to the list of products that you sell, follow these steps:

1. Click on the Products button in Fulfill 95's Main Switchboard. You'll come to the form shown in Figure C.9.
2. Click on the Add New Product button near the bottom of the form.
3. Fill in the blanks for a single product, using techniques summarized in the sections that follow.

FIGURE C.9
The Add/Edit Products We Carry form for Fulfill 95.



Entering Product Information

You can assign whatever code or abbreviation you like to each product you carry. However, each product must have its own unique identifying code. (That is, no two products can have the same ID code.) Later, when you're entering orders, Fulfill will present products in alphabetical order by code. Therefore, you should try to come up with a code that will be easy to remember.

You also need to assign the product to a product category. Choose a category from the Category drop-down list, or if you need to create a new category, double-click on the Category field. You'll come to a screen like the one in Figure C.10. Follow the on-screen instructions to add, change, or delete a product category and then choose OK to return to the Products form.

Type in the rest of the information as follows:

Product Name Enter a name or description that specifically identifies this product.

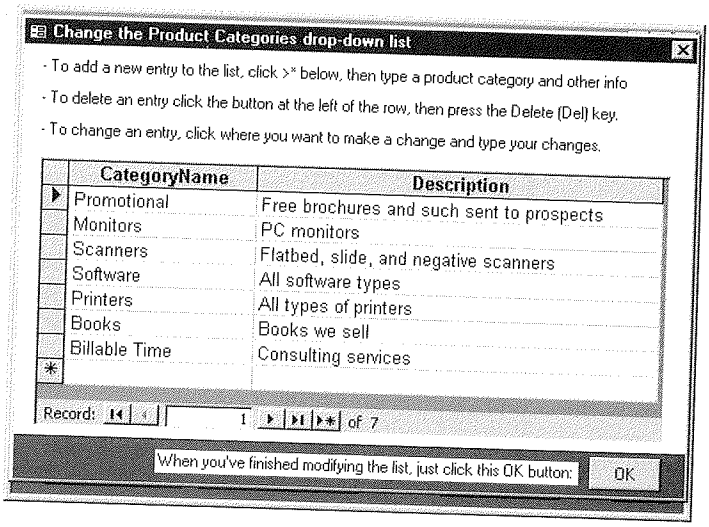
Taxable Choose Yes if this product is taxable in the region(s) in which you charge sales tax. If the product is not taxable, choose No.

Current Price Type the current selling price of the product.

Ship Charge If this product carries its own special shipping charge, type in that amount. Otherwise, leave it at $0.00. (Example: You might tack on an extra shipping charge to especially large or heavy products that are costly to ship.)

FIGURE C.10

Double-clicking on the Category drop-down list in the Products form takes you to the Change the Products Categories dialog box.



Entering Product Supplier and Other Info

If the product you're describing here is one you purchase wholesale from a supplier, then use the drop-down list button to choose that supplier. If you haven't already entered the supplier's name and address into Fulfill's Address Book, click on the Add New Supplier button and follow the instructions on the screen to fill in the new supplier's name and address.

If you are the supplier as well as the seller, enter your own name and address into Fulfill's Address Book as a Supplier. Then choose your own business as the Supplier when entering information about the product.

The yellow-tinted supplier information comes from Fulfill's Address Book and cannot be changed on this form. You may, however, click on the View/Change Supplier Info button if you need to make a quick change to the supplier's information while in the Products form. Use the Supplier's Product Code field to store the product code that the supplier uses to identify this product.

Finally, you can use the Notes field to type any other descriptive information about this product. The yellow-tinted Fulfill's Product Number code is assigned automatically.

You can select (check) the Discontinued box should you stop carrying this product at some time in the future.

NOTE

Once you start entering product information, you should complete the whole form. Otherwise, Fulfill might not save the entry when you close the form. Or you might get stuck, at which point you'll need to either complete the form or choose Edit ➤ Undo from the menu bar to cancel out the entry.

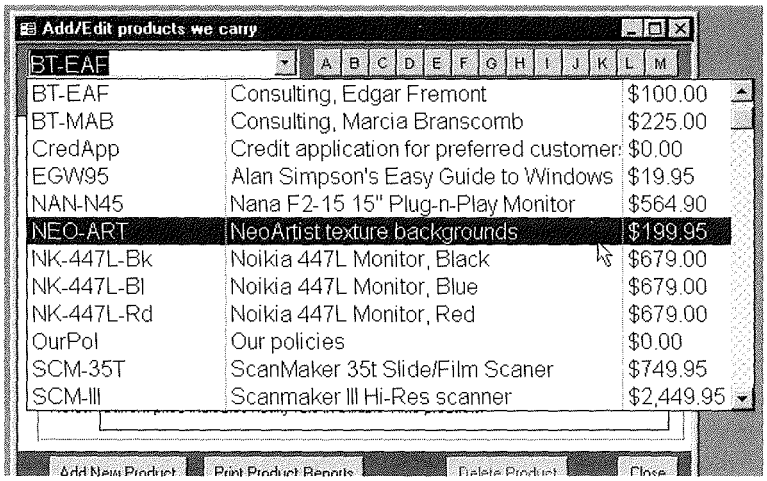
Navigating in the Products Form

You can also use the Products form to look up and change existing information. Your product list is kept in alphabetical order by your Product ID. So to quickly locate a product in your product list:

- Use the Rolodex-style buttons to click on a part of the alphabet (where you're looking to match the first letter of a product's ID code).

TIP

If on your first try, you don't land on the appropriate product, select a nearby product ID from the drop-down list, as shown below.



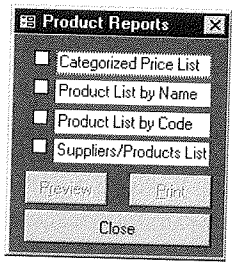
Or

- Use the VCR-style buttons to move from product to product.

Printing Product Information

To print your product list:

1. Click on the Print Product Reports button near the bottom of the Products form to display the Product Reports dialog box.



2. Select (check) the report(s) you want to print.
3. Click on the Preview button to preview the reports.
4. Click on the Print button to print the report(s).

Entering Orders

After you've entered some data into the My Biz Info, Address Book, and Products portions of Fulfill (or have copied FulfillS, which contains the sample data), most of your work with Fulfill will center around the Order Forms.

To add, view, change, or print orders, you start by clicking on the Orders Central button in Fulfill's Main Switchboard, which opens to the Orders Central switchboard shown in Figure C.11.

Entering a New Order

When you are ready to take an order, open a blank order form by clicking on the Enter A New Order button in the Orders Central switchboard. The first step in taking an order is to identify the customer (or prospect), as shown in Figure C.12.

- If a new customer or prospect (one whose name isn't already in Fulfill's Address book) is placing the order, click on the New Customer/Prospect button. Then see Adding a New Customer/Prospect On the Fly below.
- If an Existing Customer/Prospect (one whose name is already in Fulfill's address book) is placing the Order, click on the Existing Customer/Prospect button, and continue below.

FIGURE C.11

The Orders Central switchboard is where you'll do all your work with orders.

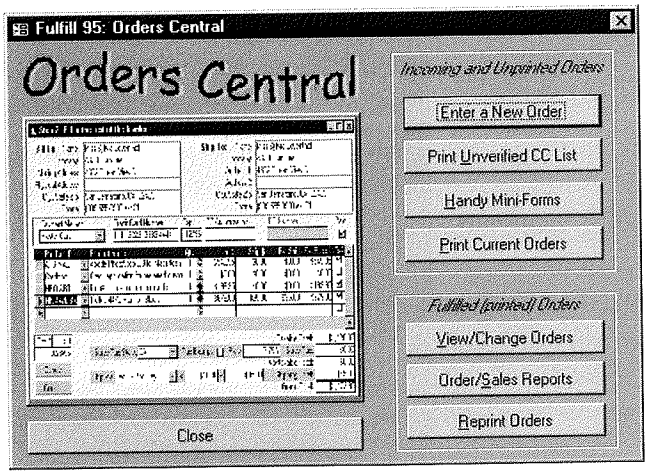
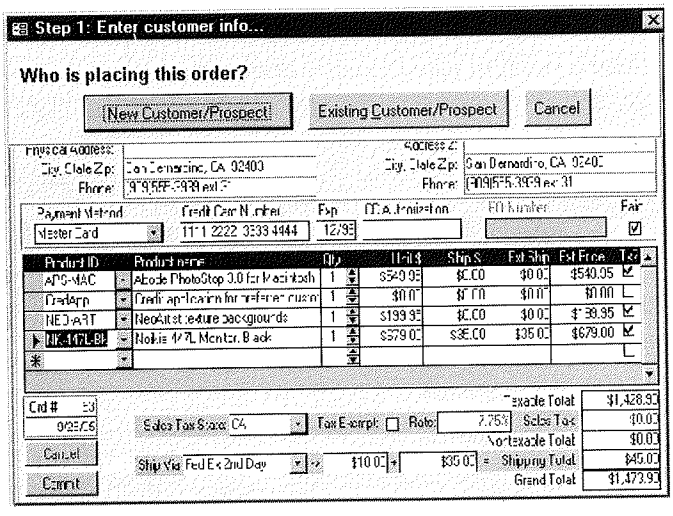


FIGURE C.12

Identify the customer/prospect.

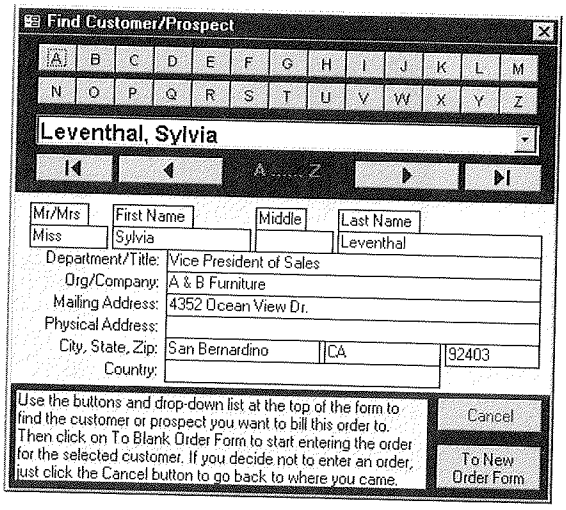


After you choose Existing Customer/Prospect, you're taken to the dialog box shown in Figure C.13. The navigation tools at the top of this dialog box work exactly as they do in the Address Book form discussed earlier in this appendix. All you need to do is locate the Customer/Prospect who is placing the order and then click on the To New Order Form button.

At this point, you can skip down to the section titled "Filling in the Rest of the Order Form."

FIGURE C.13

Use the navigation tools at the top of this form to locate the customer/prospect who is placing the order. Then click on the To New Order Form button.

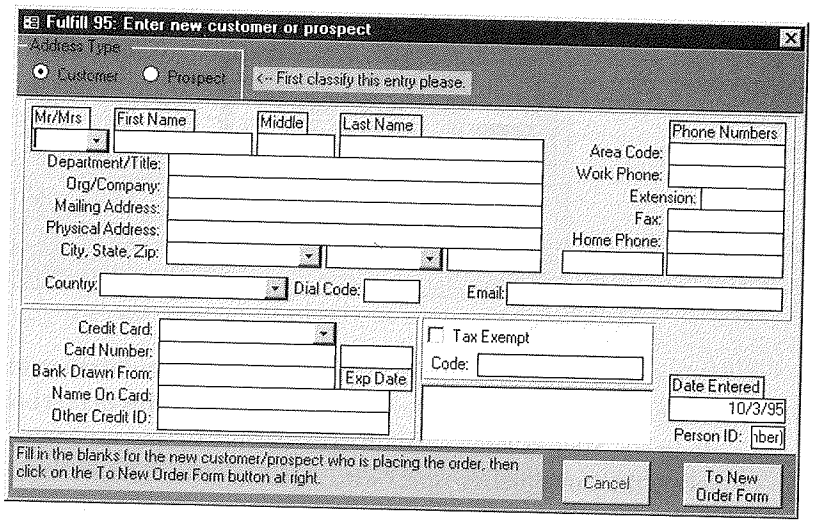


Adding a New Customer/Prospect On the Fly

If you choose New Customer/Prospect in response to the question, Who is placing this order?, you'll be asked to enter information about the new customer or prospect (see Figure C.14).

FIGURE C.14

Use this form to enter a new customer (or prospect's) name and address before moving on to the order form.



When entering the customer's (or prospect's) name and address, you can use some handy shortcuts and techniques that the Address Book offers to speed your data entry. (If you skipped all that, see "Adding a Name and Address" earlier in this appendix.) After you've filled in as many blanks as you can, click on the To New Order Form button to move to the order form.

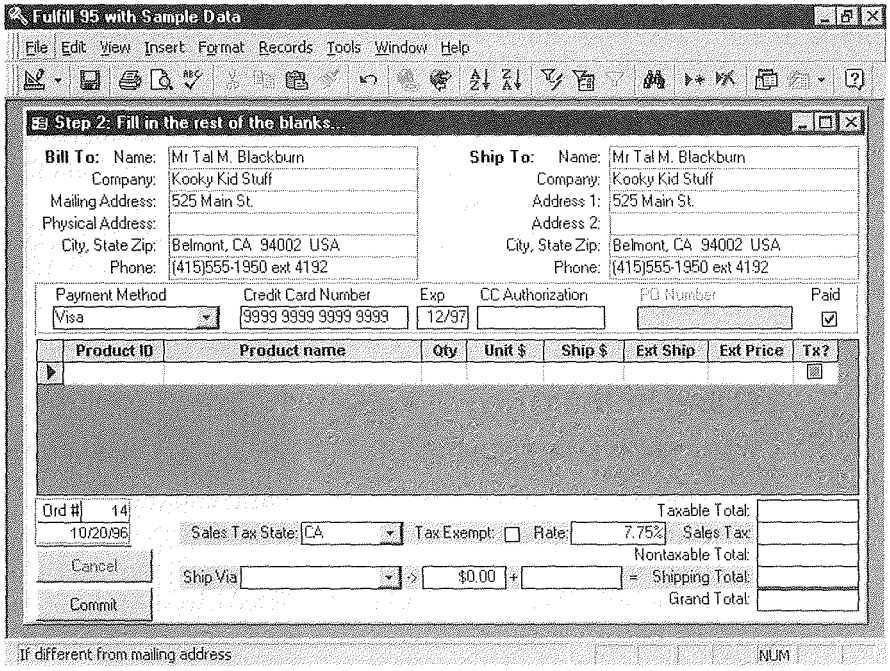
Filling In the Rest of the Order Form

The actual order form is shown in Figure C.15. Notice that much of the order form is already filled in for you:

- The Bill To name and address is already filled in with the customer's name and address. These yellow-tinted boxes cannot be changed here on the order form.
- The Ship To name and address are, initially, the same as those in Bill To. But you can change any of that information simply by clicking where you want to make a change and typing in the new information.
- If you've stored credit card information for this customer, the Payment Method is set to that credit card and the credit card number and expiration date are filled in automatically.

FIGURE C.15

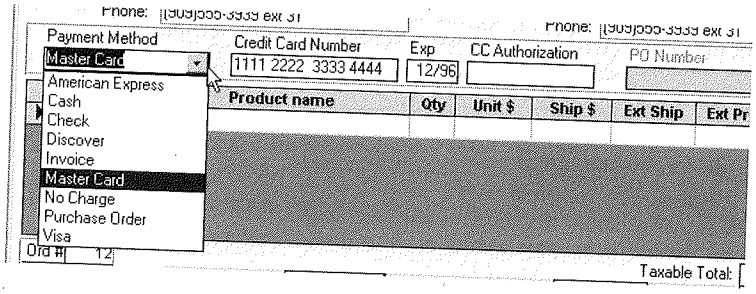
After you click on the To New Order Form button, you're taken to the actual order form with many of the blanks already filled in for you.



Filling In the Payment Information

When filling in the payment method section of the form, you can use these handy techniques and shortcuts:

- **Payment Method** Choose any payment method from the Payment Method drop-down list shown below. If you need to add a new payment method to the list, just double-click on the Payment Method field and follow the instructions on the screen.



- **Credit Card Number and Expiration Date** Fill in the Credit Card Number box only if the customer is paying by credit card. If you choose the credit card that you entered into the Address Book for this person, the card number and expiration date will be filled in automatically.
- **CC Authorization** If you have some means of verifying a credit card purchase while entering an order, you can fill in the authorization number in the CC Authorization box. Optionally, you can leave this field blank and fill it in at any time in the future, as we'll discuss a little later in this section.
- **PO Number** If you chose Purchase Order as the Payment Method for this order, you can type the customer's Purchase Order number into the PO Number field.
- **Paid** If you select (check) the Paid box, the order is marked Paid and Fulfill will print a receipt. If you leave the Paid checkbox empty, Fulfill will print an invoice.

Filling In the Order Details

When you get to the Order Details section (line items) of the form, here's how to proceed:

- **Product ID** Click on the Product ID field and then click on the drop-down list button that appears to see your list of products (as shown below). You can type the first letter of the Product ID you're looking for to quickly jump to that section of the list. When you see the Product ID you're looking for, click on it to select it. The Product Name appears, and the cursor jumps to the Qty (quantity) field.

NOTE

To add a new product to your product list on the fly, double-click on the Product ID or Product Name field. Then choose Add A New Product. Fill in the blanks and choose Close. When you get back to the order form, use the Product ID drop-down list button to select the new product.

- **Qty** The default quantity is one (1). If you need to enter a different value, just type it in. You can type in fractional numbers, such as 2.5 (for 2.5 hours of billable time, for instance).
- **Unit $** (Unit Price) and **Ship $** (per-unit shipping charge) fields are "suggestions" based on the contents of the Products field. But you can change either value for the current order. Fulfill automatically calculates the Extended shipping charge (Ext Ship) and Extended Price (Ext Price).

You can add as many line items as necessary.

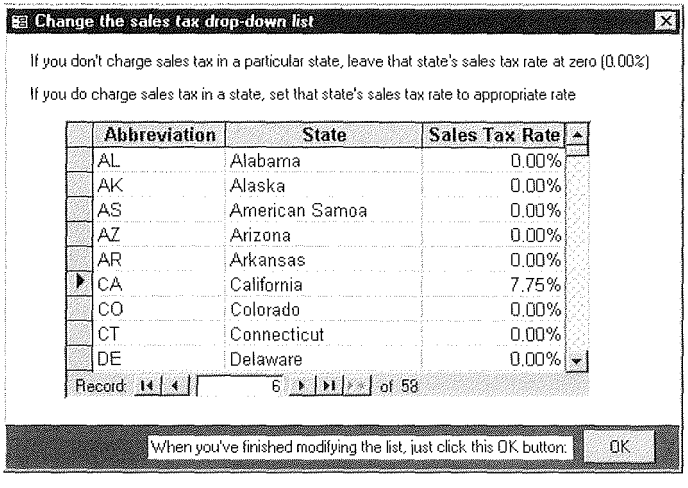
Sales Tax and Ship Via

The sales tax rate for an order is based on the values you put into the drop-down list. To change the sales tax you charge in any given state, double-click on the Sales Tax State field. You'll come to the Change the sales tax dialog box shown in Figure C.16.

You should also check the Sales Tax State entry to make sure that Fulfill has typed in the correct state, tax exempt status, and rate for you. If any one of those is incorrect, type in the corrected value. Fulfill automatically calculates the sales tax amount from your entry.

FIGURE C.16

Use this list to specify the sales tax rate you charge in states where you do charge sales tax. Leave the rate at 0.00% in states where you don't charge sales tax.



You can also choose a shipping method from the Ship Via drop-down list. If you need to add a shipping method or change its dollar amount, double-click on the Ship Via field and follow the instructions on the screen.

Note that the Shipping Total is based on the shipping charge of your Ship Via selection, plus the total per-unit ship charge from the Ext Ship column of the line items.

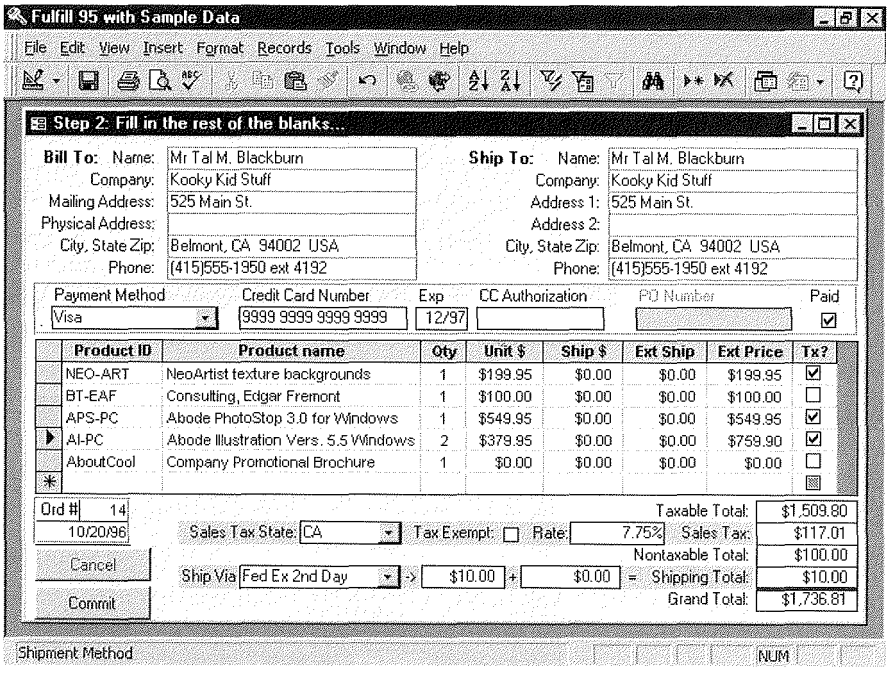
Cancel or Commit

Once you've completed one order, take a look at the form to check for any errors (it's always easier to correct errors *before* you save your work). Figure C.17 shows a sample completed order on the screen.

When you're finished, click on the Commit button to save the order. If you don't want to save this order, click on the Cancel button. You'll be returned to Orders Central. From there, you can either enter another new order (by clicking on Enter A New Order again) or return to the Main Switchboard (by clicking on the Close button) or start printing information about the current order(s), using the buttons beneath the Enter A New Order button.

FIGURE C.17

A new order entered using Fulfill 95.



Tools for Managing Current (Unprinted) Orders

Other buttons in Orders Central give you some quick tools for managing current orders (*current orders* being orders that have been typed into Fulfill, but not yet printed as invoices, receipts, and so forth). Those buttons are

- **Print Unverified CC List** Prints a list of credit card orders and expiration dates, to make it easier for you to call in for verification on those cards.
- **Handy Mini-Forms** Offers three small forms that make it easy to (1) fill in credit card verification numbers (after you've called in for authorization using whatever means you have available), (2) mark unpaid orders as Paid (after you've received payment), and (3) change any addresses currently marked as Prospect to Customer (after the prospect has actually made a purchase).
- **Print Current Orders** Prints packing slips, invoices, receipts, and labels for current orders.

You can print current orders at any time. For example, you might want to take in orders for the entire day, verify the credit card purchases, and fill in the verification numbers using the Handy Forms described above. Then at the end of the day, use the Print Current Orders button to print the day's invoices, receipts, packing slips, mailing labels, and shipping labels.

Once you have printed current orders, they are no longer considered "current." Rather, they're considered "printed" (or "fulfilled") orders. If you need to review, change, or reprint those printed orders, you can use the last three buttons in Orders Central, as discussed next.

Tools for Managing Printed Orders

The lower three buttons in Orders Central let you work with all orders in Fulfill. That is, you're not limited to current (unprinted) orders. We'll discuss these buttons in the sections that follow.

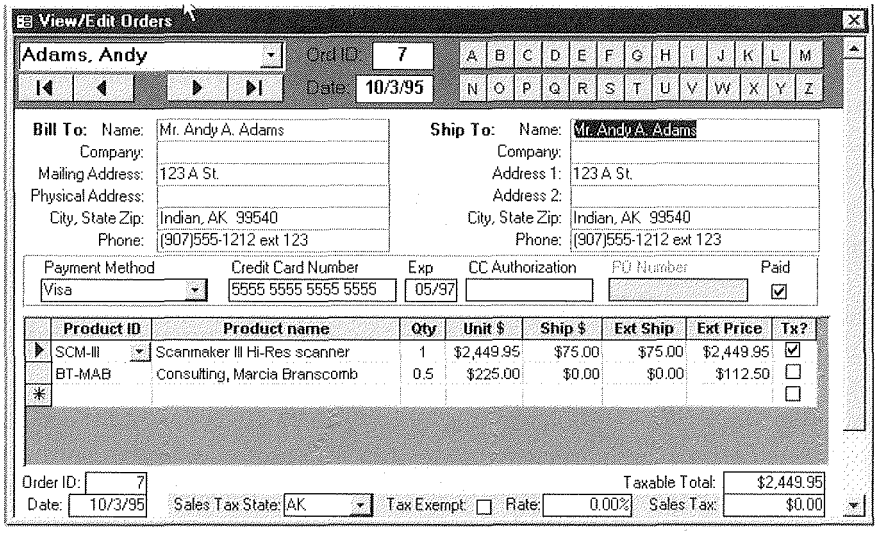
View/Change Orders

The View/Change Orders button takes you to an order form that's used strictly for viewing, changing, and printing orders. When you click on the View/Change Orders button, you come to an order form that has navigation tools. You can turn those navigation tools on or off simply by clicking on the Navigation Tools On/Off button in the toolbar. Figure C.18 shows the View/Edit Orders form with the navigation tools turned on.

To locate a specific order, you can use the same basic techniques you use to navigate the Address Book. That is, orders are organized in alphabetical order by Bill To name. To

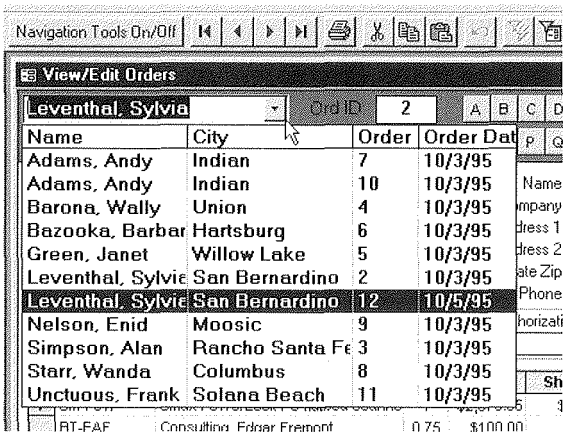
FIGURE C.18

The View/Edit Orders form with the navigation tools turned on.



look up a specific order, you first look up the name of the person or company that placed the order. The procedure goes like this:

- Click on one of the Rolodex-style buttons to jump to a section of the alphabet (where you're looking for the name of the person or company that placed the order).
- Click on the drop-down list button just to the left to see nearby orders. There may be more than one order for any customer, but the drop-down list also shows the Order ID and Order Date (see below). You should be able to zero in on a specific order easily.



- You can use the VCR-style navigation buttons to scroll through orders as well.

On a 640 × 480 monitor, the navigation buttons make it hard to see the bottom of the order form. So once you find the order you want to work with, click on the Navigation Tools On/Off button in the toolbar to hide those tools and see the entire order form.

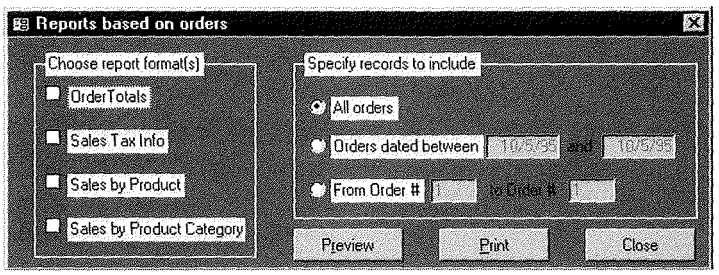
WARNING

Canceled orders do show in View/Change Orders with a check mark in the Canceled box. Canceled orders are never printed and are never used in calculating sales, sales tax, or any of the other reports available from the Order/Sales Reports button.

If you change an order and need to reprint it, you can just click on the Print This Order Now button in the toolbar. When you've finished viewing/changing orders, click on the Close button to return to Orders Central.

Orders/Sales Reports

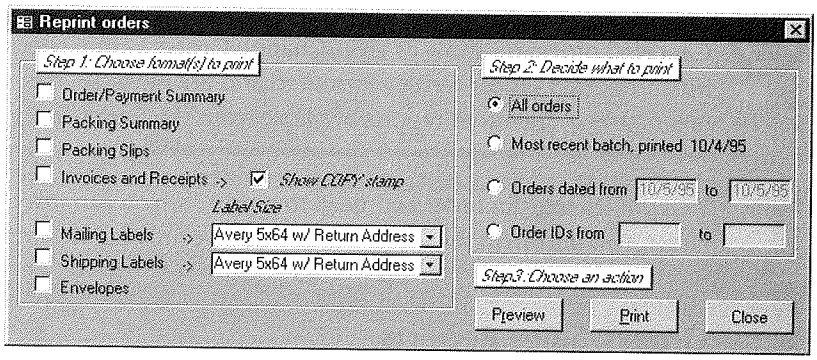
The Orders/Sales Reports button lets you print information about all orders or a range of orders you specify. When you click on the Orders/Sales Reports button, you're taken to the Orders/Sales Reports dialog box:



You can select any combination of reports simply by clicking on their checkboxes. You can also include All Orders, Orders within a certain range of dates, or a range of orders by Order ID. Then choose the Preview button to preview the reports on the screen or choose Print to actually print the reports. When you've finished with this dialog box, click on the Close button to return to Orders Central.

Reprint Orders

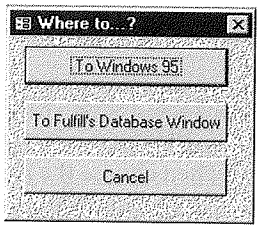
If you need to reprint any invoices, receipts, or labels for orders that you've previously printed using the Print Current Orders button, click on the Reprint Orders button. You'll be taken to the Reprint Orders dialog box:



As instructed on the screen, select a format (or formats) to print. Then decide what to print and click on the Preview or Print button. Click on the Close button to return to Orders Central.

Exiting Fulfill 95

To exit Fulfill 95, get back to the main switchboard and click on the Exit button. Choose any one of the following options in the Where to? dialog box:



- **To Windows 95** Closes Fulfill and Microsoft Access and takes you back to the Windows 95 desktop.
- **To Fulfill's Database Window** Keeps Fulfill and Access open and takes you to Fulfill's database window where you can explore behind the scenes (see Chapter 28).
- **Cancel** Neither of the above; takes you back to Fulfill's main switchboard.

Questions, Comments, Snide Remarks

As mentioned earlier in this appendix, we've made every effort to make Fulfill 95 bug free and accurate. But a freeware product such as this does not have the benefit of beta testing (i.e., testing in the field), so bugs and mistakes can slip by. The copy you have is named version 1.0. If you plan to use Fulfill for real work, you *are*, in a sense, the beta tester!

We will try to keep up with Fulfill and release additional versions with bug fixes and improvements, but we need your feedback to improve the product. If you find a problem or have a suggestion, feel free to contact the author, Alan Simpson, whose addresses are listed in the Help ➤ About ➤ Author box in Fulfill, as well as in the introduction to this book.

To check for any post-.0 versions of Fulfill or to see a list of FAQs (Frequently Asked Questions), check out Alan's Web site at `http://www.coolnerds.com/coolnerds`. TIA (thanks in advance) for taking the time to do that. :-)