Managing Projects

To create an application with Visual Basic, you work with projects. A *project* is the collection of files you use to build an application. This chapter describes how to build and manage projects.

When you create an application, you will usually create new forms; you might also reuse or modify forms that were created for previous projects. The same is true for other modules or files that you might include in your project. ActiveX controls and objects from other applications can also be shared between projects.

After all of the components in a project have been assembled and the code written, you compile your project to create an executable file.

Topics


Working with Projects

An introduction to projects, the Project Explorer, and project files.


The Structure of a Visual Basic Project

A discussion of the components that make up a project.


Creating, Opening, and Saving Projects

The basics of working with projects.


Adding, Removing, and Saving Files

The basics of working with the files within a project.


Adding Controls to a Project

The basics of working with ActiveX controls and insertable objects.


Making and Running an Executable File

The basics of compiling a project.


Setting Project Options

A discussion of some of the options that can be set on a project by project basis.


Using Wizards and Add-Ins

A discussion of additional tools for working with projects and extending Visual Basic.

Working with Projects

As you develop an application, you work with a project to manage all the different files that make up the application. A project consists of:

- One project file that keeps track of all the components (.vbp).
- One file for each form (.frm).
- One binary data file for each form containing data for properties of controls on the form (.frx). These files are not editable and are automatically generated for any .frm file that contains binary properties, such as Picture or Icon.
- Optionally, one file for each class module (.cls).
- Optionally, one file for each standard module (.bas).
- Optionally, one or more files containing ActiveX controls (.ocx).
- Optionally, a single resource file (.res).

The *project file* is simply a list of all the files and objects associated with the project, as well as information on the environment options you set. This information is updated every time you save the project. All of the files and objects can be shared by other projects as well.

When you have completed all the files for a project, you can convert the project into an executable file (.exe): From the File menu, choose the Make *project.exe* command.

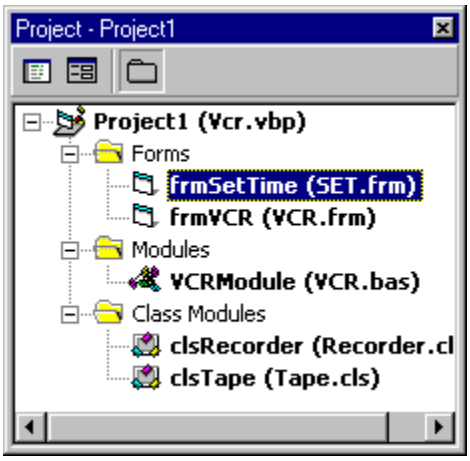
> **Note** With the Professional and Enterprise editions of Visual Basic, you can also create other types of executable files such as .ocx and .dll files. References in this chapter assume a standard .exe project; for additional information related to other project types see the *Component Tools Guide*.

For More Information For more details about creating executables, see "Making and Running an Executable File," later in this chapter. For information about binary data files and project files, see "Visual Basic Specifications, Limitations, and File Formats."

The Project Explorer

As you create, add, or remove editable files from a project, Visual Basic reflects your changes in the Project Explorer window, which contains a current list of the files in the project. The Project Explorer window in Figure 4.1 shows some of the types of files you can include in a Visual Basic project.

Figure 4.1 The Project Explorer window



The Project File

Each time you save a project, Visual Basic updates the project file (.vbp). A project file contains the same list of files that appears in the Project Explorer window, as well as references to the ActiveX controls and insertable objects that are used in the project.

You can open an existing project file by double-clicking its icon, by choosing the Open Project command from the File menu, or by dragging the file and dropping it on the Project Explorer window.

For More Information The specific format of information stored in the .vbp file is described in "Visual Basic Specifications, Limitations, and File Formats."

The Structure of a Visual Basic Project

The following sections describe the different types of files and objects that you can include in a project.

Form Modules

Form modules (.frm file name extension) can contain textual descriptions of the form and its controls, including their property settings. They can also contain form-level declarations of constants, variables, and external procedures; event procedures; and general procedures.

For More Information For more about creating forms, see "Developing an Application in Visual Basic" and "Creating a User Interface." For information about the format and content of form files, see "Visual Basic Specifications, Limitations, and File Formats."

Class Modules

Class modules (.cls file name extension) are similar to form modules, except that they have no visible user interface. You can use class modules to create your own objects, including code for methods and properties.

For More Information For information about writing code in class modules, see "Creating Your Own Classes" in "Programming with Objects."

Standard Modules

Standard modules (.bas file name extension) can contain public or module-level declarations of types, constants, variables, external procedures, and public procedures.

For More Information For information about using modules, see "Programming Fundamentals" and "Programming with Objects."

Resource Files

Resource files (.res file name extension) contain bitmaps, text strings, and other data that you can change without having to re-edit your code. For example, if you plan to localize your application in a foreign language, you can keep all of the user-interface text strings and bitmaps in a resource file, which you can then localize instead of the entire application. A project can contain no more than one resource file.

For More Information For more information on using resource files, see "Using a Resource File" later in this chapter, and "International Issues."

ActiveX Documents

ActiveX documents (.dob) are similar to forms, but are displayable in an Internet browser such as Internet Explorer. The Professional and Enterprise editions of Visual Basic are capable of creating ActiveX documents.

For More Information For more information on ActiveX documents, see "Creating ActiveX Components" in the *Component Tools Guide*.

User Control and Property Page Modules

User Control (.ctl) and Property Page (.pag) modules are also similar to forms, but are used to create ActiveX controls and their associated property pages for displaying design-time properties. The Professional and Enterprise editions of Visual Basic are capable of creating ActiveX controls.

For More Information For more information on ActiveX control creation, see "Creating an ActiveX Control" in "Creating ActiveX Components," in the *Component Tools Guide*.

Components

In addition to files and modules, several other types of components can be added to the project.

ActiveX Controls

ActiveX controls (.ocx file name extension) are optional controls which can be added to the toolbox and used on forms. When you install Visual Basic, the files containing the controls included with Visual Basic are copied to a common directory (the \Windows\System subdirectory under Windows 95 or later). Additional ActiveX controls are available from a wide variety of sources. You can also create your own controls using the Professional or Enterprise editions of Visual Basic.

For More Information For more information on using the included ActiveX controls, see "Using the ActiveX Controls" in the *Component Tools Guide*.

Insertable Objects

Insertable objects, such as a Microsoft Excel Worksheet object, are components you can use as building blocks to build integrated solutions. An *integrated solution* can contain data in different formats, such as spreadsheets, bitmaps, and text, which were all created by different applications.

For More Information For more information on using other applications' objects, see "Programming with Components."

References

You can also add references to external ActiveX components that may be used by your application. You assign references by using the References dialog, accessed from the References menu item on the Project menu.

For More Information For more information on references, see "Using Other Applications' Objects" later in this chapter.

ActiveX Designers

ActiveX designers are tools for designing classes from which objects can be created. The design interface for forms is the default designer. Additional designers may be available from other sources.

For More Information For more information about ActiveX designers, see "ActiveX Designers" in "Programming with Objects."

Standard Controls

Standard controls are supplied by Visual Basic. Standard controls, such as the command button or frame control, are always included in the toolbox, unlike ActiveX controls and insertable objects, which can be removed from or added to the toolbox.

For More Information For more information on standard controls, see "Forms, Controls, and Menus" and "Using Visual Basic's Standard Controls."

Creating, Opening, and Saving Projects

Four commands on the File menu allow you to create, open, and save projects.

Menu command	Description
New Project	Closes the current project, prompting you to save any files that have changed. You can select a type of project from the New Project dialog. Visual Basic then creates a new project with a single new file.
Open Project	Closes the current project, prompting you to save any changes. Visual Basic then opens an existing project, including the forms, modules, and ActiveX controls listed in its project (.vbp) file.
Save Project	Updates the project file of the current project and all of its form, standard, and class modules.
Save Project As	Updates the project file of the current project, saving the project file under a file name that you specify. Visual Basic also prompts you to save any forms or modules that have changed.

It is also possible to share files between projects. A single file, such as a form, can be part of more than one project. Note that changes made to a form or module in one project will be propagated amongst all projects that share that module.

For More Information For more information about sharing files, see "Adding, Removing, and Saving Files" later in this chapter.

Working with Multiple Projects

In the Professional and Enterprise editions of Visual Basic, it is possible to have more than one project open at a time. This is useful for building and testing solutions involving user-created controls or other components. When more than one project is loaded, the caption of the Project Explorer window will change to Project Group and the components of all open projects will be displayed.

To add an additional project to the current project group

1. From the **File** menu, choose **Add Project**.

   The **Add Project** dialog box is displayed.
2. Select an existing project or a new project type, and choose **Open**.

To remove a project from the current project group

1. Select a project or a component of a project in the **Project Explorer**.

2. From the **File** menu, choose **Remove Project**.

For More Information To learn more about working with multiple projects, see "Creating ActiveX Components" in the *Component Tools Guide*.

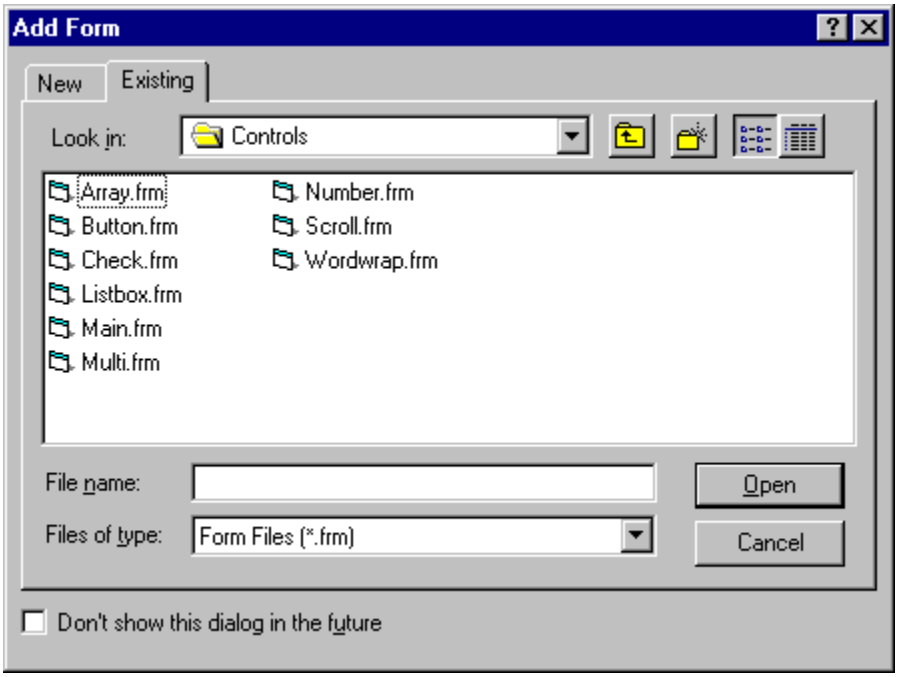
Adding, Removing, and Saving Files

Working with files within a project is similar to working with the projects themselves.

To add a file to a project

1. Select **Project**, **Add** *filetype* (where *filetype* is the type of file).
   The **Add** *filetype* dialog box (Figure 4.2) is displayed.
2. Select an existing file or a new file type, and choose **Open**.

Figure 4.2 The Add Form dialog box



When you add a file to a project, you are simply including a reference to the existing file in the project; you are not adding a copy of the file. Therefore, if you make changes to a file and save it, your changes will affect any project that includes the file. To change a file without affecting other projects, select the file in the Project Explorer, choose Save *filename* As from the File menu, and then save the file under a new file name.

Note You can drag and drop files from the Windows Explorer, File Manager, or Network Neighborhood into the Project window to add them to a project. You can also drag and drop .ocx files onto the toolbox to add new controls.

To remove a file from a project

1. Select the file in the **Project Explorer**.
2. From the **Project** menu, choose **Remove** *filename*.
3. The file will be removed from the project but not from the disk.

If you remove a file from a project, Visual Basic updates this information in the project file when you save it. If you delete a file outside of Visual Basic, however, Visual Basic cannot update the project file; therefore, when you open the project, Visual Basic displays an error message warning you that a file is missing.

To save an individual file without saving the project

1. Select the file in the **Project Explorer**.
2. From the **File** menu, choose **Save** *filename* .

Merging Text

You can also insert existing text from other files into one of your code modules. This is useful for adding a list of constants or for adding snippets of code that you might have saved in text files.

To insert a text file into your code

1. From the Project window, select the form or module into which you want to insert code.
2. Choose the **View Code** button, and move the cursor to the point in the Code Editor where you want to insert code.
3. From the **Edit** menu, choose **Insert File**.
4. Select the name of the text file you want to insert, and choose **Open**.

Note If you edit Visual Basic source files using a text or code editor other than Visual Basic, be careful not to change settings of the attribute VB_PredeclaredId. In particular, changing this attribute may cause serious problems with the GlobalMultiUse and GlobalSingleUse classes.

In general, you should not edit attributes manually, as doing so may put the module into an internally inconsistent state.

Adding Controls to a Project

The set of controls available in the toolbox can be customized for each project. Any given control must be in the toolbox before you can add it to a form in the project. The basic set of standard controls that always appear in the toolbox is described in "Forms, Controls, and Menus."

Adding ActiveX Controls to a Project

You can add ActiveX controls and insertable objects to your project by adding them to the toolbox.

To add a control to a project's toolbox

1. From the **Project** menu, choose **Components**.

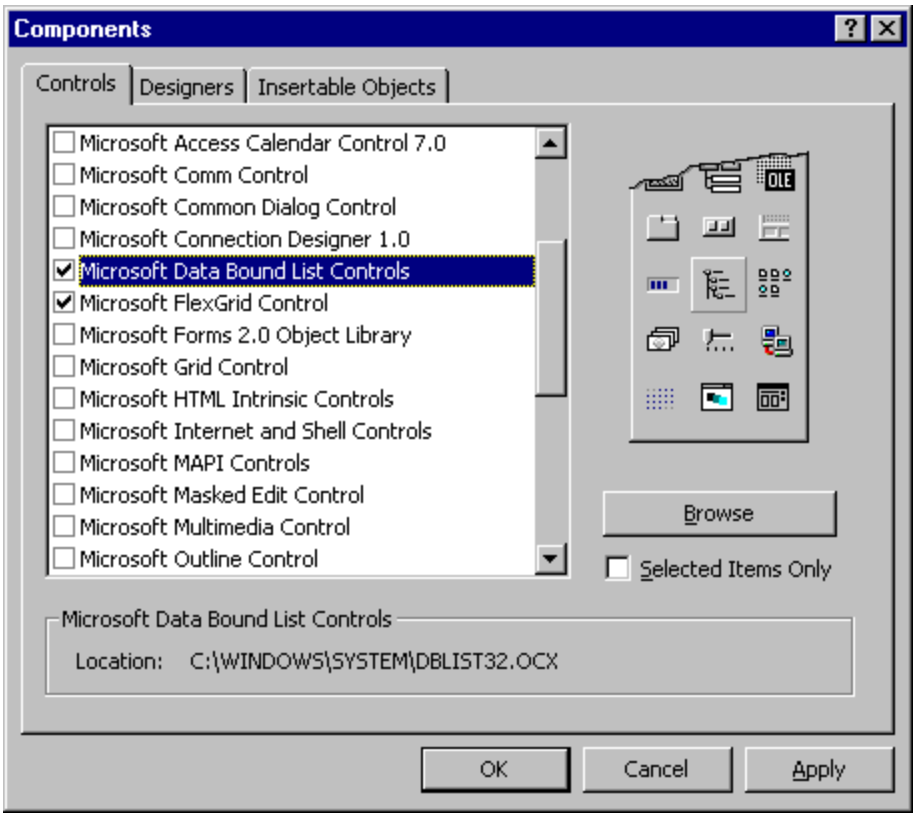
   The **Components** dialog box is displayed, as shown in Figure 4.3. The items listed in this dialog box include all registered ActiveX controls, insertable objects, and ActiveX designers.

2. To add a control (.ocx file name extension) or an insertable object to the toolbox, select the check box to the left of the control name.

   To view controls with .ocx file name extensions, select the **Controls** tab. To view insertable objects, such as a Microsoft Excel Chart, select the **Insertable Objects** tab.

3. Choose **OK** to close the **Components** dialog box. All of the ActiveX controls that you selected will now appear in the toolbox.

Figure 4.3 The Components dialog box



To add ActiveX controls to the Components dialog box, choose the Browse button, and search other directories for files with a .ocx file name extension. When you add an ActiveX control to the list of available controls, Visual Basic automatically selects the check box.

Note Each ActiveX control is accompanied by a file with an .oca extension. This file stores cached type library information and other data specific to the control. The .oca files are typically stored in the same directory as the ActiveX controls and are recreated as needed (file sizes and dates may change).

Removing Controls from a Project

To remove a control from a project

1. From the **Project** menu, choose **Components**.

   The **Components** dialog box is displayed.

2. Clear the check box next to each control you want to remove.

   The control icons will be removed from the toolbox.

Note You cannot remove any control from the toolbox if an instance of that control is used on any form in the project.

Using Other Applications' Objects

You can also use objects from other applications, such as those included in the Microsoft Excel object library, either as controls in the toolbox or as programmable objects in your code. To add objects to the toolbox, see "Adding Controls to a Project" earlier in this chapter.

To make another application's objects available in your code, but not as controls, set a reference to that application's object library.

To add a reference to another application's object library

1. From the **Project** menu, choose **References**.

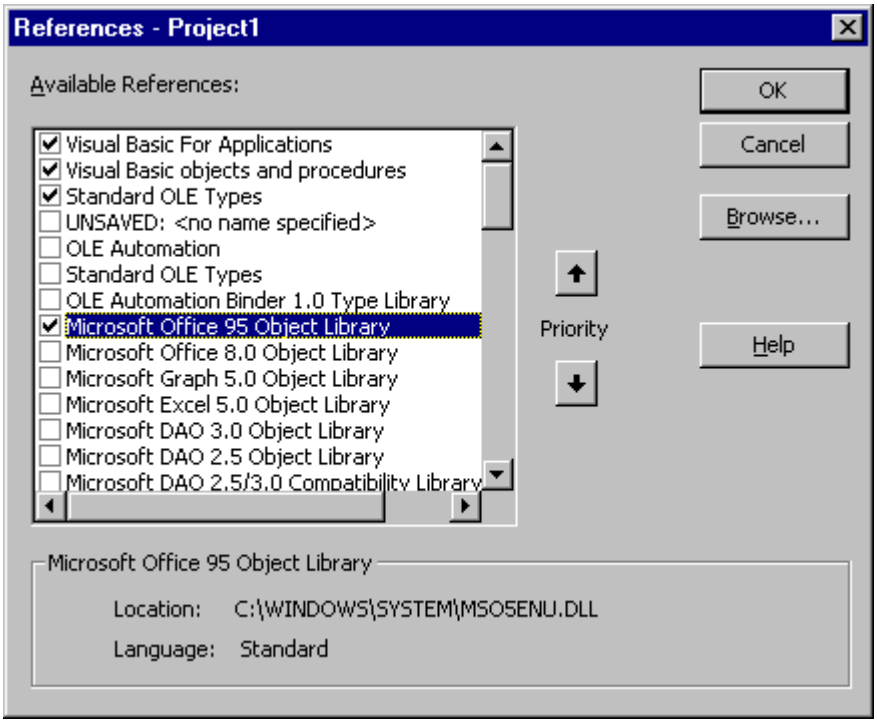
   The **References** dialog box is displayed, as shown in Figure 4.4.

2. Select the check box next to each reference you want to add to your project.

   To add references to applications not listed in the **References** dialog box, choose the **Browse** button, and then select the application.

3. Choose **OK** to add the selected references to your project.

Figure 4.4 The References dialog box



If you are not using any objects in a referenced library, you should clear the check box for that reference to minimize the number of object references Visual Basic must resolve, thus reducing the time it takes your project to compile.

Once you have set references to the object libraries you want, you can find a specific object and its methods and properties in the Object Browser by choosing Object Browser from the View menu. You can use any object listed in the Object Browser in your code.

For More Information " For information on the Object Browser, see "Finding Out About Objects" in "Programming with Objects."

Using a Resource File

A resource file allows you to collect all of the version-specific text and bitmaps for an

application in one place. This can include constant declarations, icons, screen text, and other material that may change between localized versions or between revisions or specific configurations.

To add a file to a project

1. From the **Project** menu, select **Add File**.

   The **Add File** dialog box is displayed.

2. Select an existing resource file (.res) and choose **Open**.

A single project can have only one resource file; if you add a second file with a .res extension, an error occurs.

For More Information For more information on the contents of a resource file, see "International Issues."

Making and Running an Executable File

You can make an executable file (.exe) from Visual Basic using the following procedure.

To make an executable file in Visual Basic

1. From the **File** menu, choose **Make** *projectname* **.exe** where *projectname* is the application name for the project.

2. Type a file name, or browse through the directories and select an existing file name to overwrite an existing executable with a newer version.

3. By clicking the **Options** button, you can also specify a number of version-specific details about the executable file in the **Project Properties** dialog box.

4. If you want to modify the version number of the project, set the appropriate **Major**, **Minor**, and **Revision** numbers. Selecting **Auto Increment** will automatically step the **Revision** number each time you run the **Make** *projectname* **.exe** command for this project.

5. To specify a new name for the application, under **Application**, type a new name in the **Title** box. If you want to specify a new icon, choose one from the list.

6. You can also enter version-specific commentary on a variety of issues under the **Version Information** box (comments, company name, trademark and copyright information, and so on) by selecting a topic from the list box and entering information in the text box.

7. Choose **OK** to close the **Project Properties** dialog box, and then choose **OK** in the **Make** *appname* **.exe** dialog box to compile and link the executable file.

You can run the executable file like any other Windows-based application: double-click the icon for the executable file.

Note Building an executable file from the command line in a DOS session can be useful when you want to compile a project programmatically. In a batch file, type:

Vb6 /make *projectname***[.vbp] [***exename***]**

For *projectname*, type the name of the project file. Use the variable *exename* to rename the resulting executable file.

Conditional Compilation

Conditional compilation lets you selectively compile certain parts of the program. You can include specific features of your program in different versions, such as changing the date and currency display filters for an application distributed in several different languages.

For More Information To learn more about conditional compilation, see "Using Conditional Compilation" in "More About Programming."

Setting Project Options

Visual Basic allows you to customize each project by setting a number of properties. Use the Project Properties dialog box, accessible through the Project Properties command on the Project menu. Property settings are saved to the project (.vbp) file.

The following table describes some of the options you can set.

Option	Description
Startup Object	The first form that Visual Basic displays at run time, or Sub Main ( ).
Project Name	Identifies the project in code. It can't contain periods (.), spaces, or start with a nonalphabetic character. For a public class name, the project name and class name cannot exceed a total of 37 characters.
Help File	The name of the Help file associated with the project.
Project Help Context ID	The context ID for the specific Help topic to be called when the user selects the "?" button while the application's object library is selected in the Object Browser.
Project Description	A user-friendly name for the project. Displayed in the

Many other options are also available, including those for compiling, components, and multithreading. When you're ready to access some of the more advanced options, you can find more information by searching Help.

For More Information To learn about setting environment options that affect all projects, see "Developing An Application in Visual Basic."

Using Wizards and Add-Ins

Visual Basic allows you to select and manage *add-ins*, which are extensions to Visual Basic. These extensions add capabilities to the Visual Basic development environment, for example, special source code control capabilities.

Microsoft and other developers have created add-ins you can use in your applications. Wizards are a type of add-in that can simplify certain tasks, such as creating a form. Several wizards are included in Visual Basic.

To have an add-in appear on the Add-In Manager dialog box, the developer of the add-in must ensure that it is installed properly.

Using the Add-In Manager

You can add or remove an add-in to your project by using the Add-In Manager, which is accessible from the Add-Ins menu. The Add-In Manager dialog box lists the available add-ins.

To install an add-in

1. From the **Add-Ins** menu, choose **Add-In Manager**.
2. Highlight an add-in from the list and click the desired behaviors in Load Behavior. To unload an add-in or prevent it from loading, clear all Load Behavior boxes.
3. When you are finished making your selections, choose **OK**.

   Depending upon your Load Behavior selections, Visual Basic connects the selected add-ins and disconnects the cleared add-ins.

Visual Basic saves your add-in selections between editing sessions.

Note Selecting an add-in may add menu items to the Visual Basic Add-Ins menu.

Using Wizards

Wizards make working with Visual Basic even easier by providing task-specific assistance. For example, the Application Wizard included in Visual Basic helps you to create the framework for an application by presenting a series of questions or choices. It generates the forms and the code behind the forms based on your choices; all you need to do is add code for your own specific functionality.

The Professional and Enterprise editions of Visual Basic include other wizards, including a Data Form Wizard for creating forms to be used with databases, and an ActiveX Document Wizard for converting forms for use in Internet applications.

Wizards are installed or removed using the Add-in Manager. Once installed, they will appear as selections on the Add-Ins menu. Some of the wizards also appear as icons in the related dialog boxes; for example, the Application Wizard can also be accessed using its icon in the New Project dialog box.

To start the Application Wizard

- From the **Add-Ins** menu, choose **Application Wizard**.

  –or–

1. From the **File** menu, choose **New Project**.
2. Select the **Application Wizard** icon.
