Laboratory Manual

ON

FRONT END TOOLS LAB.

(For 6th Semester (IT))

Prepared by :

SASMITA PANIGRAHI
PTGF (CSE / IT)
UCP Engineering School
Berhampur

Chapter-1

INTRODUCTION:

Welcome to the world of another Microsoft product "Visual Basic" - extremely powerful, state-of the-art and at the same time easy-to-use computer-programming environment that enables you to develop Windows applications. This environment includes everything one needs to create, modify, test and compile the application. With Visual Basic, one can automate everyday tasks,

add custom required features and functions to suit ones need, and create applications which makes ones task easier. Visual means the way you develop an application's user interface and basic is the type of programming code that you write. In this chapter, you will learn some basic concepts of this powerful language.
As a software developer or as a programmer, you are expected to design and develop every program that works correctly, efficiently and at the same time easy-to-use by any person who may or may not be well versed with the computer and its capabilities. The application programmes written in any language must be such that the user of that programme should find it extremely friendly in the sense that not much effort should be required on the part of the user to understand and use an application. It is the ability of the user interface that you create that someone sees when your program is running.

START AND EXIT VISUAL BASIC

Before trying to start the Visual Basic package on your computer, please ensure that the package has been properly installed and the shortcut for the same has been created. You will be able to see the name of this software in the sub-menu of the program menu. Although it is possible to start the package without the help of the shortcut, but it is easier to activate using the shortcut in case you are a frequent user of this package.
To start Microsoft Visual Basic, click on **Options** in the Program Manager to open its drop down menu. Like every other application in Windows, you click your mouse button when the pointer is pointing at the Visual basic option (as shown in the picture below) Double click the Visual **Basic icon.** As the Visual Basic gets loaded, it will show you Microsoft Visual Basic in design mode.
Visual Basic displays five windows when you start it i.e.
1. Microsoft Visual Basic Window (which consists of a title bar, menu bar, and the tool bar)
2. The Form window
3. Project window
4. Toolbox
5. The Properties window
In this window, like every other application in Microsoft, you will find that it is similar to other window base applications and some of the options listed are quite familiar.
Microsoft Visual Basic Window: The Main Window contains the menu bar and the toolbar.
The Form Window: A form is the window that is displayed when the application is executed. This Window is initially blank. If this is missing then in the project window, click on **Form1** to select it, then click on the **View Form** button..
The Project Window: The Project Window displays the files associated with the project that is currently active. If this is missing then choose **View, Project** to display it.
Property Window: Property Window displays the information about the form. If this is missing then choose **View, Properties** to display it.
Toolbox: Toolbox contains number of controls, which one can use in the creation of the forms. If this is missing then choose **View, Toolbox** to display it.
Like any other application in Microsoft Window, the procedure followed to close your work safely is the same. If you click at the "File" option in the Main Window, you will see the picture as shown below. To exit from Visual Basic, select Exit from the File menu.

VISUAL BASIC INTERFACE

A user interface makes the user to communicate in two ways. Firstly, the user interface makes it easy for the user to tell the computer what to do next. Secondly, a user interface enables the computer to display information on the screen the way it should convey some meaning to the user. You know that a program is defined as a set of instructions given to the computer to make it perform a specific task. To make the computer to carry out a task, you specify the steps one by one to the machine in an

interactive manner. Programming in Visual Basic comprises of instructions that tell Visual Basic, what steps you want it to perform. For displaying anything on your window, Visual Basic gives you some objects. You may modify the properties of objects, already available on the interface. Moment you change the properties of an object, the object takes care of displaying you the information on the screen.
There are number of programming languages, viz. C, Basic, Assembly language etc, that you must have already used or can use to write programs. These languages are like any other language because each one has its own set of rules of grammar and set of words. Each language has its strength and weakness. Many programmers write their program in C language because of its power and flexibility. In comparison Visual Basic, another powerful language, not only encourages you to write short programs, but also makes you to adopt the modular approach in programming that makes the programming interesting and easy to develop any application.

DEBUG WINDOW

It is extremely important to debug your programmes before they can be put to its actual usage. The Visual Basic provides you an easy way of debugging your programmes. A window using which you can keep a track of every instruction of your application while you are designing it, is called the **Debug Window.** The features of debugging tools in Visual Basic can help you locate, identify and correct your coding errors. The moment you run your application in Visual Basic, Debug window automatically opens. This window appears behind Form l. If your application is behaving in a ı manner different from its normal operation, and you are not in a position to find what the error is and where exactly the problem lies, you do not have to get tensed although it is easy to say so.

Debug window shall take care of this and shall help you. You can set up some break points (a kind of "wait" in your application) in an application while designing it. You can put the break points at any place in your application programme but generally the breaks are put where you feel the error is most likely occurring. In break mode, you can examine and make changes in the current values of variables and properties, make minor code modifications, and specify which statement the application will run next. While executing the programme i.e. at run time, you can use it to display data or messages as the programme gets executed. The debug window is a window, which you can use to write even a single line program. You can type a command and immediately see the result.

To open the debug window follow the procedure given below:

1. Select start option from the Run menu (or use F5 function key as you must have used this key if you are familiar with LOTUS 1-2-3). It tells the Visual Basic to switch from design mode to run mode. You can use to actually run your developed programs.

2. Select Break (or Ctrl+Break key pressed together) from the Run menu to bring the debug window to the front i.e. to make it an active window.
You will see a window as shown in the picture called as the Debug Window.
YOU are familiar with the word processor or a text editor. The debug window is somewhat like a text editor because characters appear on the monitor screen as you start typing. You can use the mouse to select characters and edit them. But the debug window is different from the text editor in one very important way. When you press "Enter" key, the debug window tries to execute what you have typed. To make you understand this statement in a more simple way, when you enter a BASIC command say "Beep", this will emit a beep sound from your computer speaker. This command is used to draw the attention to it self. Enter this command by typing BEEP and press Enter key. The cursor moves to the next line as shown in the picture below as your computer makes a sound.
If you type an incorrect command in the debug window and pressed Enter key; like you have typed BEAP instead of BEEP and pressed Enter key, Visual Basic displays an alert box to give

you an error message, meaning you have done something wrong and you are being asked to check and correct it.

To close the debug window, follow the procedure given below:

Type "END" in the debug window and press Enter key. You will see the debug window will disappear from the screen and Visual Basic is now back in design mode.

Visual Basic has three modes that it displays on the Title bar of the Visual basic:

Microsoft Visual Basic [design]

[design] appears while you are designingfdeveloping an application program
[run] appears while Visual Basic runstexecutes the current program
[break] appears while the program is stopped, which allows you to use the debug window

PRINT COMMAND

The Print command in Visual Basic has a very different meaning what you are generally familiar with. The Print command does not mean to send output to a printer. The Print command sends it to the screen. Isn't it not surprising that this is still called a Print command?

The "Basic" language was originally developed by two professors, John G. Kemeny and Thoumas E. Kurtz, at Dartmouth College around **1963-64.** At that time people were working with mainframe computers rather than **micro-computers/personal** computers. These computers occupied very large air-conditioned rooms. To communicate with such computers one used a Tele typewriter rather than a CRT screen. So whenever a program sent an output to the user, it was actually printed on a piece of paper rather than displayed on a screen. That is why Kemeny and Kurtz chose the same name Print.

To use Print command, Debug window must be active. When you type Print and Press Enter key, you will see the result of the command on your monitor as it gets executed:

When you press Enter key, the Print command displays the result in the debug window in the next line.

The syntax rules

Beep and End are the simplest commands. They are only one word long. But to communicate effectively with your computer, you need more words to build longer sentences.

Commands can also include additional information like equations. These equations are known as arguments.

VISUAL BASIC ARITHMETIC OPERATORS

The computer programme at times is required to perform number of arithmetic operations like additions, subtraction, division, multiplication etc. This language, like most of the other computer languages, supports all the arithmetic operations. Whenever you want to do some calculations between two numbers, use these operators.

To compute inputs from users and to generate results, we need to use various mathematical operators. In Visual Basic, except for + and -, the symbols for the operators are different from normal mathematical operators, as shown in Table 6.1.

Table 6.1: Arithmetic Operators

Operator	Mathematical function	Example>
^	Exponential	2^4=16

*	Multiplication	4*3=12,
/	Division	12/4=3
Mod	Modulus (returns the remainder from an integer division)	15 Mod 4=3
\	Integer Division(discards the decimal places)	19\4=4
+ or &	String concatenation	"Visual"&"Basic"="Visual Basic"

Chapter-2

Variables And Functions

Variables

After declaring various variables using the Dim statements, we can assign values to those variables. The syntax of an assignment is

Variable=Expression

The variable can be a declared variable or a control property value. The expression could be a mathematical expression, a number, a string, a Boolean value (true or false) and more.

Variable names in Visual Basic are made up of letters (upper and lower case) and digits. The underscore character, "_", is also permitted. **Names** must not begin with a digit. **Names** can be as long as you like.

Variable names in Visual Basic are made up of letters (upper and lower case) and digits. The underscore character, "_", is also permitted. **Names** must not begin with a digit. **Names** can be as long as you like.

Variable Names

Variable names in Visual Basic are made up of letters (upper and lower case) and digits. The underscore character, "_", is also permitted. **Names** must not begin with a digit.

Variable Type

Data types refer to an extensive system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted.

The following table shows all the data types available −

Data Type	Storage Allocation	Value Range
Boolean	Depends on implementing platform	**True** or **False**
Byte	1 byte	0 through 255 (unsigned)
Char	2 bytes	0 through 65535 (unsigned)
Date	8 bytes	0:00:00 (midnight) on January 1, 0001 through 11:59:59 PM on December 31, 9999
Decimal	16 bytes	0 through +/- 79,228,162,514,264,337,593,543,950,335 (+/-7.9...E+28) with no decimal point; 0 through +/- 7.9228162514264337593543950335 with 28 places to the right of the decimal
Double	8 bytes	-1.79769313486231570E+308 through -4.94065645841246544E-324, for negative values 4.94065645841246544E-324 through 1.79769313486231570E+308, for

		positive values
Integer	4 bytes	-2,147,483,648 through 2,147,483,647 (signed)
Long	8 bytes	-9,223,372,036,854,775,808 through 9,223,372,036,854,775,807(signed)
Object	4 bytes on 32-bit platform 8 bytes on 64-bit platform	Any type can be stored in a variable of type Object
SByte	1 byte	-128 through 127 (signed)
Short	2 bytes	-32,768 through 32,767 (signed)
Single	4 bytes	-3.4028235E+38 through -1.401298E-45 for negative values; 1.401298E-45 through 3.4028235E+38 for positive values
String	Depends on implementing platform	0 to approximately 2 billion Unicode characters
UInteger	4 bytes	0 through 4,294,967,295 (unsigned)
ULong	8 bytes	0 through 18,446,744,073,709,551,615 (unsigned)
User-Defined	Depends on implementing platform	Each member of the structure has a range determined by its data type and independent of the ranges of the other members
UShort	2 bytes	0 through 65,535 (unsigned)

Chapter-3

Build Project & Customize Form

ABOUT PROJECT

A project is defined as a collection of certain files. These files can be Project files (.VBP or .MAK), Form module files (.FRM), Custom control files (.OCX or .VBX), Standard module files (.BAS), Class module files (.CLS) and Resource Files (.RES). It is always better to create a separate directory for the new project which you have started. Whenever you start Visual Basic, it creates a new project for you and display a blank form window on the screen. In a new project, you can create many forms, class modules and standard modules.

To create a new project:

1. Create a new directory for the application in your hard disk if you have not created so far.
2. Start Window program and activate Program option.
3. Double click at the Visual Basic sub-menu to start the Visual Basic programme.
4. Now select **New project** from the file menu. As you double click the mouse button after selecting the "New Project", you will see the project window on your monitor.

The Project Window

The Project window displays a list of all the project's form modules, class modules, and standard modules. Through this window, one can get an access to the forms, modules, and custom control files that are associated with the project.

Working in the Project Window:

1. One can open the Form window for an existing form by selecting the form name and clicking the View Form button.
2. Since the project comprises of number of forms and modules, one can open as many forms and modules within a project. At a time, one can not open more than one Project Windows.
3. To remove file(s) from the project, it can be done by selecting the file in the Project window, and then Select Remove File option from the File menu.

Save a Project

Like any other Windows application, Visual Basic provides you with a File menu to save your project. Whenever you save a project, not only your project gets saved, but as well saves the forms file separately.

To save a project, first click on **Form1** form window to activate the menu bar. **Select Save Project. As** in case saving into a new file or **Save Project** if saving into the existing project, from the File menu:

It displays you a **Save File As** dialog box. From this dialog box you can save your project's form. By default it gives a form name (Form I), but you can also change this name in the File name box. This dialog box automatically adds the extension (.FRM) to the name you have given to the form file.

After saving the form file, Visual Basic displays another Save File As dialog box for saving the project. This dialog box also gives the default name for your project (Projectl) in the file name box but you may change the same as per your application. The extension to the project file name **(.VBP)** shall automatically be added by the system.

WHAT IS A FORM?

A form is nothing but a box that contains control objects. This is window or a dialog box that one creates with Visual Basic.

In Visual Basic, every project has at least one form that represents the program's main window. Projects can also have multiple forms or dialog boxes and data entry screens. Although a form's

main purpose is to hold other components such as buttons and text boxes, a form can also perform
actions in response to events such as key presses and mouse clicks.
A form represents a window's or dialog box's visual appearance. A form is actually a Visual Basic
component with its own set of properties (i;e. the special variables that permit you to change the way an object looks or works) and events (i.e. Visual Basic events include Click, Double Click, Movement of mouse. Press of any key on the keyboard). Unlike other tools, however a form Components does not appear on the toolbox.

Create a form

You can create a form object in one of two ways:

- Whenever you create a new project, Visual Basic automatically creates a form.
- Select Form from the insert menu.

You can use this method to add many forms in your project

Save a form

The forms which are created in the "Project" form part of the Visual Basic Project There is no need to save them separately because whenever one saves a project, Visual Basic automatically saves the form as well.

FORM EVENTS

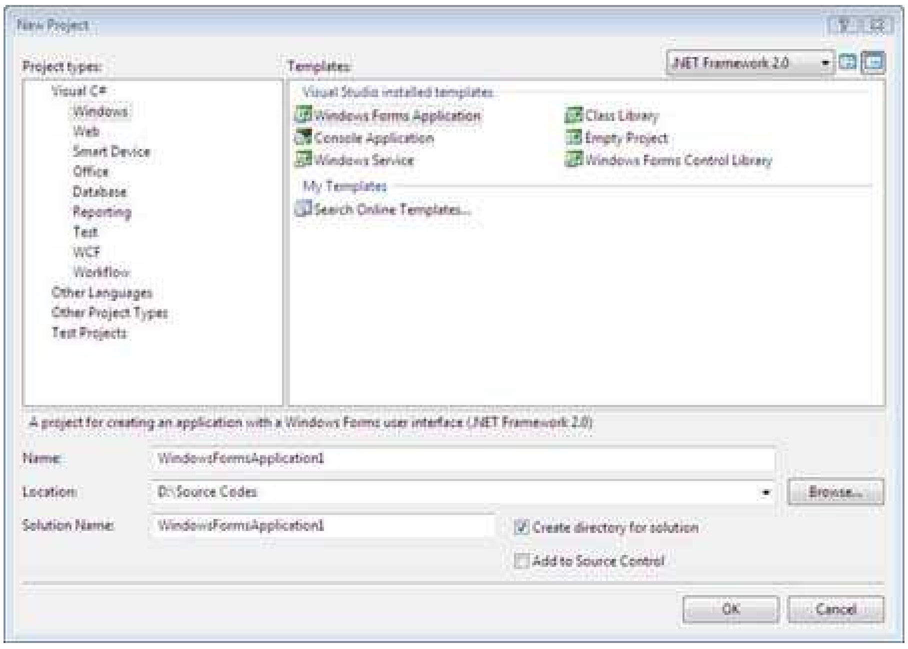
All windows base program is built around events i.e. what is going on in computer. An event is an action like a press of key or click of the mouse button etc. Each event has a particular action to be performed.

The events in the lifecycle of a Form from the time it is launched to the time it is closed are listed below:

- Move: This event occurs when the form is moved. Although by default, when a form is instantiated and launched, the user does not move it, yet this event is triggered before the Load event occurs.
- Load: This event occurs before a form is displayed for the first time.
- VisibleChanged: This event occurs when the Visible property value changes.
- Activated: This event occurs when the form is activated in code or by the user.
- Shown: This event occurs whenever the form is first displayed.
- Paint: This event occurs when the control is redrawn.
- Deactivate: This event occurs when the form loses focus and is not the active form.
- Closing: This event occurs when the form is closing.
- Closed: This event occurs when the form is being closed.

- Let us view this through an example.

1. First, launch Visual Studio IDE (2005 or 2008) and create a Windows Forms application.



A project for creating an application with a Windows Forms user interface (.NET Framework 2.0)

Name: WindowsFormsApplication1

Location: D:\Source Codes

Solution Name: WindowsFormsApplication1

[✓] Create directory for solution

[ ] Add to Source Control

Figure 1: New Project Dialog Box

2. Give a suitable name and click OK. This will create the application and open it in Design view.

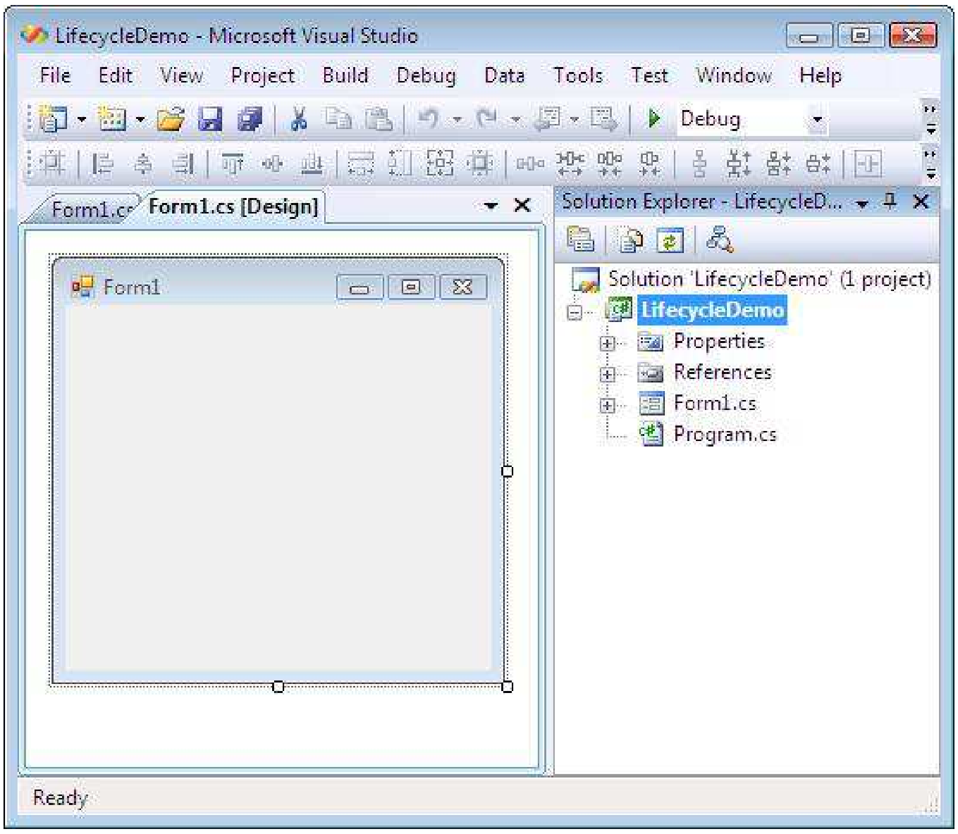
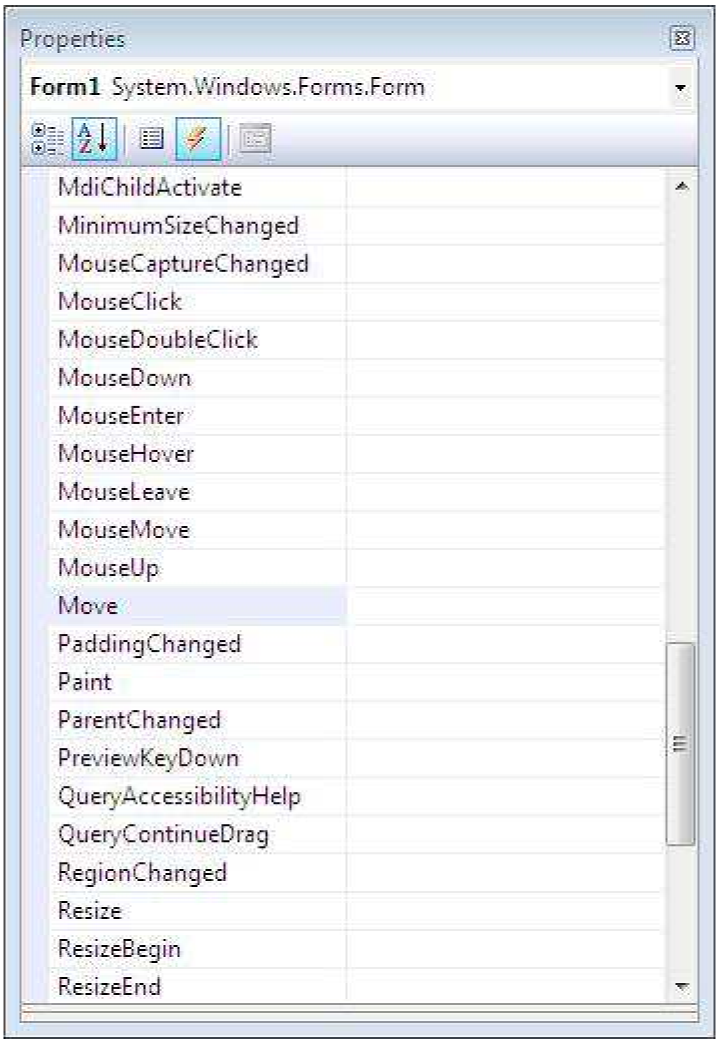


Figure 2: Application in the Design view

3. Open the Form properties window. The easiest way to do this is: select the Form in the design view and press the F4 key.

4. Click the Events tab and select the Move event.


Properties

Form1 System.Windows.Forms.Form

MdiChildActivate
MinimumSizeChanged
MouseCaptureChanged
MouseClick
MouseDoubleClick
MouseDown
MouseEnter
MouseHover
MouseLeave
MouseMove
MouseUp
Move
PaddingChanged
Paint
ParentChanged
PreviewKeyDown
QueryAccessibilityHelp
QueryContinueDrag
RegionChanged
Resize
ResizeBegin
ResizeEnd

Figure 3: Form events

5. Double click on it. This will cause the event handler to be auto-generated in the Code View.

6. Switch back to the Design View and in the Form properties window, double click the Load event.

7. Likewise, repeat this procedure for all the events that were listed earlier.

8. Open the Code View of Form1.Designer.cs and add the code marked in bold.

Visual Basic Control

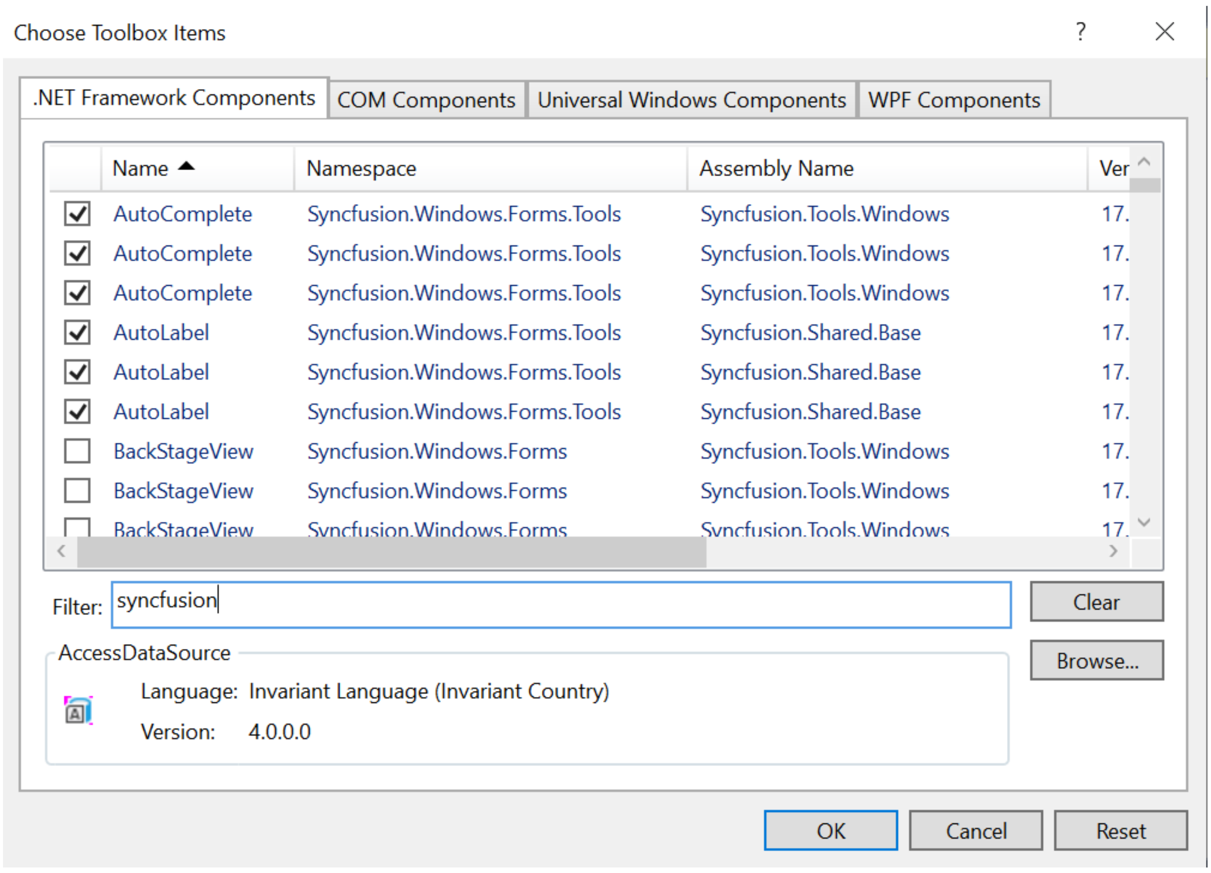
Custom Control

You can add custom controls to the Visual Studio toolbox so that they can be used easily in Visual Studio projects just by dragging them in the designer. To do so, you don’t have to add the assembly reference manually. The reference is automatically added to the Visual Studio reference manager.

Add custom controls to Visual Studio toolbox

To add custom controls to Visual Studio, create a new project (ASP.NET Web Forms, WPF, or WinForms) and open the designer. The toolbox items will only be visible in the designer window. If the toolbox is not visible, click the **View** menu, and then select the **Toolbox** option. Now, the toolbox will be displayed. Then:

1. To create a new tab for the custom controls, right-click anywhere in the Toolbox window, select **Add Tab**, and then provide a name for the new tab.
2. Then, right-click anywhere inside the new tab, and then select **Choose Items**. The Choose Toolbox Items dialog box will be displayed.



	Name ▲	Namespace	Assembly Name	Ver
✓	AutoComplete	Syncfusion.Windows.Forms.Tools	Syncfusion.Tools.Windows	17.
✓	AutoComplete	Syncfusion.Windows.Forms.Tools	Syncfusion.Tools.Windows	17.
✓	AutoComplete	Syncfusion.Windows.Forms.Tools	Syncfusion.Tools.Windows	17.
✓	AutoLabel	Syncfusion.Windows.Forms.Tools	Syncfusion.Shared.Base	17.
✓	AutoLabel	Syncfusion.Windows.Forms.Tools	Syncfusion.Shared.Base	17.
✓	AutoLabel	Syncfusion.Windows.Forms.Tools	Syncfusion.Shared.Base	17.
☐	BackStageView	Syncfusion.Windows.Forms	Syncfusion.Tools.Windows	17.
☐	BackStageView	Syncfusion.Windows.Forms	Syncfusion.Tools.Windows	17.
☐	BackStageView	Syncfusion.Windows.Forms	Syncfusion.Tools.Windows	17.

Filter: syncfusion

AccessDataSource

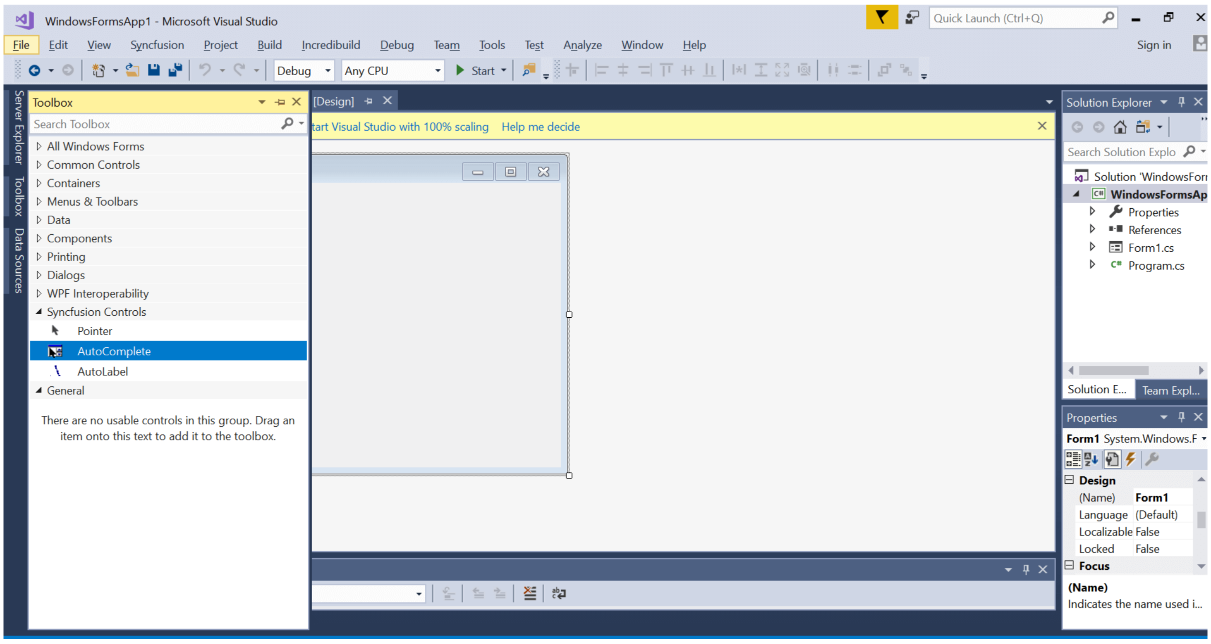
Language: Invariant Language (Invariant Country)

Version: 4.0.0.0

3. This dialog box shows a list of controls present in GAC that can be added to the toolbox. The check box in the selected state means those items will already be present in the toolbox.

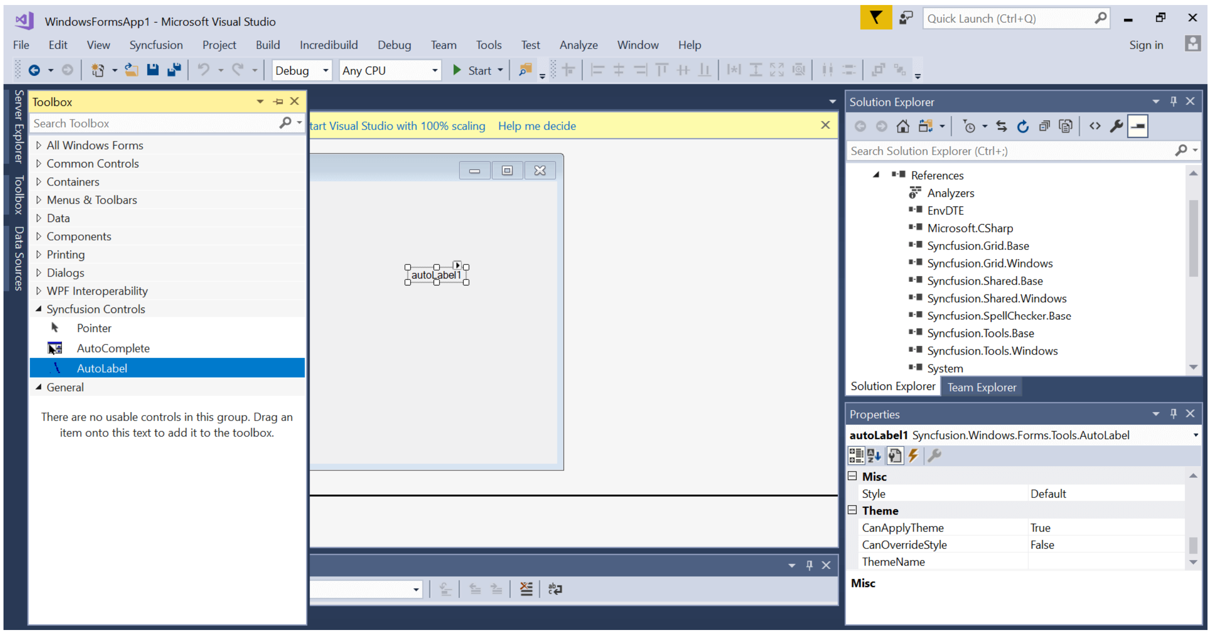
4.Select the custom controls from the available list, and then click **OK**.

5.The selected controls will now be displayed in Visual Studio toolbox.


WindowsFormsApp1 - Microsoft Visual Studio
File Edit View Syncfusion Project Build Incredibuild Debug Team Tools Test Analyze Window Help
Toolbox
Search Toolbox
All Windows Forms
Common Controls
Containers
Menus & Toolbars
Data
Components
Printing
Dialogs
WPF Interoperability
Syncfusion Controls
Pointer
AutoComplete
AutoLabel
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Solution Explorer
Properties
Form1 System.Windows.F

6.Alternatively, you can also add tools by navigating to the assembly location, and dragging the assembly directly to the newly created tab.

7.Once the controls are added, you can drag the control to the designer from the Visual Studio Toolbox and start using it. The assembly reference will automatically be added in the reference manager.



Control in Form

You can place controls on the surface of a valid VB container object. Standard container objects in VB include:

- Forms
- PictureBoxes
- Frames

Although Image controls and PictureBox controls have many features in common, Image controls cannot act as containers. You can use two different methods to add controls to a form or other container object.

The first method uses a mouse double-click and requires the following steps:

STEP BY STEP
Adding Controls to Forms: Method 1

1. Select the form or other container object (such as a PictureBox or Frame) where you wish to add the control
2. Double-click the control's icon in the ToolBox. A new instance of the control will appear in the selected container.

With the second method, you draw the control on the container surface by following these steps:

STEP BY STEP
Adding Controls to Forms: Method 2

1. Single-click the control's icon in the ToolBox.
2. Release the mouse button and move the mouse pointer to the position on the container where you want to locate the control.
3. Hold down the mouse button and draw a rectangle on the container surface to indicate the size and position for the control instance.

Example 1: Program to change background color

This example changes the background colour of the form using the **BackColor property**.

```
Private Sub Form_Load()
Form1.Show
Form1.BackColor = &H000000FF&
End Sub
```

Example 2: Program to change shape

This example is to change the control's Shape using the **Shape property**. This code will change the shape to a circle at runtime.

```
Private Sub  Form_Load()
Shape1.Shape = 3
End Sub
```

Chapter-5

Procedures and Functions in Visual Basic

Procedures and functions provide a means of producing structured programs. Rather than repeating the same operations at several different places in the program, they can be placed in a procedure or function. This not only makes the program easier to read, but saves a lot of time and effort in maintaining the program.

Using Procedures

In Visual Basic Programming, the one thing that you have learned so far is that the application is made up of small, self contained segments, similarly the code is also divided into small segments called procedures. You will work on one procedure at a time.

Procedures are usefull for supplimentry repeated tasks such as frequently used calculations. Suppose you are writing an application that at same time is converting inches to centimeters or calculating the smaller number among two numbers. You can always do calculations on time and repeat them in your code whenever they are needed or you can write a procedure that performs the calculations and call this procedure whenever needed. The advantage of using procedure is that the code is clean, easier to understand and mantain. If you discover a more efficient way to implement the particular task, then you dont have to change the whole code, Just go to that procedure and change it at that place.

The events we have been using so far are a special form of procedure known as an event procedure. For example, associating code with a CommandButton to quit an application is a procedure. The basic Syntax for a procedure is:

```
[Private | Public][Static] Sub procName ([arglist])
[statements]
[exit sub]
[statements]
[exit sub]
```

Parts of the Procedure

Part	Description
Public	Indicates that the procedure is available to all modules. If Option Private is used in the module, the procedure is not available to modules outside of the project.
Private	Indicates that the procedure is only available to other procedures or functions in the current module or form.
Static	Indicates that all variables declared within the procedure are retained, even when the procedure is out of scope.
procName	The name of the procedure. Must be unique to the module if declared Private, otherwise unique to the project. The name of the procedure follows the naming rule for Variables.
arglist	A list of variables passed to the procedure as arguments, and their data types. Multiple arguments are separated by commas. Arguments may be Optional, and may be Read Only.
statements	Any group of statements to be executed within the body of the Sub procedures

The following example is a Private Procedure to print the sum of two numbers on the Form.

```
Private Sub printSum(ByVal x As Integer, ByVal y As Integer)
  Debug.Print Str(x + y)
End Sub
```

Optional Arguments

If an argument has the Optional keyword in front of it, then that argument does not have to be provided to the procedure. MsgBox is an example of a procedure that takes optional arguments. If an argument is omitted, the comma must still be used as a placeholder. The following calls the MsgBox procedure, but omits the second parameter that describes the dialog box.

```
MsgBox "Hello", "Greeting"
```

The following example is a procedure that uses optional arguments to specify a recipient and a sender.

```
Private Sub greeting(strMessage As String, _
            Optional strRecipient As String, _
            Optional strSender As String)
If strRecipient <> "" Then
  Debug.Print "To " & strRecipient & ", ";
```

```
End If
If strSender <> "" Then
   Debug.Print strSender & " sends the message ";
End If
Debug.Print strMessage
End Sub
```

The greeting procedure may be called with any of the following:

```
greeting "Hello, how are you"
greeting "Hello, how are you?", "Computer"
greeting "Hello, how are you?", "Computer", "Gez"
greeting "Hello, how are you?", , "Gez"
```

Read Only Arguments

The optional parts ByRef and ByVal are used to determine whether an argument is a copy of the variable, or the actual variable. ByRef indicates that the variable is passed by reference, and any changes made within the procedure to the variable will be reflected to where the procedure was called. ByRef is the default in Visual Basic 6, but this is changed in VB.Net. ByVal indicates the variable was passed by value, and any changes made within the procedure to the variable will not be reflected to where the procedure was called as it's only a copy.

This example is a procedure that alters a variable. The actual variable from where it was called will be changed.

```
Private Sub changeWhereCalled(ByRef num As Integer)
   num = num + 1
   Debug.Print Str(num)
End Sub
```

This example is a procedure that alters a variable, but does not change the value where it was called.

```
Private Sub noChangeWhereCalled(ByVal num As Integer)
   num = num + 1
   Debug.Print Str(num)
End Sub
```

Using Functions

A function is similar to a procedure, except it returns a value to the calling code. The basic Syntax for a function is:

```
[Private | Public][Static] Function funcName ([arglist]) As Data Type      ' Procedure body here
   [funcName = expression]
End Function
```

The following example uses the functions "getFormattedDate", "getWeekName" and "getAdornment" to illustrate the use of functions in Visual Basic.

```
Private Function getFormattedDate(ByVal dateValue As Date) As String
  Dim strDate As String
  strDate = getWeekName(DatePart("w", dateValue))
  strDate = strDate & DatePart("d", dateValue)
  strDate = strDate & getAdornment(DatePart("d", dateValue))
  strDate = strDate & Format(dateValue, "mmmm yyyy")
  getFormattedDate = strDate
End Function
Private Function getWeekName(ByVal day As Integer) As String
  Select Case day
    Case 1
      getWeekName = "Sunday "
    Case 2
      getWeekName = "Monday "
    Case 3
      getWeekName = "Tuesday "
    Case 4
      getWeekName = "Wednesday "
    Case 5
      getWeekName = "Thursday "
    Case 6
      getWeekName = "Friday "
    Case 7
      getWeekName = "Saturday "
  End Select
End Function
Private Function getAdornment(ByVal day As Integer) As String
  Select Case day
    Case 1
      getAdornment = "st "
    Case 2
      getAdornment = "nd "
    Case 3
      getAdornment = "rd "
    Case Else
      getAdornment = "th "
  End Select
End Function
```

Modules

Modules contain procedures or functions that may be called anywhere within the project if they're declared as Public. To add a new module into the current project, either select "Add Module" from the Project menu, or right-click the Project in the Project Explorer and select

"Add", then "Module". The module only has one property, Name. The three-letter mnemonic for a Module name is "mod" (eg. modMathRoutines). There are two types of modules used in VB:

1. ***Standard Module:*** *The code that is not related to a specific form or control can be placed in the type of module called as standard module. It is stored in the file having extension (.BAS). A procedure that might be used in response to events in several different objects should be placed in a standard module, rather than duplicating the code in the event procedure for each object.*
2. ***Class Module:*** *The class module is used to create objects that can be called from procedures within your application. The class module is stored in the different file having extension (.CLS) The main difference between the class module and standard module is that a standard module contains only one code where as a class module contains both Code and Data.*

Recursion

Procedures and functions may be recursive (may call itself). Each call to itself requires that the current state of the procedure is pushed onto the stack. This is important to remember as it's easy to create a stack overflow, i.e. the stack has ran out of space to place any more data.

The following example calculates the Factorial of a number using recursion. A factorial is a number multiplied by every other integer below itself, down to 1. For example, the factorial of the number 6 is:

```
Factorial 6 = 6 * 5 * 4 * 3 * 2 * 1
```

Therefore the factorial of 6 is 720. It can be seen from the above example that factorial 6 = 6 * factorial 5. Similarly, factorial 5 = 5 * factorial 4, and so on. The following is the general rule for calculating factorial numbers.

```
factorial(n) = n * factorial(n-1)
```

The above rule terminates when n = 1, as the factorial of 1 is 1.

```
Private Function factorial(ByVal Num As Double) As Double
  'The factorial of 1 is 1, so terminate recursion,
  'otherwise, implement recursion
  If Num = 1 Then
    Factorial = 1
  Else
    Factorial = Num * Factorial(Num - 1)
  End If
End Function
```

Sub Main

Sub Main is a special procedure that may be used by Visual Basic to launch an application. The procedure should be written in a module. The procedure may not be declared using the keyword Private. The Sub Main procedure should be selected as the "Startup Object" in the "Project Properties" from the "Project" menu.

This example uses Sub Main in a Module to prompt for a name. The name is then added to the caption of the Form. To try this example, add a Module to your project and change the Startup Object in Project Properties to Sub Main. This example uses a form called frmStart. Change it to the name of the Form you want to start.

```
Public Sub Main()
  Dim strName As String
  strName = InputBox("Enter your name", "Name")
  frmStart.Caption = "Hello " & strName
  frmStart.Show
End Sub
```

Strings Functions

Sub strings

Left, Right and Mid are three Visual Basic functions to locate sections of a string. All of the following examples use the string "strExample" with the following value.

```
strExample = "Please try to do what you can do"
```

The Left function returns the requested number of characters from the left side of the string.

```
Me.Print Left(strExample, 10)
```

Returns, "Please try"

The Right function returns the requested number of characters from the right side of the string.

```
Me.Print Right(strExample, 18)
```

Returns, "do what you can do"

The Mid function returns the middle of a string. It takes three parameters, the string, the starting position and the length of the string to be returned.

```
Me.Print Mid(strExample, 8, 6)
```

Returns, "try to"

The Len function is used to determine the length of a string. The next example uses the Len and the Mid function to print a string of text down the form.

```
Dim counter As Integer
Const strExample As String = "Please try to do what you can do"
```

```
For counter = 1 To Len(strExample)
    Me.Print Mid(strExample, counter, 1)
Next counter
```

String Concatenation

Strings are joined together using the & operator.

```
Dim strFirst As String, strSecond As String, strResult As String
strFirst = "Too "
strSecond = "bad!"
strResult = strFirst & strSecond
Me.Print strResult
```

The concatenation operator is useful for building up long strings a bit at a time.

```
Dim strSQL As String
strSQL = "SELECT Surname, TelNo " & _
    "FROM Customer " & _
    "ORDER BY Surname"
```

Searching a String

InStr and InStrRev are two Visual Basic functions to locate one string inside another. InStr locates the first occurrence of the string, and InStrRev locates the last occurrence.

```
Const strExample As String = "Please try to do what you can do"
Dim pos As Integer
pos = InStr(strExample, "do")
If pos > 0 Then
    Me.Print "Found first occurrence of string at position " & pos
Else
    Me.Print "String not found"
End If
```

Returns, "Found first occurrence of string at position 15"

```
Const strExample As String = "Please try to do what you can do"
Dim pos As Integer
pos = InStrRev(strExample, "do")
If pos > 0 Then
    Me.Print "Found last occurrence of string at position " & pos
Else
    Me.Print "String not found"
End If
```

Returns, "Found last occurrence of string at position 31"

Changing the case of a String to Uppercase

The following example produces a MessageBox with, "please try to do what you can do".

```
Dim strExample As String
strExample = "Please try to do what you can do"
MsgBox UCase(strExample), vbInformation, "Uppercase"
```

Changing the case of a String to Lowercase

The following example produces a MessageBox with, "Please try to do what you can do".

```
Dim strExample As String
strExample = "Please try to do what you can do"
MsgBox LCase(strExample), vbInformation, "Lowercase"
```

Creating Arrays from Strings

Sometimes it's necessary to break up strings into smaller strings, for example if you wanted to submit a sentence as a list of words to a database query. The Split function allows you to break a string into an array, based on a delimiter. The first parameter to the Split function is the string, the second parameter is the delimiter. The function returns an array of the elements that have been split out of the string. Using a delimiter of a space will break out the words. The following example writes each word in a TextBox called txtSentence and writes the words down the form.

```
Dim list As Variant
Dim strSentence As String
Dim counter As Integer
strSentence = txtSentence.Text
list = Split(strSentence, " ")
For counter = 0 To UBound(list)
    Me.Print list(counter)
Next counter
```

Converting Data to Strings

The Str function may be used to convert a non-String data type to a String data type.

```
Dim strNumWord As String
Const num As Integer = 64
strNumWord = Str(num)
```

Selecting Text in a TextBox

The text in a TextBox can be highlighted using the Visual Basic functions SelStart and SelLength functions. SelStart specifies the start of the selection, and SelLength specifies the number of characters to be selected.

The following example highlights whatever is in the TextBox when it gains focus.

```
Private Sub txtSurname_GotFocus()
  txtSurname.SelStart = 0
  txtSurname.SelLength = Len(txtSurname.Text)
End Sub
```

The next example, sets the text to "Lemon" and highlights the portion "emo".

```
Private Sub txtSurname_GotFocus()
  txtSurname.Text = "Lemon"
  txtSurname.SelStart = 1
  txtSurname.SelLength = 3
End Sub
```

The SelText function returns the selected text.

```
MsgBox txtSurname.SelText, vbInformation, "Selected Text"
```

Use in the above function, the MsgBox would print "emo".

The following example uses the Selection functions to create Copy, Cut and Paste menu items to copy, cut and paste from whichever TextBoxes are used in the application.

Using Dates in Visual Basic

The basic data type Date is used in Visual Basic to store the date and time. Date literals are enclosed within the #operator.

```
Dim birthDate As Date
birthDate = #1/20/1964#
```

The Format Function

The Format function is used to format dates and numbers. The actual result is dependent on the country settings on the computer. The general syntax is:

```
Format(expression, formatExpression)
```

Named Number Format Expressions

Named Part	Description
General Number	Number displayed normally, with no separators.
Currency	Number displayed with currency symbol, thousand separators and decimal places. The actual format depends on the locale.
Fixed	Displays at least one digit on the left of the decimal point, and two digits to the right of the point.
Standard	Same output as Fixed, but with a thousands separator.
Percent	Output is multiplied by 100 and has a % sign at the front. Displayed with two decimal places.
Scientific	Output is displayed as standard scientific (eg. 6.42E+01).
Yes/No	Displays No if number is 0, otherwise displays Yes.
True/False	Displays False if number is 0, otherwise displays True.
On/Off	Displays Off if number is 0, otherwise displays On.

Chapter-6

Accessing a Database

INTRODUCTION

Almost all kinds of information in computer is stored in some form or the other in databases for smooth access. Most of the high level languages provide means to access the database, only thing is in one language accessing databases may require a small code of programme and in other may be a large code is required to access databases. In Visual Basic, the procedures have been made simple by linking this language with Microsoft Access i.e. a database created using Microsoft Access can be easily accessed in Visual Basic and for users who are not very comfortable with Microsoft Access, Visual Basic has a programme that can be used to create a database.

WHAT IS A DATABASE?

A database is a collection of information related to a particular topic. The information isstored in tabular form. This kind of organization in a database is called a table. The columns of the table are referred as fields and the rows are referred to as records. The collection of the tables is called as database and are stored in a file e.g. Employee records in a file cabinet, a stamp collection in an album - each of these collections is nothing but **a** database. Database, typically consists of a heading that describes the type of information it contains and each row contains some information in fields. The information access can be made faster by indexing on one or more fields. The Visual Basic creates pointers when the indexing is done on any field and only these pointers are referenced to get the information rather than the complete table of data.
A database management system (DBMS) is a system that stores and retrieves information from a database. Creating, modifying, deleting, adding data in files and using this data to generate reports is called as data management. The software that allows the user to perform these functions is called a Data Base Management System.
Visual Basic allows you to create your own database in Microsoft Access. This facility is invoked by selecting Data Manager. With the use of Data Manager one can create a database. In the Microsoft Access, you can create tables, indexes, attach external tables and set relationship between two or more tables in the database.

USING DATA MANAGER

The Visual Basic can easily access the database created using Microsoft Access. If you are familiar with Microsoft Access, you may use it to create the databases. Also Visual Basic has an add-in programme called Data Manager that can be used to create Databases. This application allows you to create new databases, and examine or map databases in a variety of formats. Users can also create Attached tables to the data.
Visual Basic shares its database engine with Microsoft Access. Databases created with Visual Basic or Data Manager can be manipulated with Microsoft Access, and similarly databases created with Microsoft Access can be manipulated with Visual Basic and Data Manager.
To open a Data Manager to create Database
To open Data Manager window, select Data Manager from the Add-Ins menu.
As you select the Data Manager option and click it, the Data Manager window, where one can design a database appears. The file menu of the Data Manager window has only two menus i.e. **File** and H**elp.**

CREATING A DATABASE

The databases you create with Data Manager are Microsoft Jet databases and can be edited in both Microsoft Access and Visual Basic.
To create a new database, select **New Database** from the File menu. As you select the New Database option and click it, a dialog box appears.
Type in the name of the database in the File name box with the extension as .MDB, select the desired directory if you have not already selected the same and click on Save button. If you have not given the extension, the system shall add .MDB extension to your database file name automatically. Do not give any other extension as this shall create problems while accessing at a later stage using Visual Basic. Now the empty database has been created.
After the database has been saved, Data Manager window displays the Database window. This window contains six buttons: Open, New, Delete, Design, Attach, and Relations. At this moment since no table is created only New and Attach buttons are active. Once new database file has been created, add new tables, fields, and indexes to contain the data created. The table name and the fields for the table are defined in the dialog box. As you finish adding fields, click OK to close the table.
You can manipulate tables using the following commands.

Command	**Description**
Open	To open an existing table in the specified database and view the data
New	To create a new table.
Design	To open an existing table and view or modify the table structure.
Delete	To delete an existing table.
Attached Tables	To attach new tables and reattach existing tables.
Relations	Defines how data is related among tables.

Once the table has been created, if required you can select a desired field to create the indexes. As YOU select the table, the table editor appears. You can now click on the Indexes button to get the index window. Select Add to create index on one or more fields. If you do not want the duplicates to appear in the index, select Unique. Say OK and now your data shall be as per the index field you have specified.

CREATING A NEW TABLE

All of the data in the database will be stored in tables. Each table is defined as a set of fields. Each field describes the kind of data to be stored by specifying the data type, size, and other attributes.
1. To create a new table, Click on New in the database window.
2. It displays Add Table dialog box, where one can create new table and add fields to that table. You can use either the TAB key or the mouse clicks to move between the boxes.
3. In the above dialog box, number of text boxes have appeared and on each box some text is written indicating what is to be inserted in each box. The first box on top has Name written on it indicating that the name of the table has to be inserted in this text box. Type table name in the name box.
4. Press TAB key or use mouse to move to the next text box. The next box you see is having **FieldName** written on it. Type field name in the FieldName box.
5. Press TAB key or use mouse click to move to the next box. The next box has DataType written i.e. the type of data that shall be stored in this field. As you click on the right of this text box, a list of valid data types appear. You may select Data type from the Data Type list.
6. Next box has Size written on it. In this box the size of the data to be filled in this field is to be specified. Type Size in the Size box.
7. To add the next field name in this table, click on right arrow button to add field.
8. As you key in the field name, you find that the names of all the fields appear in the centre box. This is the table field box and the table fields box display you all fields of the table.

9.. Click on OK button to save table permanently in your Database.

ATTACHING A TABLE

Visual Basic permits the users to attach tables from the following database applications:

- Microsoft Access (databases other than the open database)
- Paradox (version 3.x and 4 .x .DB files)
- FoxPro (version 2.0 and *2.5* .DBF files)
- dBASE 111 and dBASE IV (.DBF files)
- Btrieve (with the data definition files FILE.DDF and FIELD.DDF)
- Microsoft SQL Server, Sybase SQL Server. and ORACLE Server (select <SQL Database>)

For SQL databases, Data Manager displays a Server Login dialog box.
After you attach an external table, set field properties for the table in the Design dialog box.
Although you can not change the structure of an attached table, you can set the field properties.

1. To attach a table, Click on **Attached Tables** button in the database window.
2. It displays an Attached tables dialog box. To attach a table, click on **New** button.
3. It displays a **New Attached Table** dialog box. This dialog box has number of text boxes requiring some related information to be put.
4. The first text box is the Attachment Name. Type attachment name to display as table in the database tables list.
5. Similarly the other text boxes are also filled with required information. Type database name, from where you want to attach a table. The path for the table must be specified.
6. Select database type from the connect string list box.
7. Select table from the table to attach list box.
8. Click on attach button to attach table in your database.
9. Click on Close button.

CHANGING DESIGN OF AN EXISTING TABLE

Some times one wants to modify an existing structure of **the** table. It is possible in design mode. You can change the property of fields of the table, add a new field to the fields that are already existing and even delete a field if it is not required.

1. To open an existing table in design mode, select a table from the tables list and click on **design** button from the database window.
2. You will see a Table Editor as shown below, where you can change the structure of the table.
3. To make a change in the properties of a field, Click on field name. As you click on the field name, the first property of that field gets selected. Now click on Edit button to display Edit Field dialog box.
4. you can make changes in the properties of any of the field names. After you are through with all the changes, click on OK button.

CREATING INDEXES

1. To create an index, click on **Indexes** button from the Table Editor dialog box.
2. You will see an Indexes dialog box, where one can create indexes.
3. Since you have not still selected any of the fields for indexing, click on **Add** button. This will -

bring up the Add Index dialog box, where one can select any field on which indexing is to be done.
4. When the Add Index dialog box appears, the first text box is the Index Name. You can type name in the Index Name box.
5. Now select the field from the list available in Fields in table list box. As you select the field, the button Add (Asc) and Add (Dec) become bright. This means that you can either have index in ascending order or in the descending order. In case you do not want the duplicates to appear then the Unique button should be selected.
6. Click on Add(ASC) button to add this button and indexing will be done on ascending order.
7. If the indexing is to be done only on one field then that becomes the primary field. In case the indexing is to be done on more than one field, the Primary Index button index must be selected to tell the system about the primary field.
8. Once you have completed, click on OK button to save this index.

WORKING WITH DATA

Data Manager gives you a way to manipulate data. You can add, delete or modify an existing record. Once you have created a table with different fields, you can start entering data into it. If you go back to the Data Manager Window, since you have already created a database, the Open button shall be active. To do all the manipulations, the table must be open.

To open a table, Select a table from the tables list and click on **Open** button.
As you click on the Open, you will see a window.
To add a record in your database, click on Add button to insert a blank record in a table.
you can add the values in the text boxes yourself or use the table provided. You may use TAB key to move from one field to another. You may fill all the fields or leave few blanks as per your requirement. Most important thing is that the primary key should never be left blank. Type the field values in the text boxes.
After all the data has been completed in the record, click on add button to save a record in a table. This will clear-off the entries of the first record i.e. the table will be updated with the first record and now the second blank record appears on the screen. Enter the required values in this record and so on. Once all the records have been added, click on Update button. This will save the last record entered and then automatically display the table beginning with the first record.

The table open window provide with few buttons to go to any record of the table i.e. to go to first record, or the last record, or the next record etc. If you find any mistake and wish to correct, you must again select the Update button after the correction.

In case you wish to delete a record, select that record and click on Delete button. After deletion press the Update button. This displays the table after the deletion process.
Once you are through with all the changes, click on Close button.

Chapter-7

Create Form with Data Control

INTRODUCTION: A collection of facts in raw form that become information after proper organization or processing is known as data. A database is a collection of data files, which are arranged into a single integrated file system which helps in minimizing duplication of data (by controlling redundancy) and provides convenient access to information within the system. Database Management System involves addition, modification and deletion of data in the database. In this chapter, you will be exposed to Data Controls using which you can add, modify or delete the data from the database.

WHAT IS DATA CONTROL?

Data controls along with other controls are used to display, add, modify, and delete data, from a database. To make the connection between a form and a database, one places a data control in the form and set its properties to identify the name of the database that one wants to access. If the form contains data from more than one table, that will need to use more than one data control for each table. The Data control enables you to move from record to record and to display and manipulate data from the records in bound controls. Without a Data control, data-aware (bound) controls on a form can not automatically access data. Creating Data Used to provide access to data in databases through bound controls Form with on the form. Data Controls One can perform most of the data access operations using the Data control without writing any code at all i.e. the Visible property is made False. Data-aware controls bound to a Data control automatically display data from one or more fields for the current record or, in some cases, for a set of records on either side of the current record. The Data control performs all operations on the current record. If the Data control is instructed to move to a different record, all bound controls automatically transfer changes to the Data control to be saved in the database. The Data control theti moves to the desired record and passes back data from the current record to the bound controls where it is displayed. The Data control automatically handles a number of contingencies including empty record sets, adding new records, editing and updating existing records, and handling some types of errors. However, in more complicated applications, one needs to trap some error conditions that the Data control can not handle.

WHAT IS DATA-AWARE CONTROL?

A data-aware control is one that can provide access to a specific field in a database through a data control. A data-aware control can be bound to a data control through its data source and data field properties. When a data control moves from one record to the next, all bound controls connected to the data control change to display data from fields in the current record. When users change data in a bound control and then move to a different record, the changes are automatically saved in the database. The DBList, DBCombo, and DBGrid controls are all capable of managing sets of records when bound to a Data control. All of these controls permit several records to be displayed or to be manipulated at a time.

DbList : This is used to display a list of items from which the user can choose any one item. The list can be scrolled if it has more items than can be displayed on the screen at one time. DBList is a custom control and has increased data access capabilities.

DbCombo: This is used to draw a combination list box and text box. The I user can either choose an item from the list or enter a value in the text box. DBCombo is a custom control and has increased data-access capabilities that the standard ComboBox does not have.

DbGrid: This is used to display a series of rows and columns and to manipulate the data in its cells. DBGrid is a custom control and has increased data access capabilities that the standard Grid does not have.

The CheckBox, TextBox, Label, Picture, Image, ListBox and ComboBox controls are also dataaware controls and can be bound to a single field of a Recordset managed by the Data control. Additional data-aware controls like the MaskedEdit and 3DCheckBox controls are also available. windows Programming

CREATING A FORM USING DATA CONTROLS & Visual Basic :

1. To use the Data control in a form, select Data from the Toolbox and draw it on the form.

2. As you are aware, you can connect any type of database i.e. either from Access. Or dBASE or Excel, but we are taking the example of Access. . So select Access from the Connect property of Data control.

3. Type database name with complete path or select database from the open database dialog box.

4. Select Table name from the Recordset property.

5. To display fields and their records, select DbGrid from the Toolbox and draw it on the form.

6. Select Datasource property of DbGrid as Data1 . Data1 is the name of Data control.

7. Now select Start from the Run menu to see your data form in running mode.

MANIPULATING DATA: Creating Form with Data Controls To move between records You can manipulate the Data control with the mouse, moving from record to record or to the beginning or end of the Recordset. The Data control does not permit the user to move to either end of the Recordset using the mouse. You can not set focus to the Data control.

To insert and delete records from the table, first, change the addnew and delete property of the I DbGrid to True.

To add a record : Put a command button on the form and change its caption as Insert. To add a record in a table, write the following code in click event of this command button.

Put another command button on the form and change its caption as Save. To save a record in a table, write the following code in click event of this command button.

To delete a record : Put another command button on the form and change its caption as Delete. To delete a record from a table, write the following code in click event of this command button.

Now Select Start from the run menu and work with data in the form. Click on Insert button and then type the values in the fields and lastly click on update button to save this record in a table permanently.

CREATING THE MENU BAR: Now your form has all the controls that are required, the only thing left to do is to create the menu bar.

Visual Basic provides a visual tool for building menu bars. One can use the Menu Editor to create a menu bar that displays the names of menus available to use in the active window.

To create a menu bar :

1. Open the Form for which you want to create a menu bar. Select Menu Editor from the Tools Controls menu. You can also use the keyboard keys (CTRL+E) to get the Menu Editor.

2. It will display you the Menu Editor as shown below. You will find a few of the text boxes and check boxes in the Menu Editor dialog box. Each text box and check box expects some answer so as to create a menu.

3. In the Caption text box, type the caption as you want it to be displayed on the menu bar.

4. In the name text box, type the menu item name. This is the name that will be used in the code to refer to the menu item. Please ensure that the names given must be unique and should not be repeated.

5. You may also specify the shortcut keys if so desired. To specify the shortcut key, select the shortcut key from the shortcut list box.

6. Click on Next button to insert other items.

7. In the Caption text box, type the caption as you want it displayed as menu item of the menu bar.
8. Click on right arrow button to tell Visual Basic, this is the menu item of the menu bar. In the same way you can also create the submenu of the menu item.

9. Similarly you can create the other menu items.

10. To create a separator bar in the menu, type a hyphen (-) in the Caption box.

11. In the Name text box, type a control name that you will be used to refer to the menu item in code. This name can be identical to the caption.

12. In the bottom of the Menu Editor, display the menu bar and menu item details.

13. To insert a new menu item, select the item before which you want to insert and then click on the insert button.

14. To delete a menu item, select the item and click on the Delete menu.

15. Click OK button to accept all the changes and apply them to the selected Form. Finally, Menu will be displayed in the form. Creating Form with Data Controls You can arrange the menu items in order you want them to appear in the application menu bar and drop-down menus. The menu you created is visible at design time, but no events are generated until you write code. So you can write code for menu items like you have written for the Command Buttons. Just double click on the menu item to display code window and write your code.

DISPLAYING A MENU ITEM CODE : To display the code for a menu item, use the mouse to choose the menu item from the form menu bar or display the code window for the form that contains the menu item. In the object box, select the desired menu item. Thus a code window shall be displayed that contains items click procedure.

Chapter – 8

Object Linking and Embedding

Introduction:

Object Linking and Embedding means to create small and simple programs at different locations and then combine them to make a complicated large program. A file created under one application can be linked with another file created in some other application and if the changes are made in one file the changes are automatically transferred in the composite file.
About OLE:

Linking means storing multiple files in separate locations but displaying them within a single programme i.e even if the two files are created under different applications, they can be combined and viewed under one application. If the changes are made in one file then these shall get reflected in the combined file automatically.
Embedding means inserting one file into another file may be created under different applications.
In this since one file gets inserted into another file, any change made later in a file does not affect these changes in the combined file.
The main difference between linking and embedding is that linking stores files in separate locations, but embedding stores multiple files in one file itself.

TERMS IN OLE

Object linking and embedding (OLE) enables a programmer of Windows-based applications to create an application that can display data from many different applications. This also enables the user to edit that data from within the application in which it was created. The changes made in the primary file get automatically transferred in the linked file. In some cases, the user can even edit the data from within the Visual Basic application. The following terms and concepts are fundamental to understanding the concepts of OLE in Visual Basic.

Object:

An **object** refers to a discrete unit of data supplied by an application. An application can expose many types of objects. For example, a spreadsheet application can expose a worksheet, macro sheet, chart, cell, or range of cells all as different types of objects.
There are three ways to create an object in Visual Basic:
1. In this technique the object is embedded within the interface of a form in the application. The object is added to the toolbox using the Custom Controls command on the Tools menu, and then the object is drawn directly on a form.
2. In this technique the object is created in a running instance of the application that provides the object. Use the Createobject or Getobject functions to create the object in code.
3. In this technique one can change the objects on the form at run time, create linked objects, and bind the OLE container control to a Data control. Thus one can embed or link the object within an OLE container control.

Linked Object

Linking an object means that the data associated with a linked object is stored by the application that supplied the object. The application only stores link references that display a snapshot of the source data. The location of the file does not get changed.
When you link an object, any application containing a link to that object can access the object data and change it. For example, if you link a text file to a Visual Basic application, the text file can be modified by any application linked to it. The modified version appears in all documents linked to

this text file. You use the OLE container control to create a linked object in your Visual Basic application.

Embedded Object

Embedding means inserting one file into another file. In this since one file gets inserted into another file, any change made later in a primary file does not affect these changes in the combined file. When you create an embedded object, all the data associated with the object is contained in the object. For example, if a spreadsheet were an embedded object, all the data associated with the cells would be contained in the OLE container control object, including any formula. The name of the application that created the object gets saved along with the data.

If the user selects the embedded object while working with the Visual Basic application, the uspreadsheet application can be started automatically so the user can edit the cells. When an object is embedded in an application, no other application has access to the data in the embedded object. You can use embedded objects to maintain data in your application that is produced and edited in another application.

Container Application

Container Application means an application that receives and displays data of an object. For example, a Visual Basic application that uses an OLE container control to embed or link data from another application is a container application.

Source Application

The application, form, or control that sends information and commands when two or more programs that support dynamic data exchange (DDE) are running simultaneously.

OLE AUTOMATION

Some applications provide objects that support OLE Automation. You can use Visual Basic to manipulate the data in these objects through programming. Some objects that support OLE Automation also support linking and embedding.

If an object in an OLE container control supports OLE Automation, you can access its properties and methods using the Object property. If you draw the object directly on a form or create it in - code, you can directly access the properties and methods of the object.

OLE CONTROL POPUP-MENU

Each time you draw an OLE control on a form, the insert object dialog box is displayed. You use this dialog box to create a linked or embedded object. By choosing Cancel, an object is not created. At design time you click the OLE control with the right mouse button to display a pop-up menu. The commands displayed on this pop-up menu depend on the state of the OLE control as displayed in the following table.

Insert Object : This is used to insert a new object into the document. This command is always enabled.

Paste Special: This is used to copy a part of an OLE object. The part of the object is copied on to the clipboard. The clipboard contains a valid OLE object.

Delete Embedded Object: The OLE control contains an embedded object.

Delete Linked Object: The OLE control contains a linked object.

Create Link : The sourceDoc property is set.

Create Embedded Object: The Class or SourceDoc property is set.

Create OLE Object at Design Time

Once an OLE control is placed on a form, the Insert Object dialog box appears as shown in the figure below. This dialog box is used to specify the OLE object you would like to link or embed in the Visual Basic program. Since Insert Object option is enabled, the Insert Object dialog box

automatically appears. The steps involved to create a program are as follows:
1. Once the Insert Object dialog box has appeared, click on Cancel to remove this dialog box. Click on the command button icon and draw two buttons on the form.
2. Click on command1 button and press F4. The properties window shall appear. This window can be got by choosing View -> Properties.
3. Change the caption of command 1 to Use OLE.
4. Click on command2 button and press F4. The properties window shall appear
5. Change the caption of command2 to Exit.
6. Double click the Exit button to open the Code window.
7. Type End in the command2 procedure.
Private Sub Command2_click()
End
End Sub .
8. Choose commandl by clicking on Object list box.
9. Type OLE1 .InsertObjDlg in the commandl procedure.
Private Sub Commandl-click()
OLEI. InsertObjDlg
End Sub
When you specify an OLE object at design time, Visual Basic automatically sets the control properties. However, they can be modified through the properties window. The Visual Basic code tells how to use Insert Object dialog box to link or embed files.

CREATING PART OF AN OLE OBJECT

The above procedure links or embeds the complete file. In case a part of the file is to be used as an object, the steps involved are as follows.
The procedure explained above upto point number **8** is the same.
Type the following code in the commandl procedure.

```
Private Sub Command I-click()
If OLEI.PasteOK = True then
   OLEI. PasteSpecialDlg
End if
IfOL.EI.OLEType = 3 then
  MsgBox "Nothing has been copied"
End if
End Sub
```

This code shows how to paste an OLE object stored on the clipboard.

TESTING EMBEDDING / LINKING

To test the embedding capabilities once the Visual Basic program has been created, the following steps are followed.
1 . Switch on Paint application and draw any picture.
2. Select part of the picture using the selection tool.
3. Select Edit -> Copy from the menu bar. By this a portion of the picture that has been selected has been copied on the clipboard.
4. Switch on Visual Basic application.
5. Click the Use OLE button. The Paste Special dialog box appears.
6. Choose Bitmap Image in the Object list box.
7. Click OK. The Visual Basic program shall display the embedded data.
8. Click Exit to exit the program.
To test the linking capabilities once the Visual Basic program has been created, the following

1. Switch on any application like Microsoft Word or Microsoft Excel. Create any file and save it

under that application.
2. Highlight or select the contents of the file created and then select Edit -> Copy from the menu bar.
3. Switch on Visual Basic application and run your program by pressing F5.
4. Click the Use OLE button. The Paste Special dialog box appears.
5. Click on Paste Link radio button. Now click OK. The program shall display the linked OLE data.
6. Switch back to the file you created in Word or Excel.
7. Now switch back to Visual Basic program to see the changed file.
8. Click on Exit to close the program.
Thus one can test whether the embedding or linking of an object which has been created are working or not.