

TegoSoft Inc.

Copyright © 1994-1996 TegoSoft Inc. ® All Rights Reserved

Web Site: <http://www.tegosoft.com>

 e-mail: tegosoft@msn.com

Lesson 5 - Menus


Menus


Almost every Windows program has a menu. In this lesson you'll learn to implement menus.


Create the Directory of the MyMenu Program, Start Visual Basic 4, and Save the Form and the Project Files

In this lesson you are going to create several files. So first of all, let's create a directory where the files that you'll create during the course of this lesson are saved.


 Create the **C:\VBMyProg\Lesson05** directory.

 Start Visual Basic, and then select **New Project** from the **File** menu of Visual Basic.

 Select **Project** from the **View** menu of Visual Basic to display the Project window. Click the **View Form** button of the Project window to make **Form1** the selected form.


 While Form1 is the selected window, select **Save File As** from the **File** menu of Visual Basic.

Visual Basic responds by displaying the **Save File As** dialog box.

 Save the Form file as **MyMenu.frm** inside the **C:\VBMyProg\Lesson05** directory.


 Select **Save Project As** from the **File** menu of Visual Basic.

Visual Basic responds by displaying the **Save Project As** dialog box.

 Save the project file as **MyMenu.vbp** inside the C:\VBMyProg\Lesson05 directory.

Set the Properties of Form1

You'll now set the properties of **Form1**.

 Make the **Form1** window the selected window, and then press the F4 key on your keyboard to display the **Properties** window of Form1.


 Set the properties of **Form1** as follows:

Property	Setting
Name	frmMyMenu
Caption	The MyMenu Program
Height	4065
Width	4095
BackColor	White

Implementing a Menu

As it turns out, implementing a menu is a matter of knowing how to use the tools of Visual Basic.

You'll now implement a menu.

 While frmMyMenu is the selected window, select **Menu Editor** from the **Tools** menu of Visual Basic.

Visual Basic responds by displaying the Menu Editor dialog box shown in Figure 5.1.

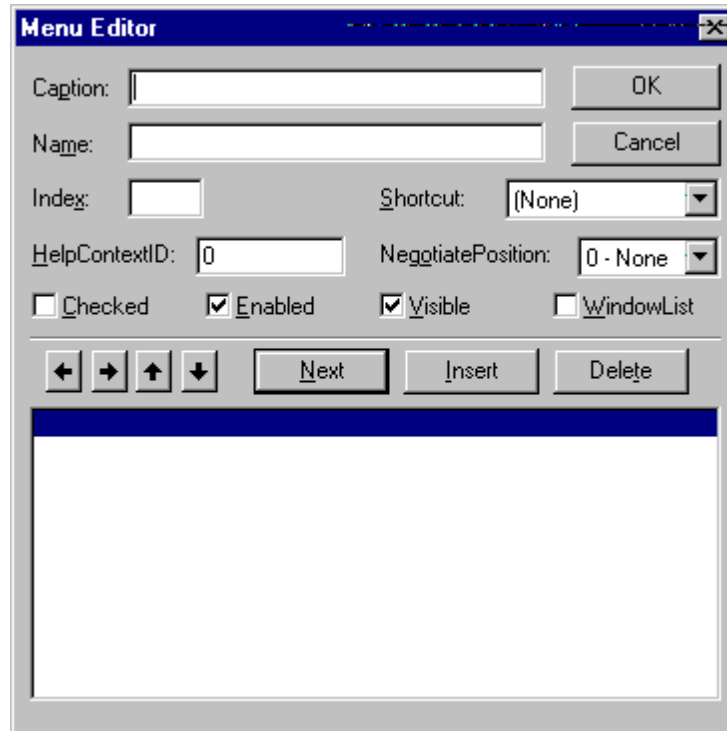




Figure 5.1. *The Menu Editor dialog box.*

What is the Menu Editor dialog box shown in Figure 5.1? This is a dialog box that lets you incorporate menus in the form. As you'll soon see, using the Menu Editor is actually very easy.

 Inside the **Caption** edit box of the Menu Editor dialog box type: **&File**

 Inside the **Name** edit box of the Menu Editor type: **mnuFile**

Your Menu Editor dialog box should now look as shown in Figure 5.2.

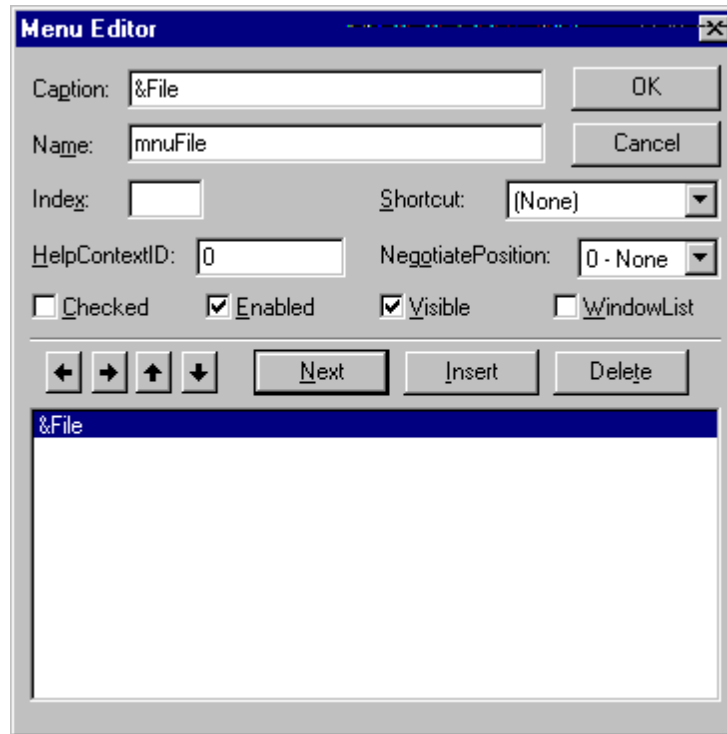



Figure 5.2. *Declaring the File menu.*

 Click the **Next** button inside the Menu Editor dialog box.

The Menu Editor responds by highlighting the second (empty) line that follows the &File line (see Figure 5.3.).

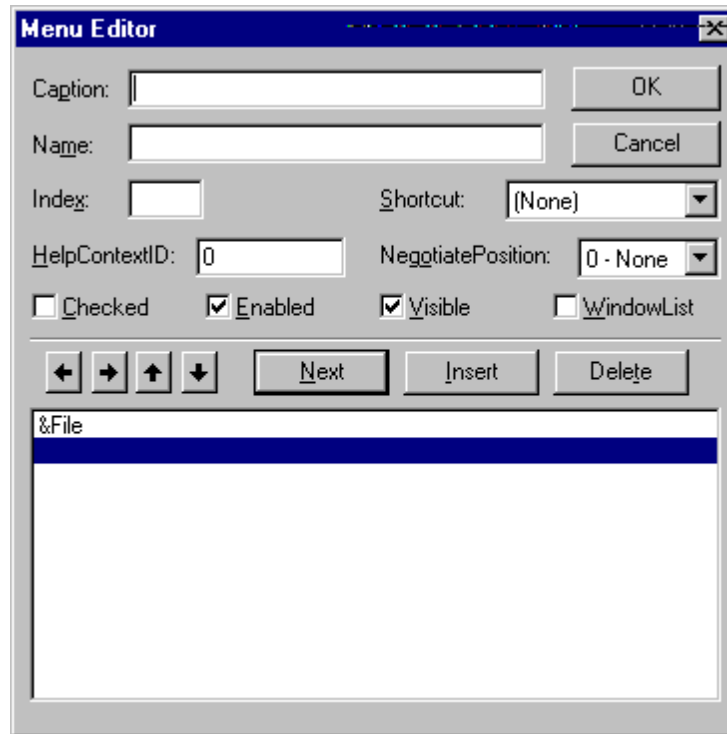



Figure 5.3. Adding a menu item.

 Inside the **Caption** edit box of Menu Editor dialog box type: **E&xit**

 Inside the **Name** edit box of Menu Editor dialog box type: **mnuExit**

Your Menu Editor dialog box should look as shown in Figure 5.4.

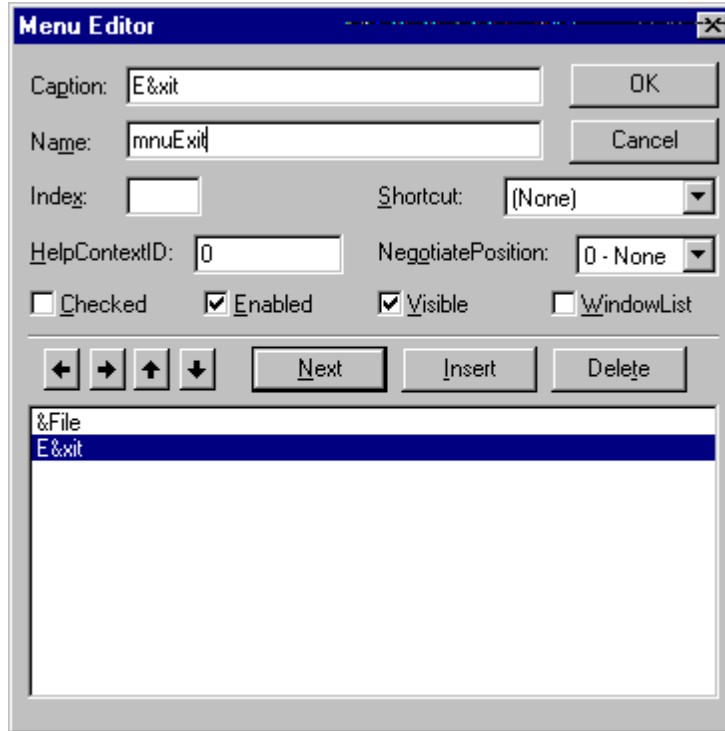


Figure 5.4. Adding the E&xit item.

Your objective now is to design a menu that looks like the one shown in Figure 5.5. That is, the menu bar has the **File** menu in it. When the user clicks the **File** menu, the menu opens, and the menu includes the **Exit** item in it. In other words, the **Exit** menu item is **inside** the File menu.

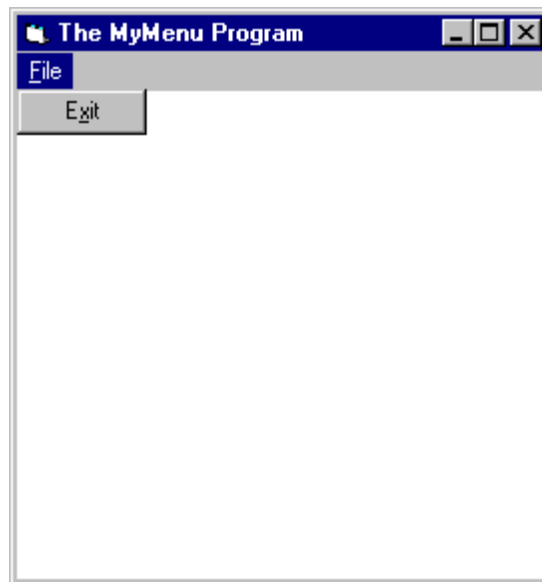



Figure 5.5. *The File menu has the Exit menu item in it.*

To make the **Exit** menu item part of the **File** menu item, you have to do the following:

 Make sure that the **Exit** menu item is highlighted inside the Menu Editor dialog box, and then click the button that has a picture of an arrow pointing to the right.

As shown in Figure 5.6, now the **Exit** item is one level to the right reference to the **File** item.

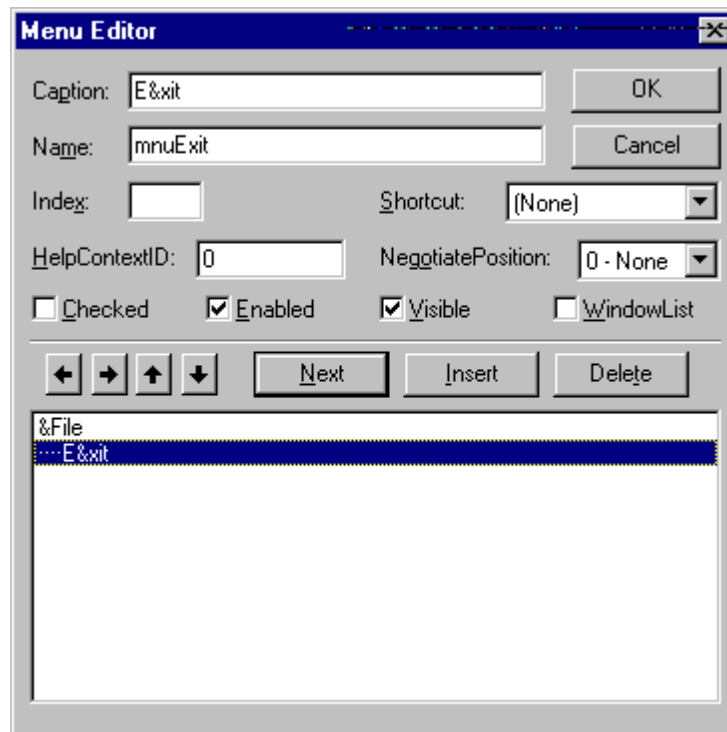




Figure 5.6. *The E&xit item is one level to the right from the &File item.*

Ok, let's see what you have accomplished so far:

 Click the **OK** button of the Menu Editor dialog box.

 Select **Save Project** from the **File** menu of Visual Basic.

Now let's execute the MyMenu program:

 Select **Start** from the **Run** menu of Visual Basic.

The window of the MyMenu program appears as shown in Figure 5.7. As you can see, the MyMenu program has the **File** menu on its menu bar.

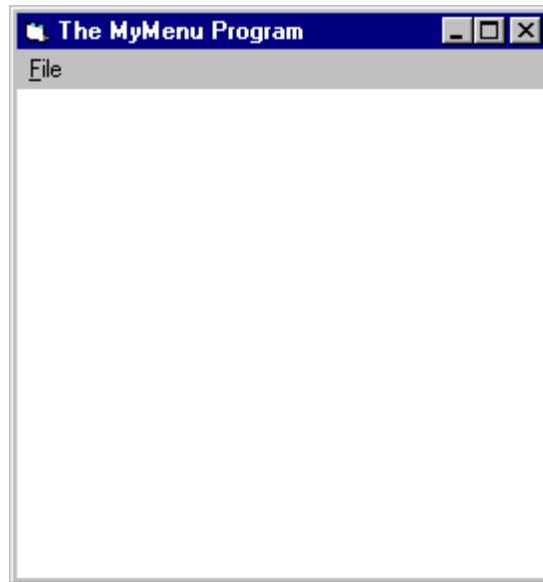



Figure 5.7. *The MyMenu program with its File menu.*

 Select the **File** menu.

The menu of the **File** menu opens as shown in Figure 5.8. As you can see, the **File** menu includes the **Exit** menu item.

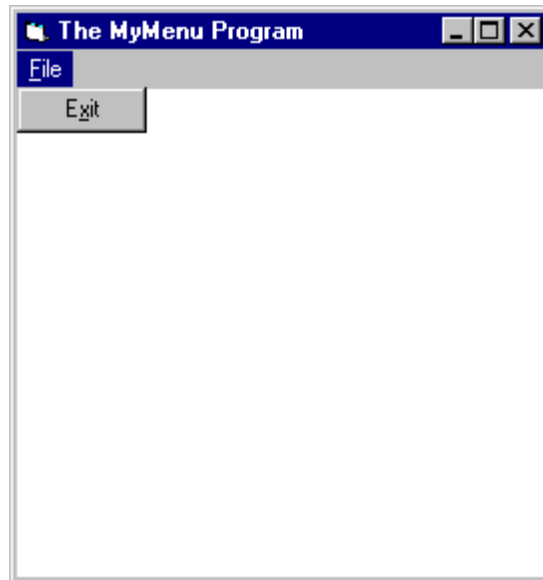




Figure 5.8. *The MyMenu program with its File menu open. The Exit menu item of the File menu is shown.*

Of course, at this point in the development of the MyMenu program, the **Exit** menu item is not operational yet, because you did not yet attach code to the **Exit** menu item.

 Terminate the MyMenu program (for example, select **End** from the **Run** menu of Visual Basic).

Attaching Code to the Exit Menu Item

You'll now attach code to the **Exit** menu item. Your objective is to attach code to the **Exit** menu item, so that when the user selects **Exit** from the **File** menu, the MyMenu program terminates.

 While in design mode (that is, while the MyMenu program is not running), click the **File** menu.

The **Exit** menu item appears as shown in Figure 5.9.

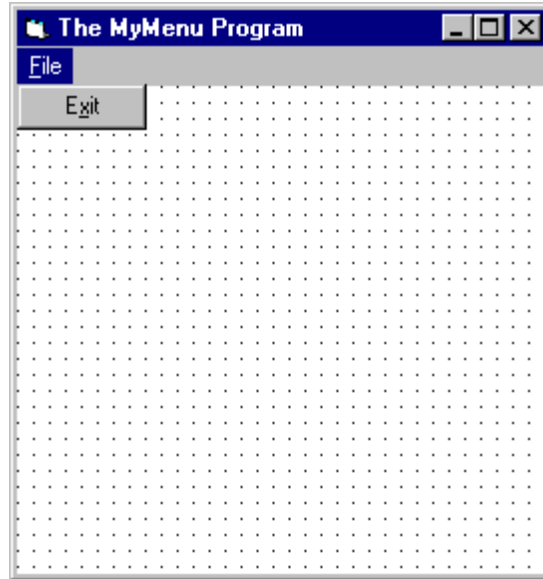


Figure 5.9. *The Exit menu item in design mode.*

 Click the **Exit** menu item.

Visual Basic responds by displaying the Code window with the `mnuExit_Click()` procedure ready to be edited by you.

Here is the code that Visual Basic wrote for you already:

```
Private Sub mnuExit_Click()
```

```
End Sub
```

Recall that you set the **Name** edit box of the **Exit** menu item to **mnuExit** (see Figure 5.6). The Code window now has the **Object** list box set to **mnuExit**, and the **Proc** list box set to **Click**. This means that the `mnuExit_Click()` procedure is automatically executed whenever the user clicks the **Exit** menu item.

 Type the following code inside the `mnuExit_Click()` procedure:

```
Private Sub mnuExit_Click()
```

End


End Sub

So whenever the user clicks the **Exit** menu item, the MyMenu program terminates.

Save your work:

 Select **Save Project** from the **File** menu.

Execute the MyMenu program:

 Select **Start** from the **Run** menu.

 Select **Exit** from the **File** menu.

The MyMenu program responds by terminating itself.

Adding the Options Menu

You'll now add another menu to the MyMenu program, the **Options** menu.



NOTE

The **Tools** menu of Visual Basic displays the **Menu Editor** item, **provided** that the form is the selected window. If for example you display the **Tools** menu while the Code window is the selected window, the **Menu Editor** item inside the **Tools** menu is grayed (unavailable).

 Make sure that the frmMyMenu window is the selected window, and then select **Menu Editor** from the **Tools** menu of Visual Basic.

Visual Basic responds by displaying the Menu Editor dialog box.

 Add the **&Options** item as shown in Figure 5.10.

That is, make sure the Exit item is highlighted, and then click the **Next** button.

Visual basic responds by displaying an empty line, ready to be edited by you.

Set the **Caption** edit box to **&Options**, and set the **Name** edit box to **mnuOptions**.

Also, because the last item that you entered is **Exit** and **Exit** is one level to the right, the **Options** item is also one level to the right. You do **not** want **Options** to be one level to the right. You want **Options** to be on the same level as the **File** item. So make sure to click the button that has a picture of an arrow pointing to the left. This will move the **Options** item to the left (to the same level as the **File** item).

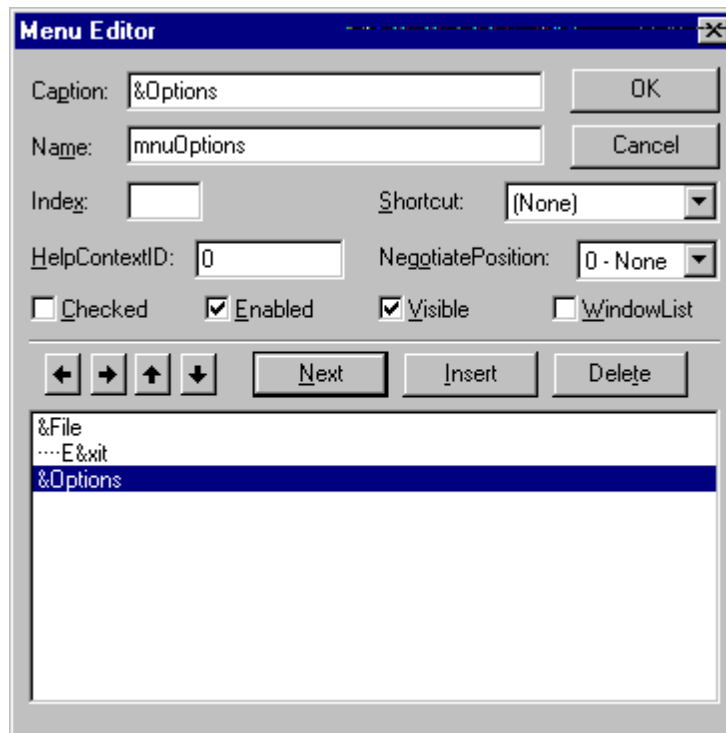



Figure 5.10. *The Options menu item.*

 Add the **&Red** item as shown in Figure 5.11.

That is, make sure that the **Options** item is highlighted, and then click the **Next** button.

Visual Basic responds by displaying an empty line ready to be edited by you.

Set the **Caption** to: **&Red**

Set the **Name** to: **mnuRed**

Click the button that has a picture of an arrow pointing to the right (because you want the **Red** menu item to be inside the **Options** menu).

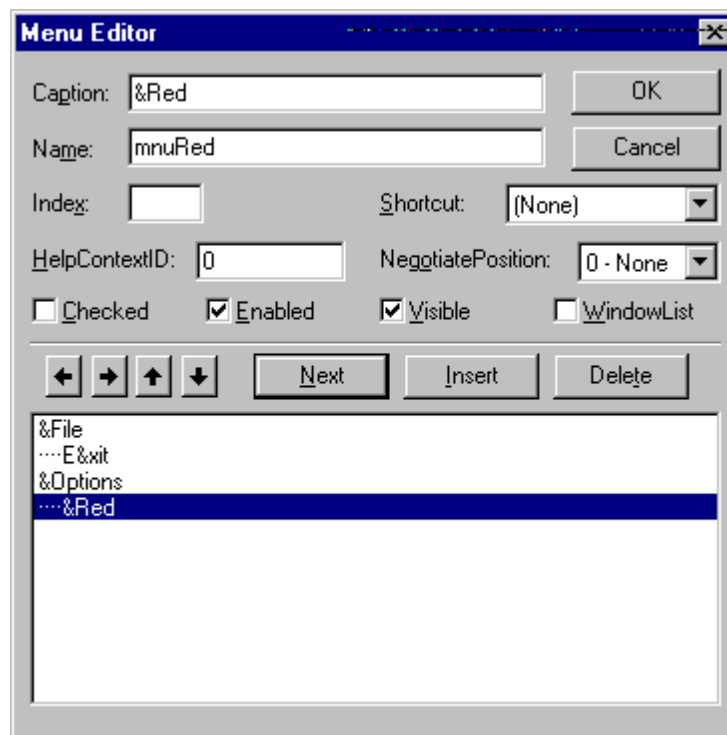



Figure 5.11. *The Red item was added to the Options menu.*

 In a similar manner, add the **&Black** and **&White** menu items as shown in Figure 5.12.

Your Menu Editor dialog box should now look as shown in Figure 5.12.

The **Caption** of the Black menu item is: **&Black**

The **Name** of the Black menu item is: **mnuBlack**

The **Caption** of the White menu item is: **&White**

The **Name** of the White menu item is: **mnuWhite**

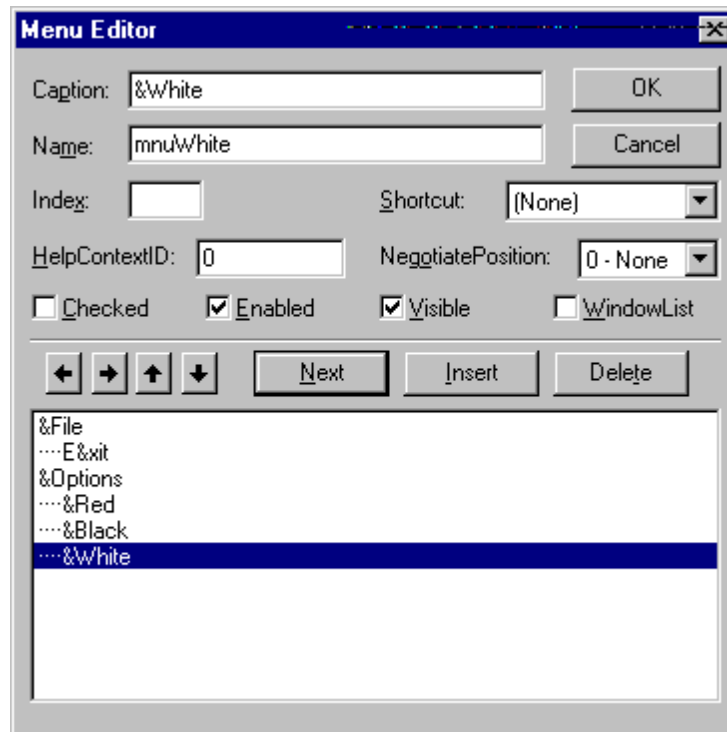




Figure 5.12. *The Options menu with three menu items in it.*

 Click the **OK** button of the Menu Editor dialog box.

Save your work:

 Select **Save Project** from the **File** menu of Visual Basic.

Let's execute the MyMenu program:

 Select **Start** from the **Run** menu of Visual Basic.

The window of MyMenu appears as shown in Figure 5.13. As you can see, the window has the **File** menu and the **Options** menu.

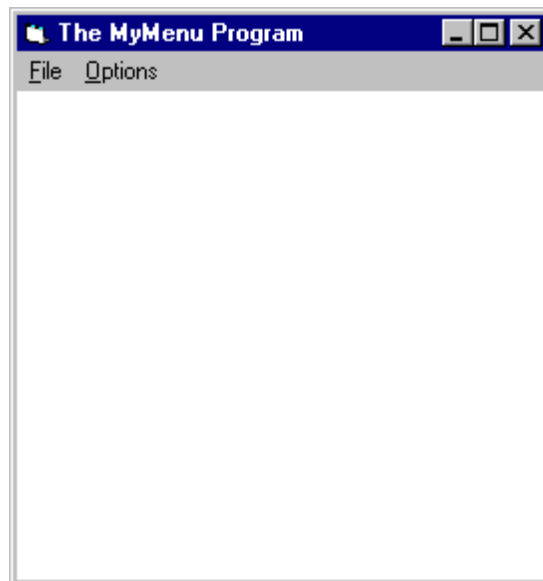


Figure 5.13. *The window of the MyMenu program with its File and Options menus.*

 Select the **Options** menu.

The MyMenu program responds by opening the **Options** menu with the Red, Black and White items in it as shown in Figure 5.14.

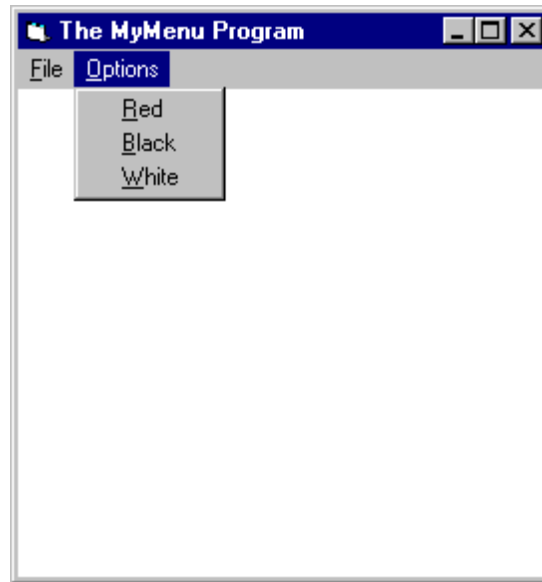



Figure 5.14. *The window of the MyMenu program with its Options menu open.*

Of course, as of yet, the Red, Black, and White menu items are not operational, because you did not yet attach code to these menu items.

 Select **Exit** from the File menu.

MyMenu responds by terminating itself.

Attaching Code to the Red Menu Item

You'll now attach code to the Red menu item.


 Click the Options menu, and then click the Red menu item.

Visual Basic responds by displaying the `mnuRed_Click()` procedure ready to be edited by you as follows:

Private Sub mnuRed_Click()

End Sub

The `mnuRed_Click()` procedure is automatically executed whenever the user clicks the Red menu item of the Options menu.

 Attach the following code to the `mnuRed_Click()` procedure:

Private Sub `mnuRed_Click()`

```
Me.BackColor = QBColor(12)
```


End Sub

The code that you types sets the **BackColor** property of the **frmMyMenu** form to `QBColor(12)`.

`QBColor(12)` represents the red color. So you set the background of **Me** to red.

Note that **Me** is the form in which code is executed. The code is executed from the **frmMyMenu** form. So in this case, **Me** is the same as **frmMyMenu**.

Save your work:

 Select **Save Project** from the **File** menu of Visual Basic.


Attaching Code to the Black Menu Item

You'll now attach code to the Black menu item.

 Click the Options menu, and then click the Black menu item.

Visual Basic responds by displaying the `mnuBlack_Click()` procedure ready to be edited by you.

The `mnuBlack_Click()` procedure is automatically executed whenever the user clicks the Black menu item of the Options menu.

 Attach the following code to the `mnuBlack_Click()` procedure:

Private Sub `mnuBlack_Click()`


```
Me.BackColor = QBColor(0)
```

End Sub

The code that you types sets the **BackColor** property of the **frmMyMenu** form to **QBColor(0)**.

`QBColor(0)` represents the black color. So you set the background of **Me** (which is the `frmMyMenu` form) to black whenever the user selects Black from the Options menu.

Save your work:

 Select Save Project from the File menu of Visual Basic.

Attaching Code to the White Menu Item


You'll now attach code to the White menu item.

 Click the Options menu, and then click the White menu item.

Visual Basic responds by displaying the `mnuWhite_Click()` procedure ready to be edited by you.

=====

The `mnuWhite_Click()` procedure is automatically executed whenever the user clicks the White menu item of the Options menu.

 Attach the following code to the `mnuWhite_Click()` procedure:

Private Sub mnuWhite_Click()


```
Me.BackColor = QBColor(15)
```

End Sub

The code that you types sets the **BackColor** property of the **frmMyMenu** form to **QBColor(15)**.

`QBColor(15)` represents the white color. So you set the background of **Me** (which is the `frmMyMenu` form) to white whenever the user selects White from the Options menu.

Save your work:

 Select **Save Project** from the **File** menu of Visual Basic.


Seeing the Options Menu in Action

Ok, you are now ready to see the Options menu that you implemented in action:

 Select **Start** from the **Run** menu.

 Select **Red** from the **Options** menu.

MyMenu responds by displaying its window with red as shown in Figure 5.15.

 Select **Black** from the **Options** menu.

MyMenu responds by displaying its window with black as shown in Figure 5.16.

 Select **White** from the **Options** menu.

MyMenu responds by displaying its window with white as shown in Figure 5.17.

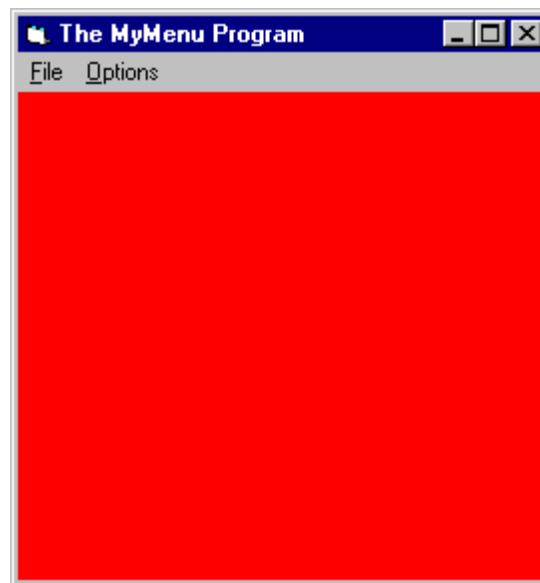


Figure 5.15. *The window of the MyMenu program after selecting Red from the Options menu.*

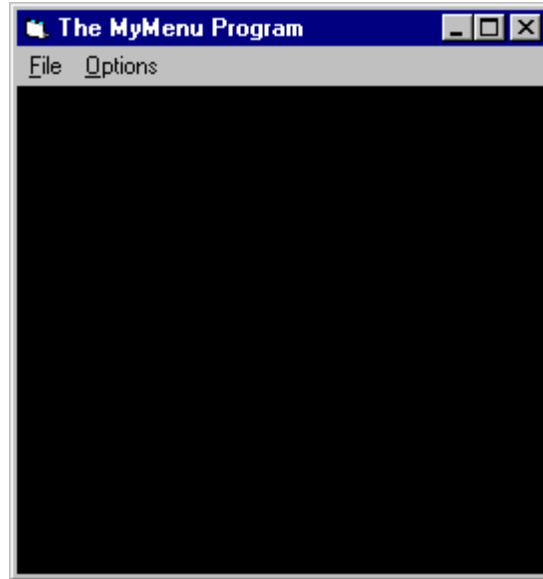


Figure 5.16. *The window of the MyMenu program after selecting Black from the Options menu.*

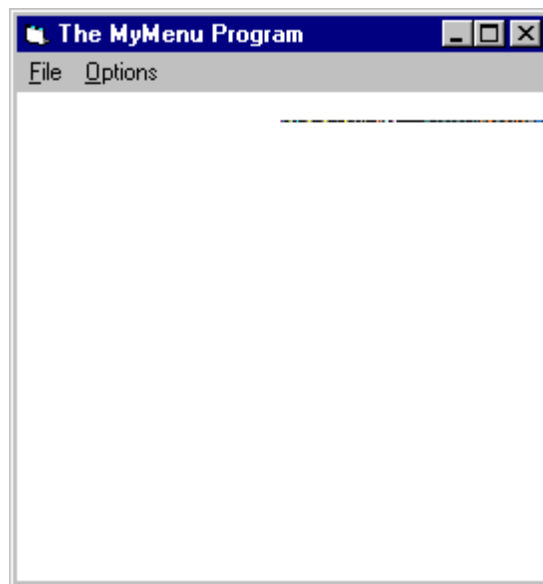



Figure 5.17. *The window of the MyMenu program after selecting White from the Options menu.*

 Experiment with the MyMenu program, and then select Exit from the File menu to terminate the program.

What You Accomplished in This Lesson



You completed Lesson 5 of the Self Study Visual Basic tutorial.

The MyMenu program that you implemented in this lesson is the one that introduces you to the subject of implementing menus in Visual Basic.

Frequently Asked Questions



Q1. The Names of the menu items that I used in this lesson are **mnuFile**, **mnuExit**, **mnuOptions**, and so on. In other words, the names start with the characters **mnu**. Must I start a name of a menu item with the characters **mnu**?

A1. No. You can use any name you wish. However, your program will be easier to read and understand if you start the name of every menu item with the characters **mnu**. This way, when you see a name of an object such as **mnuExit**, you can immediately tell that this object is a menu item.

Q2. I would like to attach a check mark to the Red, Black, and White menu items. That is, when the Red menu item is selected, I would like to see a check mark next to this menu item. When the Black menu item is selected, I would like to see a check mark next to the Black menu item, and when the White menu item is selected, I would like to see a check mark next to the White menu item. Is it possible? I saw this feature implemented in other Windows programs, so I should be able to implement this feature in the MyMenu program, right?

A2. Yes. The programs that you implemented with Visual Basic in this tutorial are Windows programs. As such, all the "standard" features that you see in other Windows programs can be implemented with Visual Basic. In fact, in the Project section of this lesson you'll enhance the MyMenu program, and you'll implement this feature.

Q3. I set the **Caption** of the **File** menu to **&File**, so that **File** appears with its **F** underlined. Why?

=====

A3. During the execution of your MyMenu program, the user can press Alt+F on the keyboard, and this will open the **File** menu. Some users like to use the keyboard instead of the mouse. So be nice and accommodate these users.

In a similar way, you set the **Caption** of the **Option** menu to **Option** (with **O** underlined), so the user can press Alt+O and the **Option** menu opens.

Q4. I set the **Caption** of the **Exit** menu to **E&xit**, so that **Exit** appears with its **x** underlined. Why?

A4. During the execution of your MyMenu program, the user can open the **File** menu and he/she will see the **Exit** item in the File menu. Now the user can click the **Exit** menu item, or press **x** to exit the program.

In a similar way, you set the **Caption** of the **Red** menu to **Red** (with **R** underlined), so the user can open the Options menu, and then press **R** to select the red menu item.

Exam



1. To create a menu with Visual Basic you:

(a) Use the Menu Editor (select Menu Editor from the Tools menu of Visual Basic).

(b) Unfortunately, it is impossible to implement menus in Visual Basic.

2. The **Me** keyword represents:

(a) The person who uses the program.

(b) The person who develops the program.

(c) The form in which code is executed.

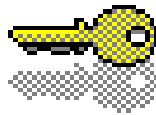
3. The **Me** keyword represents the form in which code is executed. So why not use the following code:

```
frmMyMenu.BackColor = QBColor(12)
```

Instead of the following code:

```
Me.BackColor = QBColor(12)
```

Answers to Exam



1. (a) Use the Menu Editor (select Menu Editor from the Tools menu of Visual Basic).

2. (c) The **Me** keyword represents the form in which code is executed. In the `mnuRed_Click()` procedure of the MyMenu program you used the **Me** keyword as follows:

Private Sub mnuRed_Click()

```
Me.BackColor = QBColor(12)
```

End Sub

The preceding code means that the **BackColor** property of **Me** is set to red. **Me** represents the form in which code is executed. The code is inside the `frmMyMenu` form. So putting it altogether, the **BackColor** property of `frmMyMenu` is set to red.

3. Yes, you can use the statement:

```
frmMyMenu.BackColor = QBColor(12)
```

And you can use the statement:

```
Me.BackColor = QBColor(12)
```

In both cases, the results are the same.

So which statement is better? Try to use the **Me** keyword whenever possible. Why? Because when developing Visual Basic programs, you usually borrow code from previous programs that you developed. For example, you now developed the MyMenu program. In the future you'll develop another program. You may remember the MyMenu program, and you will copy and paste code from the MyMenu program to your new program. So for example, you may copy the following statement:

```
frmMyMenu.BackColor = QBColor(12)
```

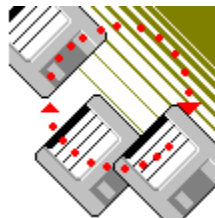
However, in your new program, there is no such form as frmMyMenu. So when you execute your new program, you'll get an error, telling you that frmMyMenu is unknown.

If in your new program you are using a statement such as:

```
Me.BackColor = QBColor(12)
```


The preceding statement will be valid in your new program.

Project




You'll now enhance the MyMenu program, so that a check mark is placed next to the menu item of the Options menu whenever the menu item is selected. For example, once the user selects the Black menu items, a check mark appears next to the Black menu item. So if now the user opens the Options menu again, he/she will see a check mark next to the

Black menu item. Similarly, a check mark will be placed inside the Red and White menu items when these are selected.

 Make the frmMyMenu form the selected window, and then select **Menu Editor** from the **Tools** menu.

Visual Basic responds by displaying the Menu Editor dialog box.

 Make sure that the White item is highlighted, and then place a check mark inside the **Checked** check box.

The Menu Editor dialog box should now look as shown in Figure 5.18.

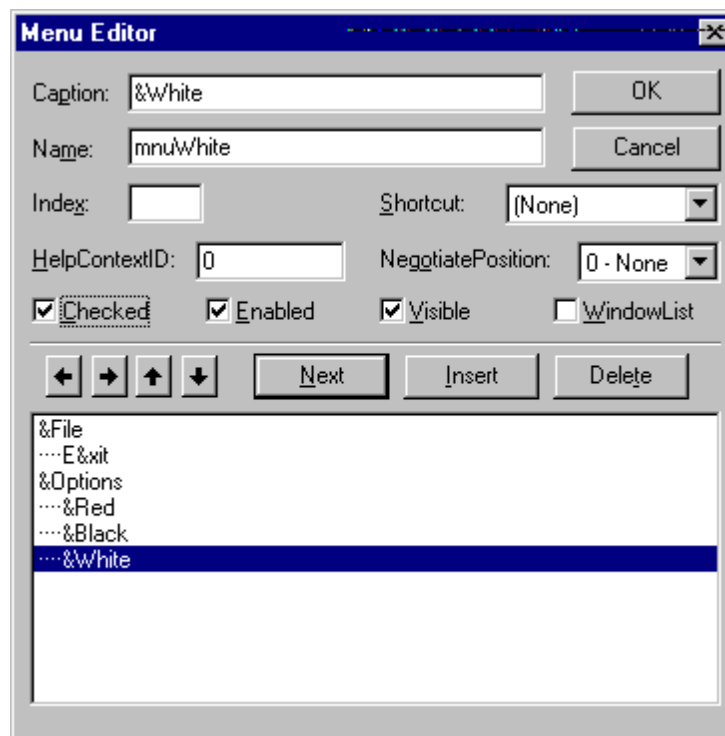




Figure 5.18.: *Placing a check mark inside the Checked check box of the White menu item*

 Click the **OK** button of the Menu Editor dialog box.

Save your work:

 Select **Save Project** from the **File** menu.

Let's see what you have accomplished:

 Select **Start** from the **Run** menu.

The MyMenu program is executed.

 Select the **Options** menu.

The **Options** menu appears as shown in Figure 5.19.

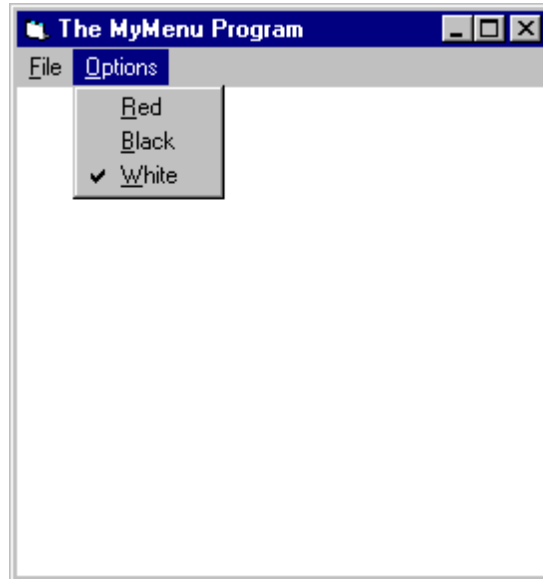



Figure 5.19.: A check mark appears next to the White menu item.

The point to know at this point is that you can now select other menu items from the **Options** menu, but the check mark remains on the **White** item. Why? Because you did not yet write any code that places the check mark on the selected menu item. You only placed a check mark next to the **White** menu item during design time.


 Select **Exit** from the **File** menu to terminate the MyMenu program.

Placing a Menu Check Mark on the Appropriate Menu Item

Now you'll write code that places the check mark on the appropriate menu item (Red, Black, or White).

 Click the Options menu, and then click the Red menu item.

Visual Basic responds by displaying the mnuRed_Click() procedure ready to be edited by you.

 Add code to the mnuRed_Click() procedure so that it will look as follows:

Private Sub mnuRed_Click()

```
Me.BackColor = QBColor(12)
mnuRed.Checked = True
mnuBlack.Checked = False
mnuWhite.Checked = False
```

End Sub

The code that you added to the mnuRed_Click() procedure sets the **Checked** property of the mnuRed menu item to **True**. The **Checked** property of the mnuBlack and mnuWhite menu items are set to **False**.

Putting it all together, a check mark is placed inside the Red menu item, and no check marks are displayed inside the Black and White menu items.

 Add code to the mnuBlack_Click() procedure so that it will look as follows:

Private Sub mnuBlack_Click()

```
Me.BackColor = QBColor(0)
mnuRed.Checked = False
mnuBlack.Checked = True
mnuWhite.Checked = False
```

End Sub

The code that you typed places a check mark inside the Black menu item, and this code removes check marks from the Red and White menu items.

 Add code to the mnuWhite_Click() procedure is that it will look as follows:


Private Sub mnuWhite_Click()

```
Me.BackColor = QBColor(15)
mnuRed.Checked = False
mnuBlack.Checked = False
mnuWhite.Checked = True
```

End Sub

The code that you typed places a check mark inside the White menu item, and this code removes check marks from the Red and Black menu items.

Save your work:

 Select **Save Project** from the **File** menu of Visual Basic.

Let's see your code in action:

 Select **Start** from the **Run** menu of Visual Basic.

The window of the MyMenu program appears.

 Select Red from the Options menu.

The MyMenu program responds by changing the background color of its window to red.

 Select the Options menu.

The Options menu opens. As you can see (see Figure 5.20), the Red menu item has a check mark next to it.

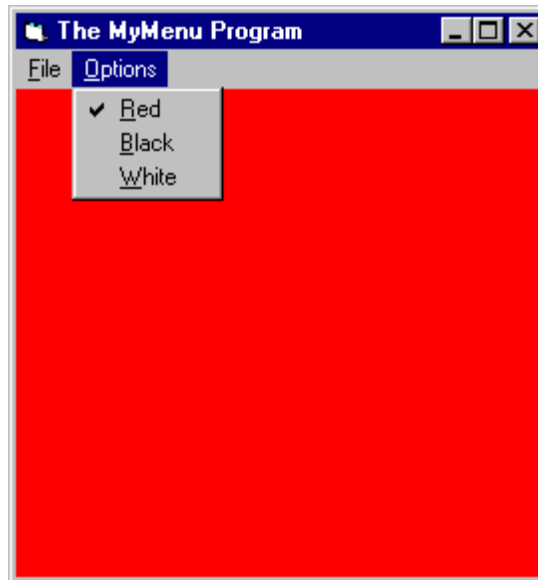



Figure 5.20. A check mark appears next to the Red menu item.


 Select Black from the Options menu.

The MyMenu program responds by changing the background color of its window to black.

 Select the Options menu.

The Options menu opens. As you can see, the Black menu item has a check mark next to it.

 Experiment with the MyMenu program. In particular, verify that the last menu item that you selected has a check mark next to it.

 Select Exit from the File menu to terminate the MyMenu program..

The check mark next to the menu items is typically used when you want your program to tell your user that a certain setting was selected. From example, in the MyMenu program., the user can open the Options menu, and see that either the Red, Black, or the White color was the last selected color.

How To Contact TegoSoft



You can contact TegoSoft Inc. by any one of the following methods:

- Use TegoSoft Internet Web site:
<http://www.tegosoft.com>
- Send TegoSoft an e-mail:
tegosoft@msn.com
- Send TegoSoft a letter:
TegoSoft Inc.
P.O.Box 389
Bellmore, NY 11710
USA

Technical Support



If you have a technical question, you can post the question to the TegoSoft Technical Support staff, and they will try to answer your question.

The **best** way to post a technical question is by sending TegoSoft an e-mail.

- The e-mail of TegoSoft is:
tegosoft@msn.com

When sending TegoSoft an e-mail with a technical question, please follow the following format:

Date: _____
Your name: _____
Company (if applicable): _____
Your phone number: _____
Your e-mail: _____
Country (if not USA): _____
State (if inside USA): _____

Operating System used: _____
Programming language and version : _____



My technical question is:

Copyright © and Notices

Copyright © 1994, 1995, 1996 by TegoSoft Inc. ® All Rights Reserved

TegoSoft Self Study Tutorials & Software

Copyright ©1994, 1995, 1996 by TegoSoft Inc. ® All Rights Reserved

Although precaution was taken in the preparation of this document, TegoSoft assumes no responsibility for errors or omissions. TegoSoft is not liable for damages resulting from the use of the information contained in this document. Use this document at your own risk. This document is copyright protected. This means that you should treat this document like any other copyright material. No part of this document should be copied in any way. You are not allowed to use this document or any part of this document for any purpose other than read it. You are not allowed to publish this document or any part of this document in any way. You are not allowed to sell this document or any part of this document, you are not allowed to incorporate this document or any part of this document in any book, magazine, Web Site, Electronic forums, disks, CDs, or any other media. The only thing that you are allowed to do with this document is read it for the sole purpose of studying the material that is presented in this document. If this document includes software, the software must be treated in the exact same way that this document is treated. You are not allowed to distribute the software in any way. The sole purpose of supplying the accompanying software is to enable you to experience with it. No part of this document should be modified. If accompanying software is included with this document, you are not allowed to modify the software.

Rev. 031796-1