Chapter Four: Projects

4.1 Projects

An application in Visual Basic is created as a 'project'. **A project** is a collection of files which are dependent on each other. A project will normally consist of:

- One file for each form module (extension .frm).
- One file for each standard (base) module of code (extension .bas)

Event Procedures: - Code related to some object. This is the code that is executed when a certain event occurs.

General Procedures: - Code not related to objects. This code must be invoked by the application.

Standard or Base Modules: - Collection of **general procedures**, variable **declarations,** and **constant definitions** used by application.

Form Modules: - Collection of **Event procedures**.

Note:-

Global variables cannot be defined in general section of the form module ,because they are allocated in memory only when the form is loaded. If a form is unloaded, they lose their value. So to make variables visible to all forms, place them in the Definitions block (general section) of the **base (standard) code module**, as mentioned in the previous section.

To Add Base (standard) module to the project do the following

1- click the project menu

2- Select the Add Module. See figure 1 below

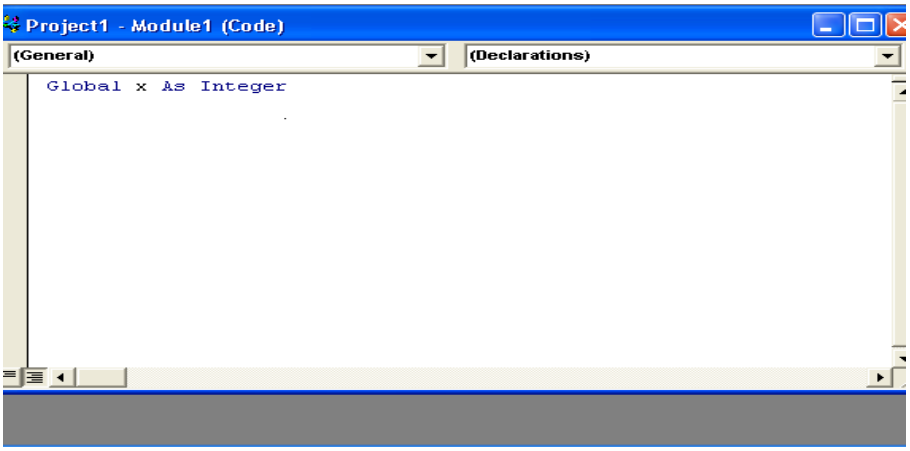


Figure 1 (Base Module)

Then we declare variable **X** as **Global** in general section of base **module code**, so the variable **X** can be accessed by all forms inside the applications. In our project, the project contain **two forms** and **one base module** see figure 2 below.

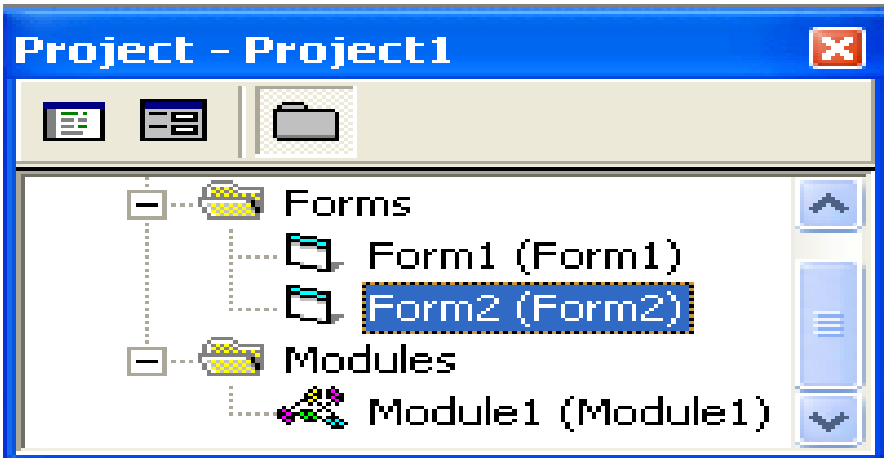
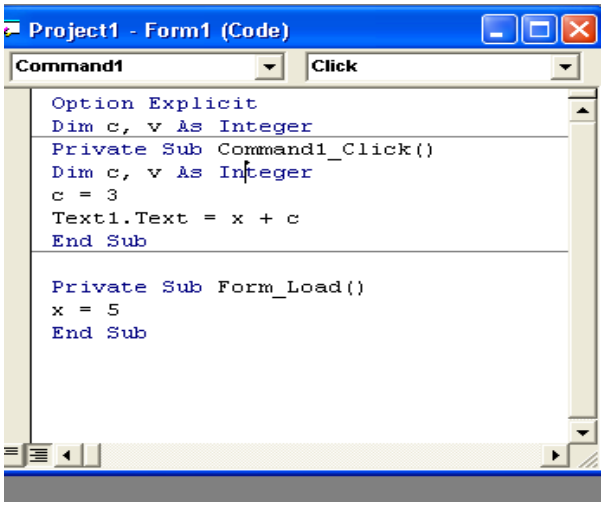
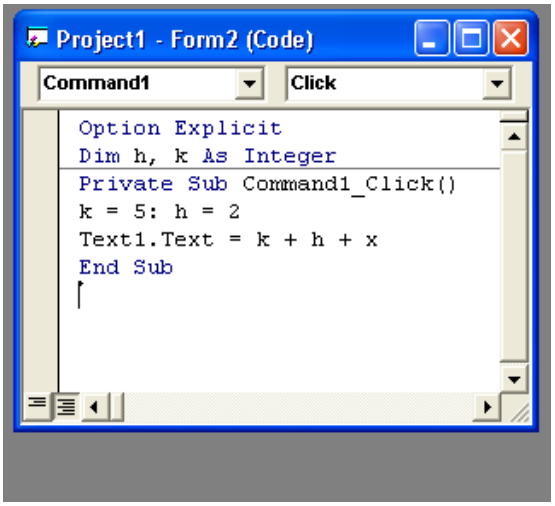


Figure 2

So the two forms can be accessed to the variable **X**, as shown below



Form1



Form2

When we begin with **form1**, the value of **X** is **5** and the result of **text1.text** is **8** and don't need to declare **X** in **form1**, then when execute the **form2** the value of variable **X** remain **5** and the value of **text1.text** in the **form2** is **12** and also don't need to declare **X** in **form2** because the variable **X** is declare as **global in general section of base module code**

4.2 Saving Visual Basic Applications:

There are four save commands available under the File menu in Visual Basic:

Save [Form Name]	Save the currently selected form or module with the current name.
Save [Form Name] As	Like Save File, however you have the option to change the file name
Save Project	Saves all forms and modules in the current project using their current names
Save Project As	Like Save Project, however you have the option to change file names.

4.3 Object Event

1-Form Events:

Event	Description
Click	Form_ Click event is triggered when user clicks on form.
DblClick	Form_ DblClick event is triggered when user double- clicks on form.
Load	Form _Load event occurs when form is loaded. This is a good placeto initialize variables and set any run-time properties.

2-Command Button Events:

Event	Description
Click Event	Triggered when button is selected either by clicking on it or by pressing the access key.

3-Label Events:

Event	Description
Click Event	Triggered when user clicks on a label.
DblClick Event	Triggered when user double-clicks on a label

Object Method

In previous work, we have seen that each object has properties and events associated with it. A third concept associated with objects is the method. A method is Built-in procedure or function that imparts some action to an object.

Methods are always enacted at run-time in code. The format for invoking a method is: ObjectName.Method {optional arguments}

1-Form Methods:

Method	Description
Cls	Clears all graphics and text from form. Does not clear any objects.
Print	Prints text string on the form.
Hide	Hide the form
Show	Show the form

Examples

1-frmExample.Cls	' clears the form
2-frmExample.Print "This will print on the form"	‘print the string on the form
3-frmExample.Hide	‘ hide the form called frmExample
4-frmExample.Show	‘ show the form called frmExample

2-Text Box Methods:

Method	Description
SetFocus	Places the cursor in a specified text box.

Example

txtExample.SetFocus	' moves cursor to box named txtExample

VB Functions

The main purpose of the functions is to accept certain inputs and pass them on to the main program to finish the execution. They are two types of function, the **built-in functions (or internal functions)** and the **functions created by the programmers.**

The general format of a function is

functionName(arguments)

Where arguments are values that are passed on to the functions.

In this lesson, we are going to learn two very basic useful internal functions i.e. the **MsgBox( )** and **InputBox ( ) functions** and **the common built in function** .

1-A MsgBox() and InputBox()

You use input boxes and message boxes when you need to ask the user questions or display error messages and advice to the user.

A **message box**: - is a dialog box you display to give the user information.

An **input box**: - is a dialog box you display to ask the user questions.

MsgBox()

1-MsgBox Return value

The objective of MsgBox is to produce a pop-up message box and ask the user to click on a command button before he /she can continues. This message box format is as follows:

yourMsg=MsgBox(Prompt, Style Value, Title)

The first argument, Prompt, will display the message in the message box. The Style Value is an optional numeric value or constant name that will determine what type of command buttons appear on the message box and any icon to show. Table 1 and table2 that refer for types of command button displayed and the type of the icon to display in the message box. The Title argument is an optional string that represents the text in the message box's title bar.

Table.1. The command buttons displayed in a message box.

Style Value	Named Constant	Buttons Displayed
0	vbOkOnly	Ok button
1	vbOkCancel	Ok and Cancel buttons
2	vbAbortRetryIgnore	Abort, Retry and Ignore buttons.
3	vbYesNoCancel	Yes, No and Cancel buttons
4	vbYesNo	Yes and No buttons
5	vbRetryCancel	Retry and Cancel buttons

We can use **named constant** in place of **integers** for the second argument to make the programs more readable. Infact, VB6 will automatically shows up a list of names constant where you can select one of them.

Example:

yourMsg=**MsgBox**(**"Click OK to Proceed"**, **1**, **"Startup Menu"**)

or

yourMsg=**MsgBox** (**"Click OK to Proceed", vbOkCancel**, **"Startup Menu"**)

are the same.

To make the message box looks more sophisticated, you can add an icon besides the message. They are four types of icons available in VB as shown in Table 10.3

Table 2. The icons displayed in a message box.

Value	Named Constant	Icon

16	VbCritical	
32	vbQuestion	
48	Vb Exclamation	
64	vbInformation	

Example:-

The following MsgBox() function below produces the message box shown in Figure 3

yourMsg = **MsgBox ("Click to Test"** , **vbYesNoCancel + vbExclamation** , **"Test Message")**

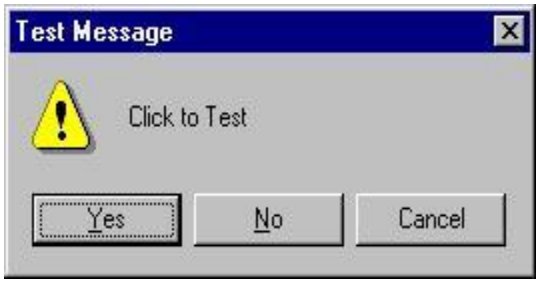


figure3

yourMsg is a variable that holds values that are returned by the **MsgBox ( ) function**. The values are determined by the type of buttons being clicked by the users. It has to be declared as Integer data type in the procedure or in the general declaration section. Table 3 shows the values, the corresponding named constant and buttons.

Table 3: MsgBox() return values.

Value	Named Constant	Button Clicked
1	vbOk	Ok button
2	vbCancel	Cancel button
3	vbAbort	Abort button
4	vbRetry	Retry button
5	vbIgnore	Ignore button
6	vbYes	Yes button
7	vbNo	No button

Example 1:-

1. **The Interface**:

You draw one command button and a label as shown in Figure 4, and set the properties as follows

Command1	
Name	test
Caption	test me
labels	
Name	display
Appearance	0-flat
Caption	Blank

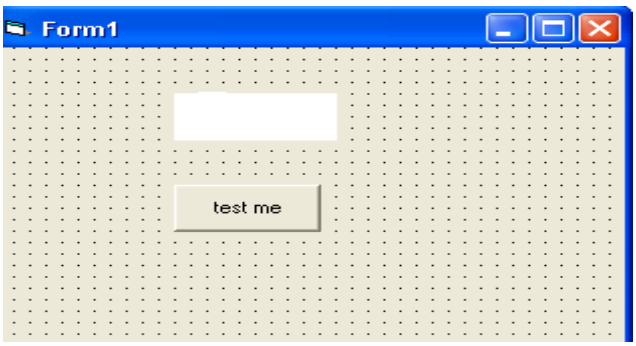


Figure 4

2. The procedure for the test button:

```
Private Sub Test_Click()
Dim testmsg As Integer
testmsg = MsgBox("Click to test", 1, "Test message")
If testmsg = 1 Then
Display.Caption = "Testing Successful"
Else
Display.Caption = "Testing fail"
End If
End Sub
```

When a user click on the test button, the image like the one shown in Figure 5 will appear. As the user click on the OK button, the message "Testing successful" will be displayed and when he/she clicks on the Cancel button, the message "Testing fail" will be displayed.

Figure 5

To make the message box looks more sophisticated, you can add an icon besides the message.

Example 2:-

In this example, the following message box will be displayed

You could draw the same Interface as in example 1 but modify the codes as follows:

```
Private Sub test_Click()
Dim testMsg As Integer
testMsg = MsgBox("Click to Test", vbYesNoCancel + vbExclamation, "Test
Message")
If testMsg = 6 Then
display.Caption = "Testing successful"
ElseIf testMsg = 7 Then
display.Caption = "Are you sure?"
Else
display.Caption = "Testing fail"
End If
End Sub
```

2- MsgBox return No value

The statement form of the message box returns no value (it simply displays the box): MsgBox Message, Type, and Title Where

Message	Text message to be displayed
Type	will determine what type of command buttons appear on the Message box and any icon to show
Title	Text in title bar of message box

In this case don't need to appear command buttons in the message box but leave it empty,,

Example:-

X=10

Msgbox "the value of x = "& x," value of variables "

2-The InputBox( ) Function

An InputBox( ) function will display a message box where the user can enter a value or a message in the form of text. **The format is**

MyMessage=Input Box(Prompt, Title, default_text, x-position, y-position)

myMessage is a variant data type but typically it is declared as string, which accept the message input by the users. The arguments are explained as follows:

- **Prompt**: - The message displayed normally as a question asked.
- **Title:** - The title of the Input Box.
- **Default-text**: - The default text that appears in the input field where users can use it as his intended input or he may change to the message he wish to key in.
- **X-position and y-position:** - the position or the coordinate of the input box.

Example:-

1. **The Interface**

Draw two labels and one command and then set the following properties see figure 6.

Label1	
Caption	your message
Lable2	
Caption	blank
Apperance	0-flat

Command1	
Name	ok
Caption	ok

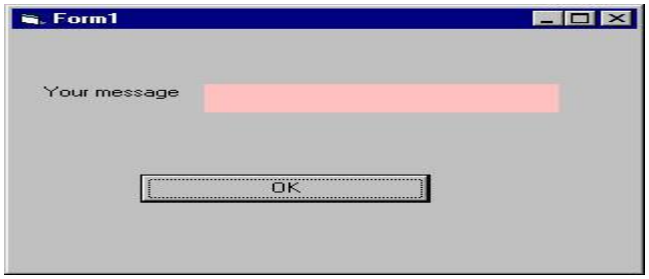


Fig 6

2. The procedure for the OK button

```
Private Sub OK_Click()
Dim userMsg As String
userMsg = InputBox("What is your message?", "Message Entry Form", "Enter your messge here", 500, 700)
If userMsg <> "" Then
message.Caption = userMsg
Else
message.Caption = "No Message"
End If
End Sub
```

When a user click the OK button, the input box as shown in Figure 7 will appear. After user entering the message and click OK, the message will be displayed on the caption, if he click Cancel, "No message" will be displayed.