Training Manual

FOR

Microsoft Visual Basic Programming

Compiled by **Technology Ed**

April 2012

CHAPTER-5: FORMS

This chapter describes the Forms in Visual Basic programming. For additional information on Forms, refer to Section 2.2, **Error! Reference source not found.**.

5.1 Using Forms

Prior to drawing any controls on a Form, you must perform the following:

Name the form and save it:

A Form has a name (which is how you refer to it in code) and a file name with an .frm extension. An .frm file is actually a straight text file which contains instructions that VB uses to create the Form. Load an .frm file into any text editor if you want to see these instructions. I usually right click on the Form in Project Explorer and select "Save Form1 As", give the file a name and save it, and then immediately go to the Properties Window and give the Form a descriptive name with an "frm" prefix.

Decide what kind of border to use:

By default, your Form will have the familiar blue title bar on top, a "Form" icon, a "Form1" title, minimize, maximize, and close buttons, (referred to as the Control Box), and be sizable at run time. Any of this can be changed while designing the Form, but only the title (Caption property) and the icon (Icon property) can be changed at run time.

BorderStyle property:

- 0 - None No border, title, icon, or control box. The form will not be sizable, or able to be moved by the user. You are responsible for writing code to close the Form at run time.
- 1 - Fixed Single - same as a default form, but the user cannot resize it
- 2 – Sizable- default
- 3 - Fixed Dialog - no minimize or maximize buttons, no Icon, and not sizable by the user. This setting is used for specialized dialog boxes
- 4 - Fixed ToolWindow A Tool Window has a slimmer title bar, and no minimize or maximize buttons. Use this setting as a utility form, such as a toolbox.
- 5 - Sizable ToolWindow Same as above but able to be resized by the user.

MinButton, MaxButton, and ControlBox properties:

These three properties can be set to True or False. Setting ControlBox to False will remove all three buttons. Setting MinButton or MaxButton to False will not totally remove the button, but rather just gray it out so it cannot be used. Setting both MinButton and MaxButton to False will remove both buttons. You cannot gray out the close button (the 'x') without some help from API calls and you can only remove it by removing the entire Control Box.

WindowState property:

Set this property at design time to indicate how the Form will be displayed initially.

- 0 - Normal The Form will display according to the size and location you decided at design time. In most cases, you'll use this setting which is also the default.
- 1 - Minimized The Form will start up minimized as an icon in the Windows Taskbar.
- 2 - Maximized The Form will run full screen, but this may not be a good idea if you're eventually going to run your program on different machines, due to the different screen sizes. You might think your controls are nicely placed until you run the Form on a different computer with a larger screen size and find a lot of empty space or on a smaller screen size and the whole Form doesn't seem to be there.

StartUpPosition property:

If your Form is not maximized, you'll need to place it in an appropriate place on the screen. You can do this with the Form Layout Window or set the StartUpPosition property.

- 0 - Manual You'll set the Left and Top properties of the Form in the Form's Load event.
- 1 - CenterOwner The form will be centered on the form which owns it.
- 2 - CenterScreen The Form will run centered on the screen. A most pleasing effect in my opinion.
- 3 - Windows Default This is the default setting. You're putting the decision into the hands of Windows, which can be unpredictable.

Moveable property:

By default, the Moveable property is set to True, meaning users will be able to move your Forms by dragging on the title bar. Set this property to False to create a non-moveable Form. This won't affect whether or not the Form can resized.

Caption property:

This is the title of your Form. It will appear on the left side of the title bar next to the Icon. You can set this property dynamically, (which simply means at run time), if your Form has multiple purposes and needs different titles at different times.

Icon property:

Set this property to an icon picture file if you don't like the "Form" icon. Actually, using the default form icon is somewhat non-professional. The icon can be changed at run time by setting the property with the LoadPicture function or setting it to a picture which is already a property of another Form or Control with a Picture property. The standard Windows icon is also available by going to a Form's Properties Window, highlighting the (Icon) text and pressing the delete key.

Picture property:

Set this property to any valid picture file to set a background for your Form. The picture file should be of the correct size, you cannot stretch, shrink, center, or tile it without some help from Bit Block Transfer See Image control for a list of valid picture file formats..

5.2 Multiple Forms

Many projects will use more than one form. When you have more than one module (form or standard) in a project, you must specify a "Startup object", which is done via the Project menu, Properties item. The startup object tells VB which form or standard module is to have its code run first (if a standard module is the startup object, it must contain a public subroutine called "Main", and "Sub Main" is specified for the Startup object). By default, the startup object is the initial form that is supplied with every new VB Project.

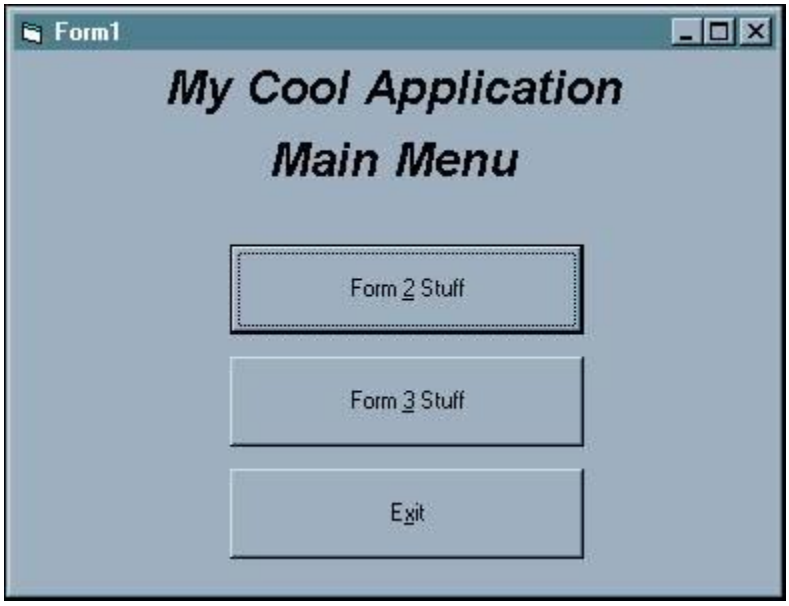
Here are a couple of scenarios for projects with multiple forms:

You have an application that performs a variety of functions, requiring multiple "screens" (forms). You can use one form as a "switchboard" or "main menu" form that connects to the other forms.

You may wish to have a "splash screen" start off your application before the main form is displayed.

Using Multiple Forms: Example

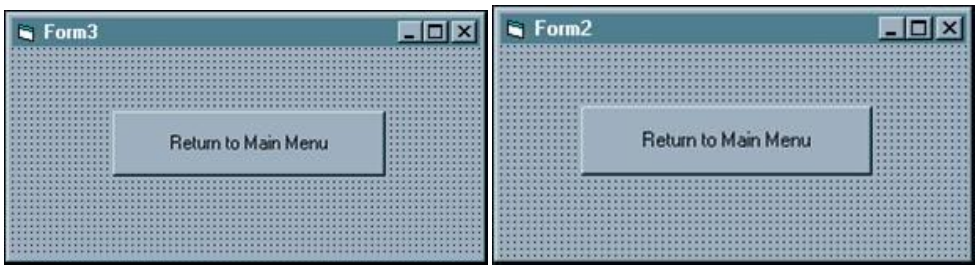
Assume you have a project with three forms: Form1, Form2, and Form3. (To add forms to a project, go to the Project menu and select Add Form.) Form1 serves as the switchboard form, which contains three command buttons: cmdForm2, cmdForm3, and cmdExit:



The code behind these buttons is:

```
Private Sub cmdForm2_Click()

Form2.Show vbModal

End Sub

Private Sub cmdForm3_Click()

Form3.Show vbModal

End Sub

Private Sub cmdExit_Click()
```

```
Unload Me

End Sub
```



For the sake of the example, Form2 and Form3 simply contain one button with the caption "Return to Main Menu". The code behind the command button on each of these forms is simply Unload Me. When either of these forms is unloaded, the main menu form will then become the active form:

5.3 Order of Events

In Visual Basic, a single action, such as moving from one control on an object to another control, can trigger several different events, which occur in a particular sequence. Knowing when events occur and in what order they occur is important, because it can affect how and when your event procedures run. For example, if there are two event procedures that are to be run in a certain order, you want to make sure that the events that they are associated with occur in that same order.

For example, the following sequence of events occurs when a form is loaded in Visual Basic:

- Form_Initialize
- Form_Load
- Form_Resize
- Form_Activate
- Form_GotFocus
- Form_Paint
- Form_QueryUnload
- Form_Unload
- Form_Terminate