

applications. Before you can do that, you must first be sure that the application provides an object library.

#### To see if an application provides an object library

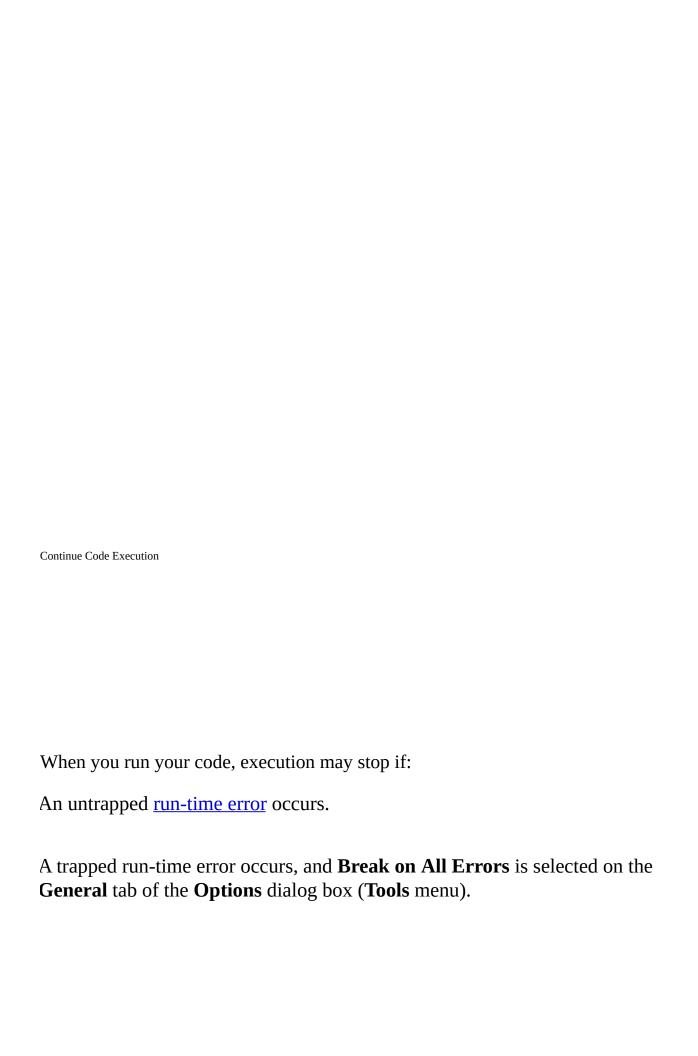
From the **Tools** menu, choose **References** to display the **References** dialog box.

The **References** dialog box shows all object libraries registered with the operating system. Scroll through the list for the application whose object library you want to reference. If the application isn't listed, you can use the **Browse** button to search for object libraries (\*.olb and \*.tlb) or executable files (\*.exe and \*.dll on Windows). References whose check boxes are checked are used by your project; those that aren't checked are not used, but can be added.

#### To add a object library reference to your project

Select the object library reference in the **Available References** box in the **References** dialog box and click **OK**.

Your Visual Basic project now has a reference to the application's object library. If you open the <u>Object Browser</u> (press F2) and select he application's library, it displays the objects provided by the selected object library, as well as each object's <u>methods</u> and <u>properties</u>. In the **Object Browser**, you can select a <u>class</u> in the **Classes** box and select a method or property in the **Members** box. Use copy and paste to add the syntax to your code.



A previously set <u>breakpoint</u> is encountered.

A **Stop** statement in your code is encountered, switching the mode to <u>break</u> mode.

An **End** statement in your code is encountered, switching the mode to <u>design</u> time.

You halt execution manually at a given point.

A <u>watch expression</u>, which you set to break when the value has changed or break when the value is true, is encountered.

#### To halt execution manually

To switch to break mode, choose **Break** (CTRL+BREAK) from the **Run** menu, or use the toolbar shortcut: .

To switch to design time, choose **Reset <projectname>** from the **Run** menu, or use the toolbar shortcut:

#### To continue execution when your application has halted

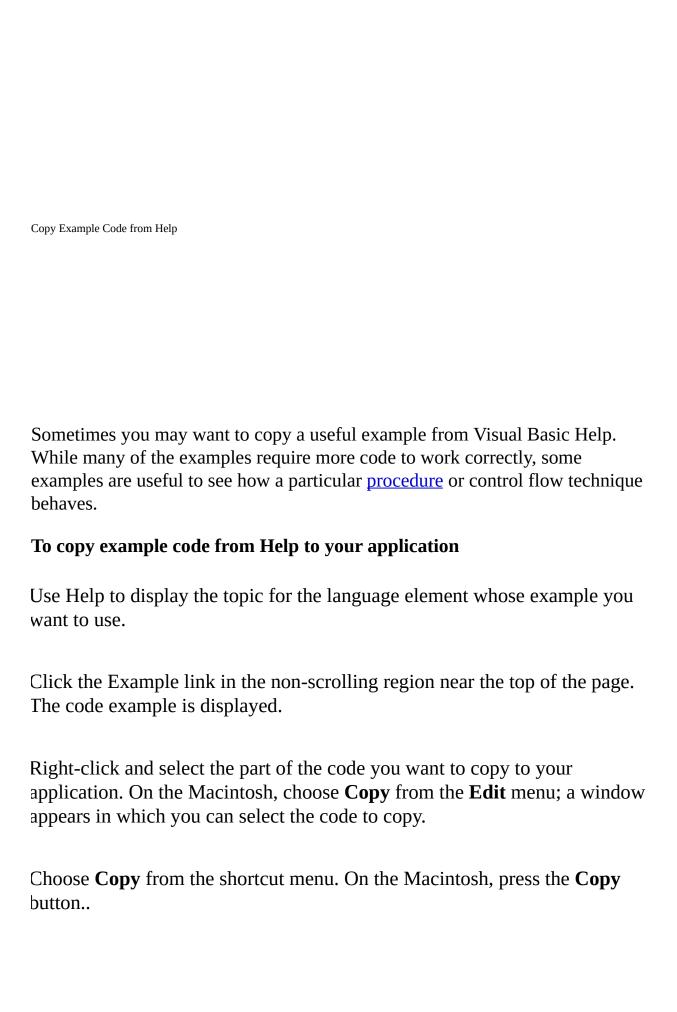
On the **Run** menu, click **Continue** (F5), or use the toolbar shortcut: .

On the **Debug** menu, click **Step Into** (F8), **Step Over** (SHIFT+F8), **Step Out** (CTRL+SHIFT+F8), or **Run To Cursor** (CTRL+F8)(.

To continue execution when your application has halted because of a handled error

Press ALT+F8 to step through the error-handler.

Press ALT+F5 to resume execution by running through the error-handler.



Move the focus back to the **Code** window and position the mouse pointer where you want the code example to be inserted.

Right-click again and choose **Paste** to insert the code example into the **Code** window.

**Fip** You can also press CTRL+C to copy a selected example in a Help window. Press CTRL+V to paste the example into the **Code** window.

Create a Procedure
Code within a module is organized into procedures. A procedure tells the application how to perform a specific task. Use procedures to divide complex code tasks into more manageable units.
To create a procedure by writing code
Open the module for which you want to write the procedure.
You can create a <b>Sub</b> , <b>Function</b> , or <b>Property</b> procedure.
Type <b>Sub</b> , <b>Function</b> , or <b>Property</b> .  Press F1 to get Help with syntax, if necessary.
Type code for the procedure.

Visual Basic concludes the procedure with the appropriate End Sub, End Function, or End Property statement.

### To create a procedure using the Insert Procedure dialog box

Open the module for which you want to write the procedure.

On the **Insert** menu, click **Procedure**.

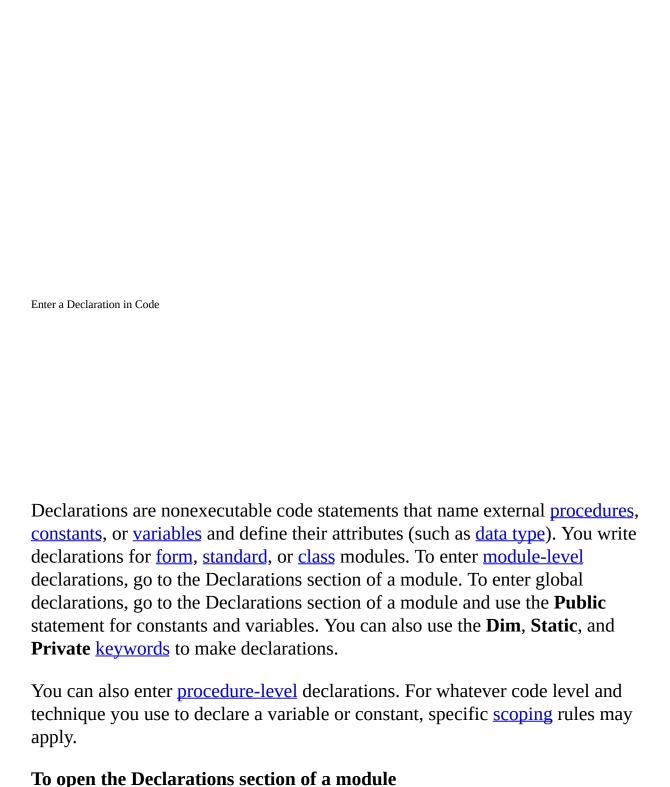
Type the name for the procedure in the **Name** box of the **Insert Procedure** dialog box.

Select the type of procedure you want to create: Sub, Function, or Property.

Set the procedure's scope to either Public or Private.

You can select the **All Local Variables as Statics** to add the **Static** keyword to the procedure definition.

Click **OK**.



In the **Project** window, select the form, standard, or class module you want

## to open and click the **View Code** button.

- Or -

Right-click and choose **View Code** from the context menu.

### In the Object box, select (General).

 $\label{eq:conditional} \Gamma he \ \underline{Procedure \ box} \ automatically \ displays \ \textbf{(Declarations)}.$ 

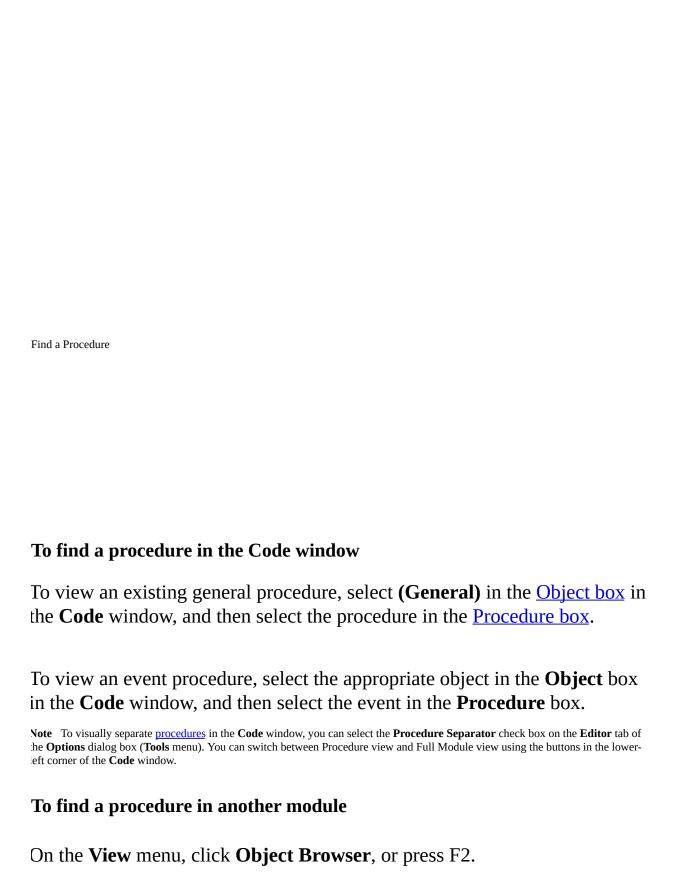
Enter one or more declarations.

Execute a Specific Statement
While execution of your code is halted, you can control the execution sequence of <u>statements</u> within a <u>procedure</u> . You can resume execution at a statement you choose without executing any intervening code.
To set the next statement to be executed
In the <b>Code</b> window, position the insertion point anywhere within the statement.
On the <b>Debug</b> menu, click <b>Set Next Statement</b> (CTRL+F9).  - or on Windows –
Position the mouse pointer in the <u>margin indicator</u> next to the current execution point.
Drag the yellow arrow in the margin indicator to the statement you want to

#### execute next.

**Note** You can only skip to statements within the same procedure.

Used in combination with **Step Into**, executing specific statements with the **Set Next Statement** command enables you to step through procedures one statement at a time, and to closely examine your code. It's also helpful for correcting or avoiding <u>run-time error</u> conditions.



Select the project in the **Project/Library** box.

Select the module in the **Classes** list.

Double-click the procedure name in the **Members of** list.

The selected procedure is displayed in the  ${f Code}$  window.

You can use the following keyboard shortcuts:

Press	То
CTRL+DOWN ARROW	Display the next procedure.
CTRL+UP ARROW	Display the previous procedure.
PAGE DOWN	Page down through the procedures in your code.
PAGE UP	Page up through the procedures in your code.
F2	Display the <b>Object Browser</b> .

Fin	d a Variable Definition
To	o view the definition of a variable
In	the <b>Code</b> window, select the <u>variable</u> whose definition you want to
Fr	rom the <b>View</b> menu, choose <b>Definition</b> (SHIFT+F2).
To	return the mouse pointer to its previous position
Oı	n the <b>View</b> menu, click <b>Last Position</b> (CTRL+SHIFT+F2).

#### To replace text in a module

On the **Edit** menu, click **Replace**.

Γhe **Replace** dialog box appears.

In the **Find What** box, type the text you want to search for.

In the **Replace With** box, type the replacement text.

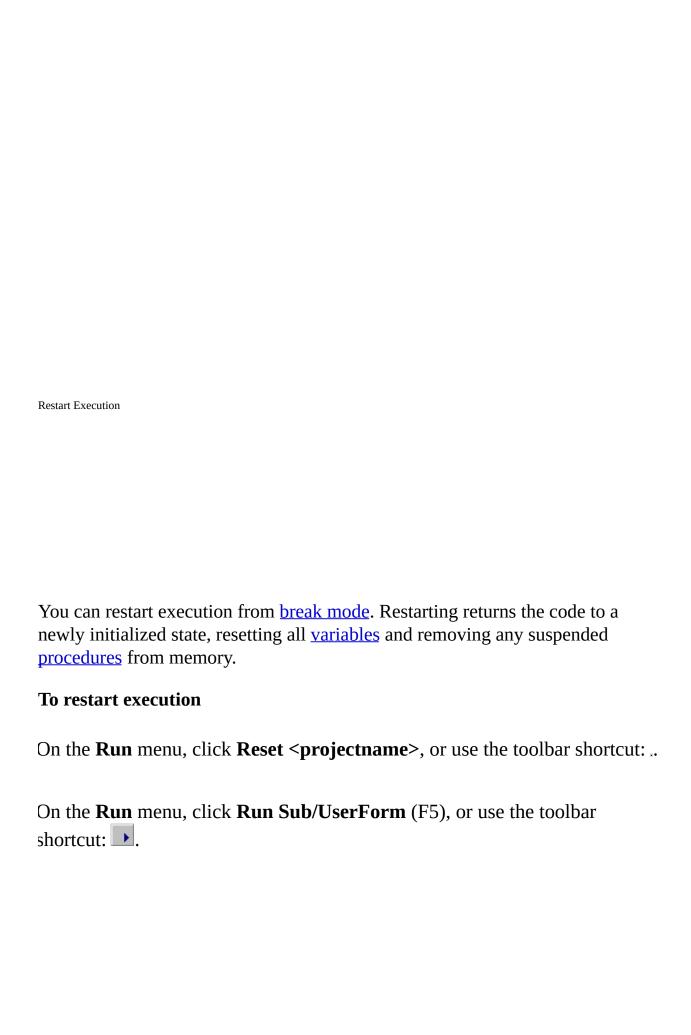
Select a **Search** option to specify where to look for the text.

Select a direction from the **Direction** list to specify the direction of the search.

To set limits on the search, select:

- Find Whole Word Only to search for the complete word by itself, and not as part of another word.
- Match Case for an exact match.
- Use Pattern Matching to use wildcard characters.

Choose **Find Next** if you want to confirm the change before replacing the text; choose **Replace** to replace the highlighted occurrence of the found text and automatically perform a **Find Next**; or choose **Replace All** to change all occurrences of the search text automatically.



### To search for specific text in a module

On the **Edit** menu, click **Find**.

Гhe **Find** dialog box appears.

In the **Find What** box, type the text you want to search for (if you didn't select the text before displaying the dialog box).

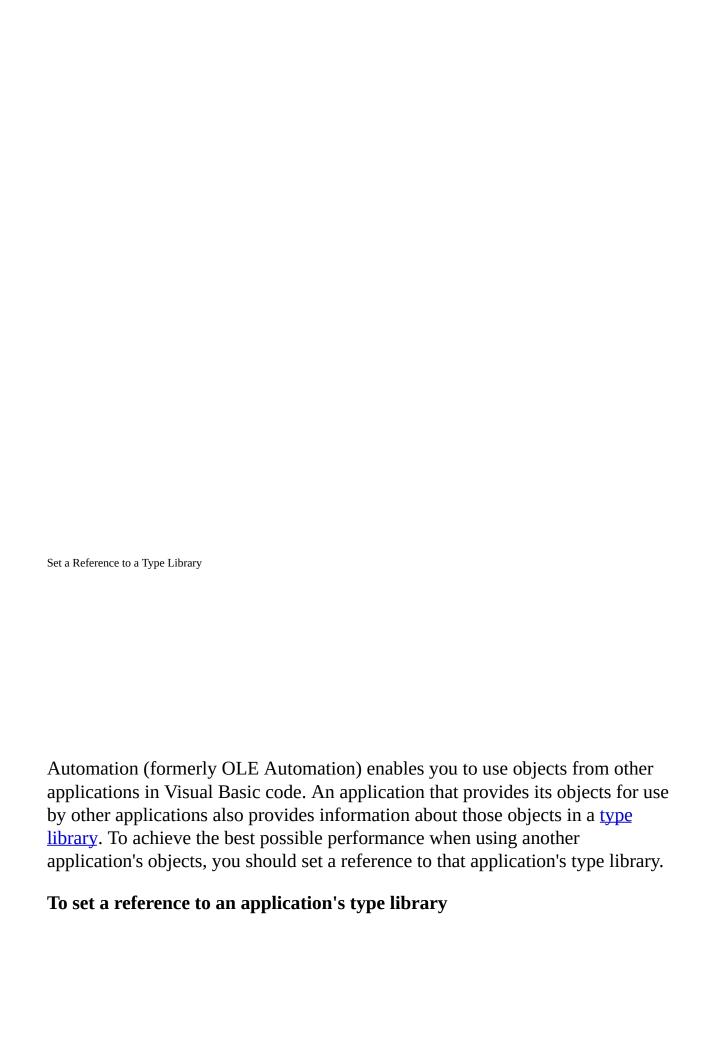
Select a **Search** option to specify where to look for the text.

Select a direction from **Direction** list to specify the direction of the search.

To set limits on the search, select:

- Find Whole Word Only to search for the complete word by itself and not as part of another word.
- Match Case to find an exact match.
- Use Pattern Matching to use wildcards or ranges.

Choose **Find Next**.



#### Click **References** on the **Tools** menu.

Select the check boxes for the applications with type libraries you want to reference.

If you are writing code that manipulates objects in another application, you should set a reference to that application's type library for best possible access to those objects. You don't have to set a reference to use another application's objects, but doing so provides several advantages for your application.

Your code will run faster if you set a reference to another application's type library before you work with its objects. If you set a reference, you can declare an <u>object variable</u> representing an object in the other application as its most specific type. For example, if you are writing code to work with Microsoft Excel objects, you can declare an object variable of type **Excel.Application** if you created a reference to the Microsoft Excel type library. The following code is the fastest way to create a variable to represent the Microsoft Excel **Application** object.

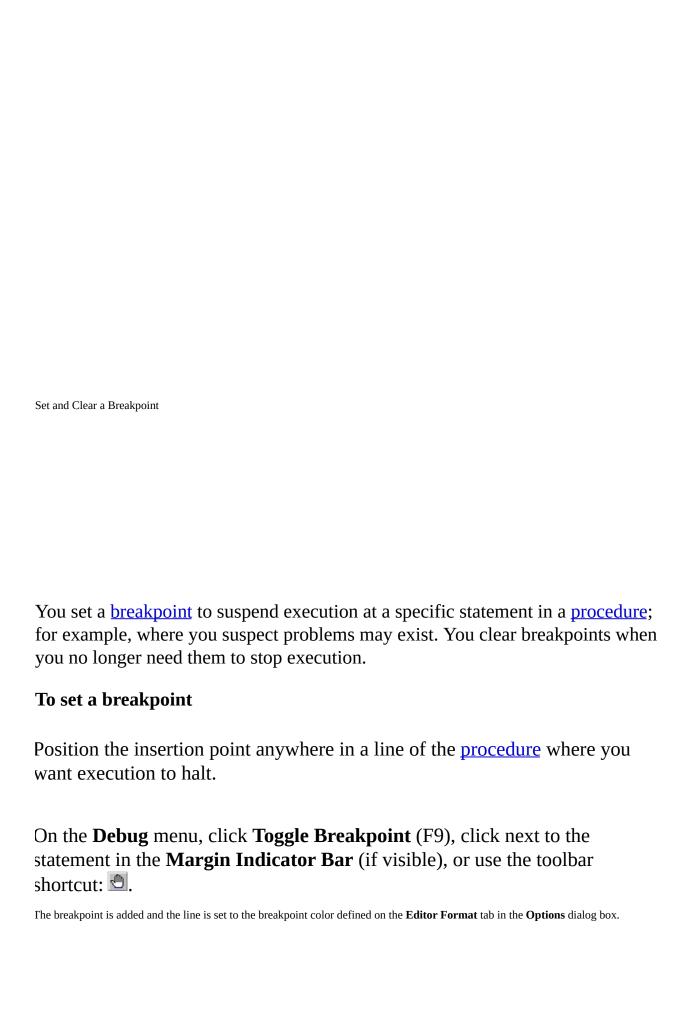
### Dim appXL As Excel.Application

If you haven't set a reference to the Microsoft Excel type library, you must declare the <u>variable</u> as a generic variable of type <u>Object</u>. The following code runs more slowly.

### Dim appXL As Object

If you set a reference to an application's type library, all of its objects and their methods and properties are listed in the **Object Browser**. This makes it easy to determine what properties and methods are available to each object.

For Microsoft applications that can also serve as Automation servers, you can set references to their type libraries from another application, and control their objects from that application.



If you set a breakpoint on a line that contains several statements separated by colons (:), the break always occurs at the first statement on the line.

#### To clear a breakpoint

Position the insertion point anywhere on a line of the procedure containing the breakpoint.

From the **Debug** menu, choose **Toggle Breakpoint** (F9), or click next to the statement in the **Margin Indicator Bar** (if visible.)

The breakpoint is cleared and highlighting is removed.

#### To clear all breakpoints in the application

From the **Debug** menu, choose **Clear All Breakpoints** (CTRL+SHIFT+F9).

**Note** Breakpoints set in code are not saved when you save your code.



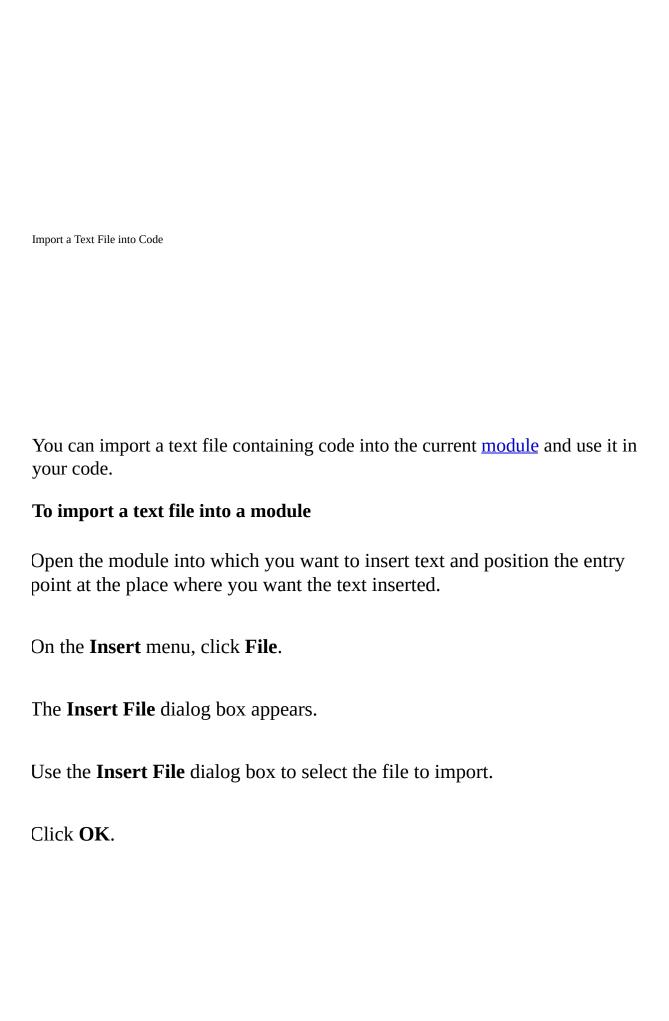
#### **To set Project Properties**

From the **Tools** menu, choose projectname> Properties

Use the **General** tab in the **projectname> Properties** dialog box to specify the following:

- Name of your project.
- Description of your <u>project</u>.
- Name of the Help file associated with your project
- Context ID for the specific Help topic to be called when the user clicks the **Help** button while the application's <u>object library</u> is selected in the **Object Browser**.

Use the **Protection** tab in the **projectname Properties** dialog box to lock the project from viewing by others and specify a password for access to project properties. After you set protection, you must save and close your project for the protection to take effect.





**Editor** tab to specify Code window and Project window settings.

**Editor Format** tab to specify the appearance of your code.

**Genera**l tab to specify form, error handling, and compile settings for your project.

**Docking** tab to specify whether a window is attached or "anchored" to one edge of other dockable or application windows.

#### **To set Environment options**

On the **Tools** menu of the Visual Basic editor, click **Options**. Each option is described in the following tables.

#### **Editor**

Option	Description
Auto Syntax Check	Visual Basic automatically verifies correct syntax after you enter a line of code.
Require Variable Declaration	Explicit variable declarations are required in modules.
Auto Indent	After tabbing the first line of code, all subsequent lines start at that tab location.
Tab Width	The tab width, which can range from $1 - 32$ spaces. (Default is 4 spaces.)
Default to Full Module View	<u>Procedures</u> for new modules are displayed in the <b>Code</b> window as a single, scrollable list or one procedure at a time.
Procedure Separator	Display separator bars at the end of each procedure in the <b>Code</b> window.
Auto List Members	At the insertion point, Visual Basic displays information that logically completes a

	statement.
Auto Quick Info	Information about functions and their <u>arguments</u> is displayed as you type.
Auto Data Tips	Automatically display the value of any <u>variable</u> on which you place the mouse pointer. Available only in <u>break mode</u> .
Drag-Drop in Text Editing	Code elements can be dragged from the <b>Code</b> window into the <b>Immediate</b> or <b>Watch</b> windows.

#### **Editor Format**

Option	Description
Foreground, Background, and	The color of different categories of text listed
Indicator	in the <b>Code Colors</b> list.
Font	The font used for displaying code.
Size	The size of the font used for code.
Margin Indicator Bar	Display the Margin Indicator Bar.

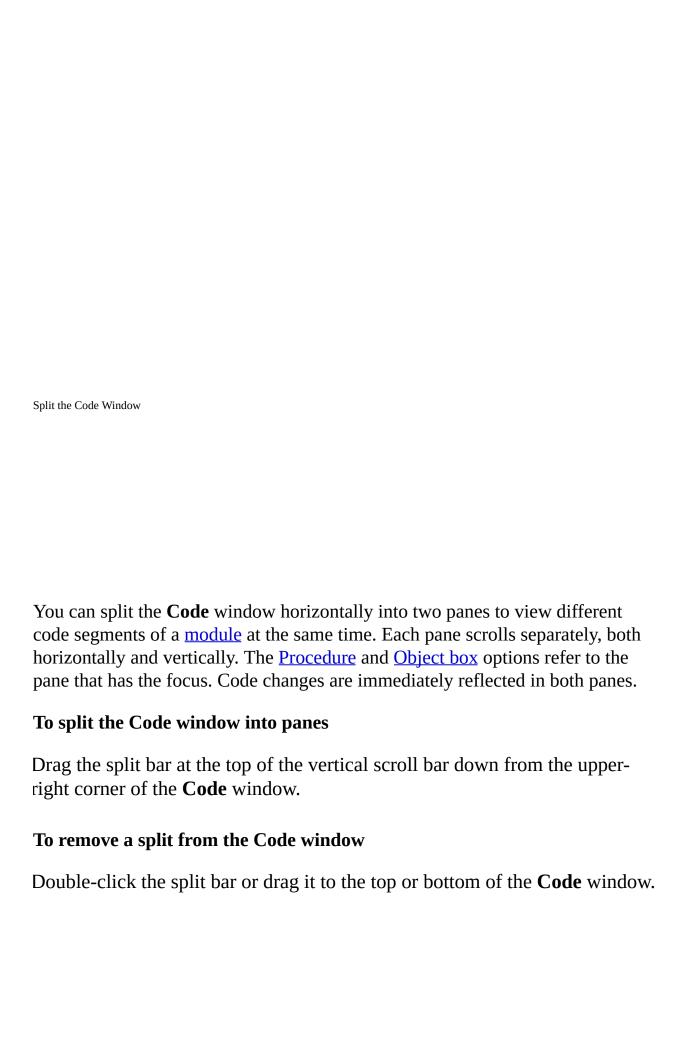
### General

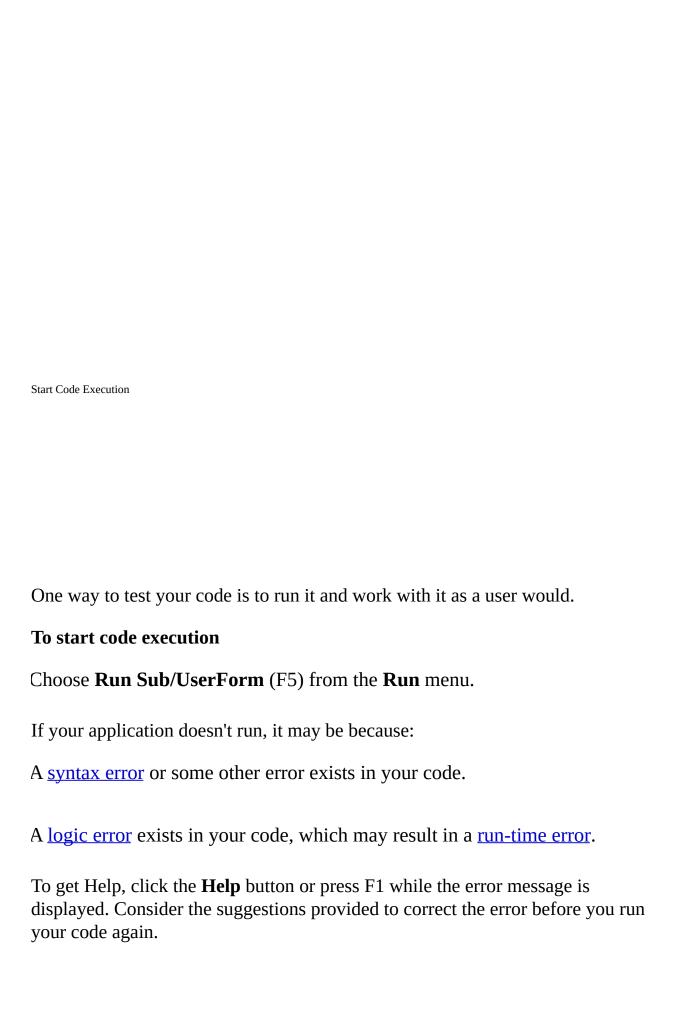
Option	Description
Show Grid	Display a grid on a form.
Grid Units	Lists the unit of measurement for units in the grid.
Width	The width of the grid cells on a form.
Height	The height of the grid cells on a form.
Align Controls to Grid	Automatically position the outer edge of controls on the closest grid lines.
Show ToolTips	Display ToolTips for toolbar buttons.
Collapse Proj. Hides Windows	Automatically close the project, <b>UserForm</b> , object, or module windows when a project is

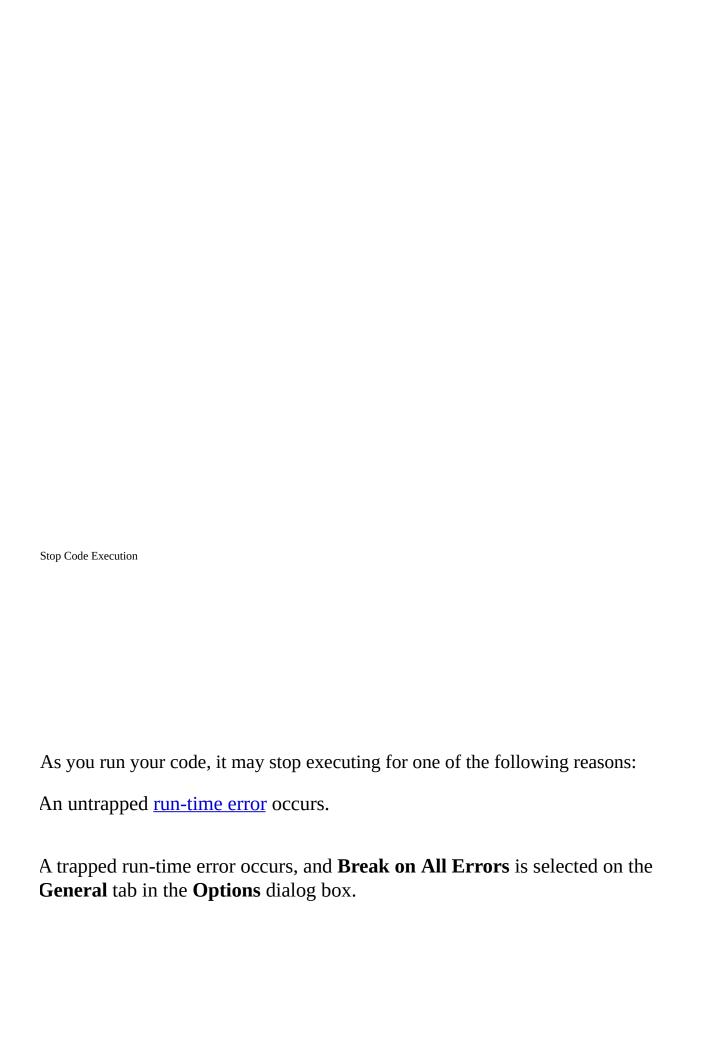
	collapsed in the <b>Project Explorer</b> .
<b>Notify Before State Loss</b>	Display a message that a requested action will cause all module-level variables to be reset for a running project.
Break on All Errors	Any error causes the project to enter break mode, whether or not an error handler is active, and whether or not the code is in a class module.
Break in Class Module	Any unhandled error produced in a class module causes the project to enter break mode at the line of code which produced the error.
Break on Unhandled Errors	Any other unhandled error causes the project to enter break mode.
Compile On Demand	A project is fully compiled before it starts, or code is compiled as needed.
Background Compile	Use idle time during run time to finish compiling the project in the background. (Available only if <b>Compile On Demand</b> is set.)

## Docking

Option	Description
The check box for the	A window can be anchored to an adjacent
appropriate window	dockable window or the Visual Basic Editor
	window.







A <u>breakpoint</u> is encountered.

A **Stop** statement is encountered in your code, switching the mode to <u>break</u> mode.

An **End** statement is encountered in your code, switching the mode to <u>design</u> time.

You halt execution manually at a given point.

A <u>watch expression</u> that you set to break if its value changes or becomes true is encountered.

#### To halt execution manually

To switch to break mode, from the **Run** menu, choose **Break** (CTRL+BREAK), or use the toolbar shortcut: ...

To switch to design time, from the **Run** menu, choose **Reset** <**projectname**>, or use the toolbar shortcut: .

### To continue execution when your application has halted

From the **Debug** menu, choose **Step Into** (F8), **Step Over** (SHIFT+F8), **Step Out** (CTRL+SHIFT+F8), or **Run To Cursor** (CTRL+F8.

Trace Code Execution

You trace code execution because it may not always be obvious which statement is executed first. Use these techniques to trace the execution of code:

**Step Into**: Traces through each line of code and steps into <u>procedures</u>. This allows you to view the effect of each statement on <u>variables</u>.

**Step Over**: Executes each procedure as if it were a single statement. Use this instead of **Step Into** to step across procedure calls rather than into the called procedure.

**Step Out**: Executes all remaining code in a procedure as if it were a single statement, and exits to the next statement in the procedure that caused the procedure to be called initially.

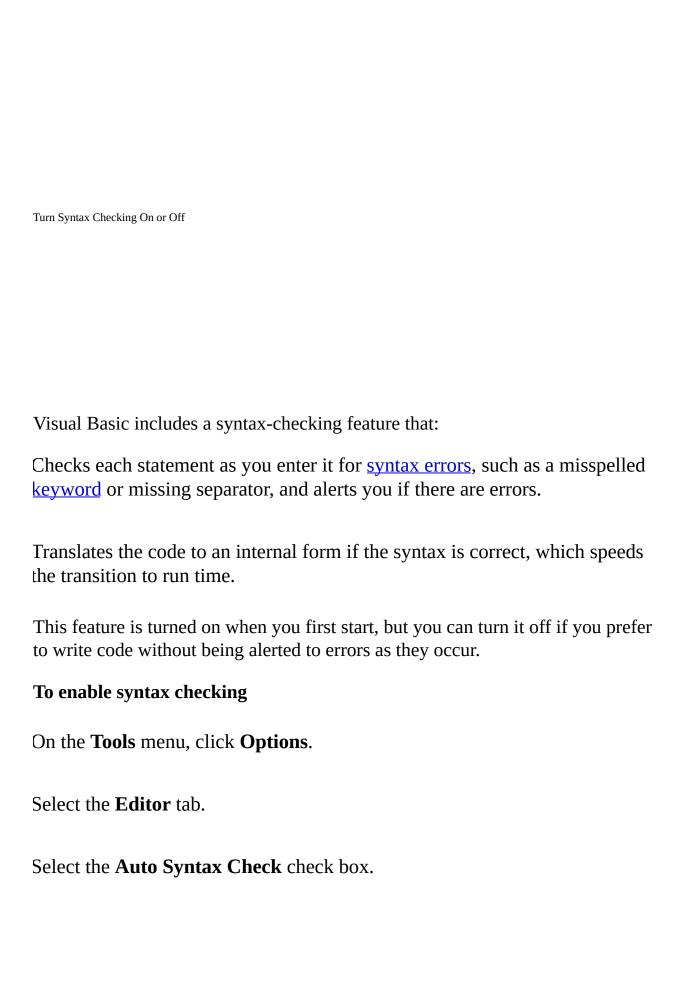
**Run To Cursor**: Allows you to select a statement in your code where you want execution to stop. This allows you to "step over" sections of code, for example, large loops.

To trace execution from the current statement

From the **Debug** menu, choose **Step Into** (F8), **Step Over** (SHIFT+F8), **Step Out** (CTRL+SHIFT+F8), or **Run To Cursor** (CTRL+F8).

### To trace execution from the beginning of the program

From the **Debug** menu, choose **Step Into** (F8), **Step Over** (SHIFT+F8), **Step Out** (CTRL+SHIFT+F8), or **Run To Cursor** (CTRL+F8).



Click **OK**.

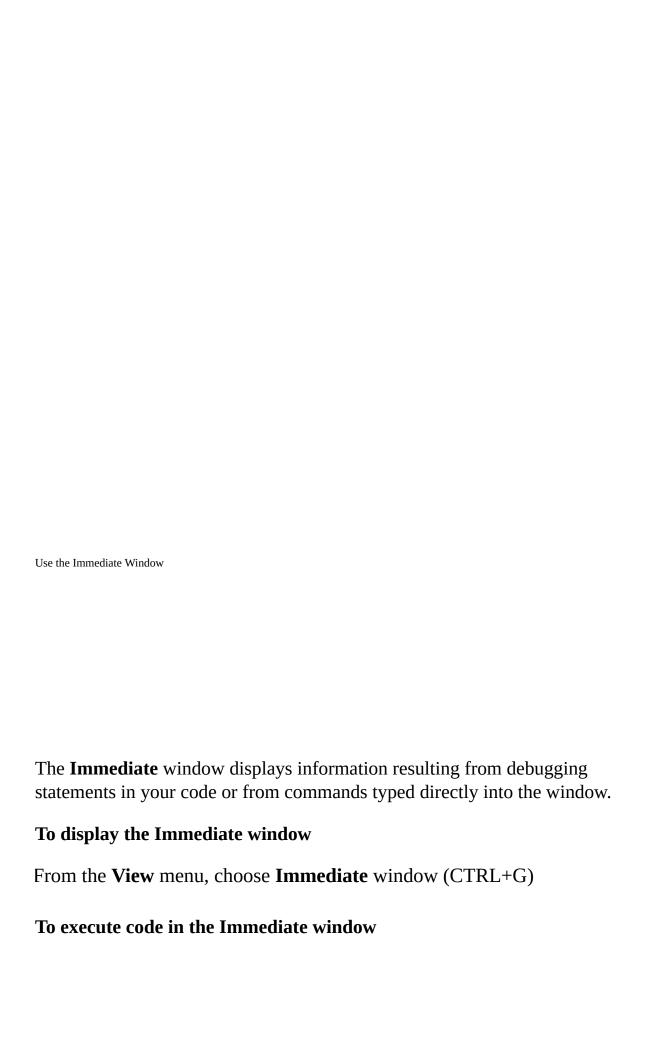
To disable syntax checking

Click **Options** on the **Tools** menu.

Select the **Editor** tab.

Clear the **Auto Syntax Check** check box.

Click **OK**.



Type a line of code in the **Immediate** window.

Press ENTER to execute the statement.

Use the **Immediate** window to:

Test problematic or newly written code.

Query or change the value of a <u>variable</u> while running an application. While execution is halted, assign the variable a new value as you would in code.

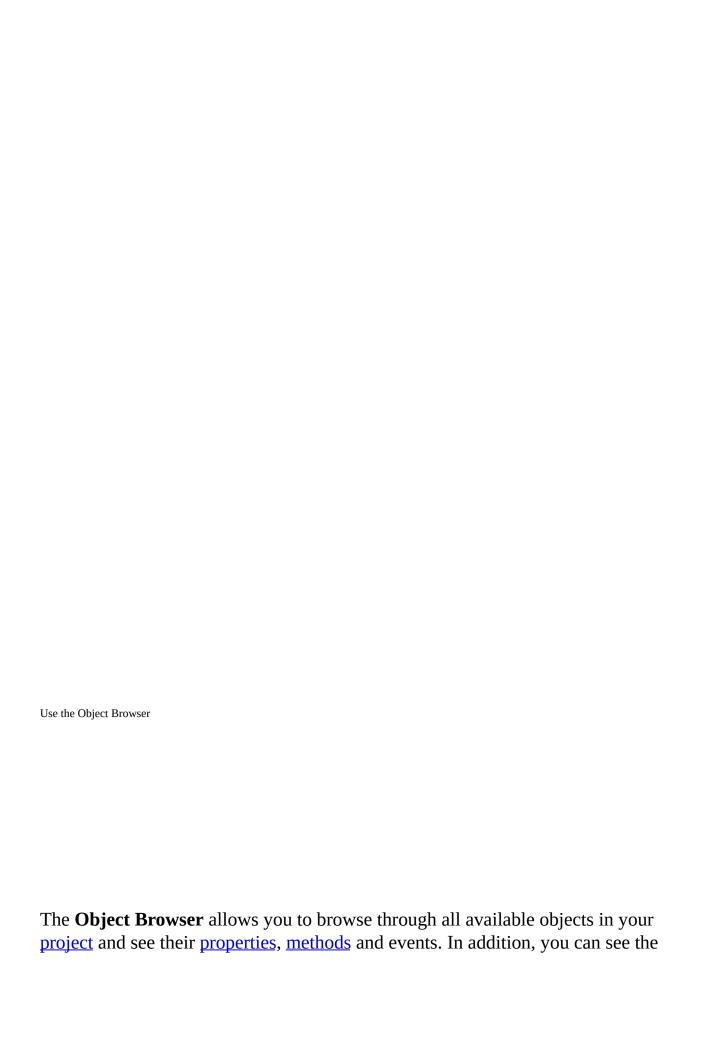
Query or change a property value while running an application.

Call procedures as you would in code.

View debugging output while the program is running.

Note Immediate window statements are executed in a context — that is, as if they are entered in a specific module.

If you need help on syntax for functions, statements, <u>properties</u>, or <u>methods</u> while working in the **Immediate** window, select the <u>keyword</u>, the property name, or the method name, and press F1.



procedures and constants that are available from object libraries in your project. You can easily display online Help as you browse. You can use the **Object Browser** to find and use objects you create, as well as objects from other applications.

You can get help for the **Object Browser** by searching for **Object Browser** in Help.

#### To navigate the Object Browser

Activate a module.

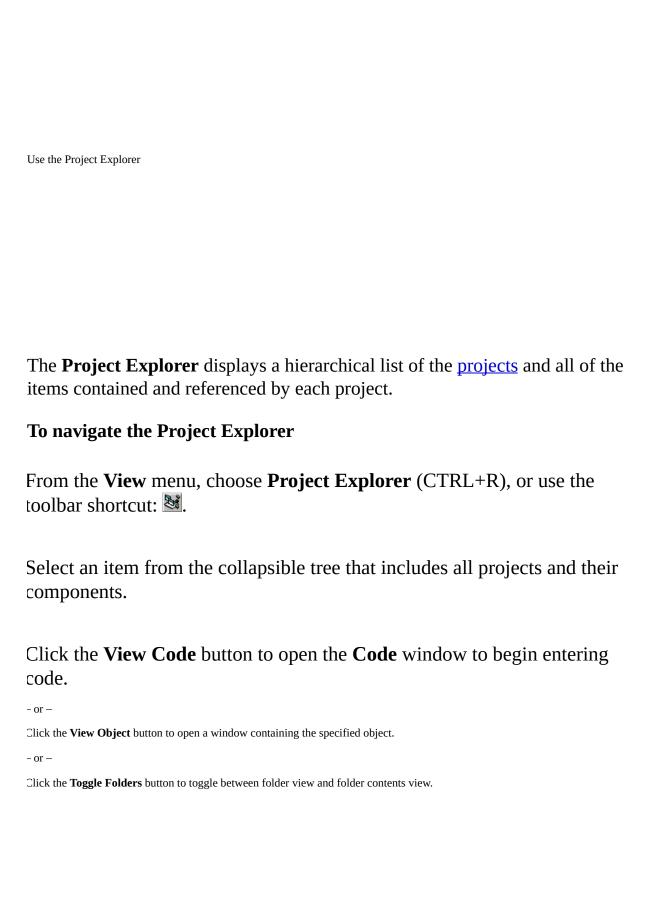
From the **View** menu, choose **Object Browser** (F2), or use the toolbar shortcut: **3**.

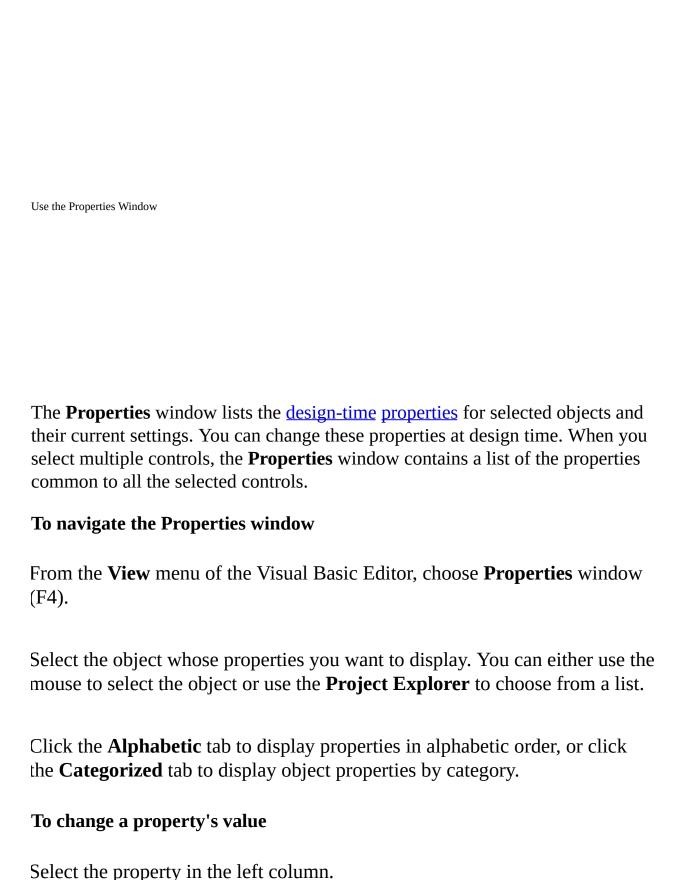
Select the name of the project or library you want to view in the **Project/Library** list.

Use the **Class** list to select the <u>class</u>; use the **Member** list to select specific members of your class or project.

View information about the class or member you selected in the **Details** section at the bottom of the window.

Use the **Help** button to display the Help topic for the class or member you selected.





# Change the property's value in the right column.

**Note** You can enter a property's value in the right column. For those properties that have a predefined set of values, click the value and then select one from the values displayed in the list box.