DEPARTMENT OF COMMERCE (CA)

VISUAL BASIC (Semester-III)

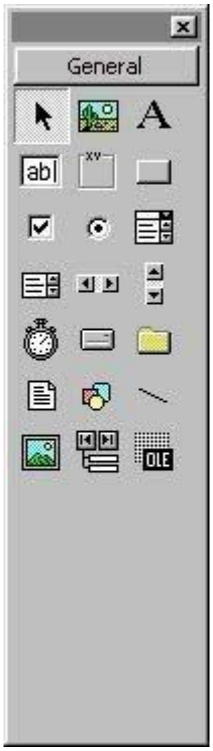
II M.COM (CA) Sub Code-18MCC32C

UNIT - II

Intrinsic Controls: Text box controls, Label and frame controls, command button, check box and option button controls, list box and combo controls, picture and image controls, drive-list box, dir-list box and file list box controls and other controls, control arrays.

Intrinsic Controls

When you start VB, you'll always find the intrinsic controls displayed in the toolbox. The controls are built-in to the VB files and do not exist in an external file (with a .OCX extension) the way the ActiveX controls do. In the toolbox, each of the controls has its own distinctive icon.



The Most Useful Intrinsic Controls

These nine intrinsic controls are pretty much used on every VB application I've written. Start your learning with these and then branch out. Further down on this page I have a brief comment on each of the controls.

Command Button

Image

CheckBox

TextBox

Label

Option Button

Picture Box

List Box

Combo Box

The Rest of the Intrinsic Controls

The other eleven intrinsic controls are also valuable but I find myself using these less often than the others. Also, you'll find that you use fewer of these within an application than you do of the nine that I listed as the most useful controls.

ADO Data Control

DirListBox

DriveListBox

FileListBox

Frame

Horizontal Scroll Bar

Vertical Scroll Bar

Line

Shape

OLE

Timer

Database Features

I've put the discussion of databases elsewhere in the tutorial, but you should know right now that several of the intrinsic controls can display or edit data directly out of a database. With VB, the ADO Data Control is used to access the database information and to distribute it on to the other intrinsic controls which can handle database information. VB uses the terminology "databound" to describe controls which have built-in features for handling database access.

Comments on Each Control

- **Command Button**
  This one works just like you expect. Press the button and it executes a block of code.
- **CheckBox**
  Typically this is used for turning on/off some particular feature of your program.
- **TextBox**
  This is the standard way of letting a user edit information. To make it convenient for the users, learn about the .SELLENGTH and .SELSTART properties which highlight text in the the textbox.
- **Image**
  Use this to display a picture. Use it over the PictureBox because it takes less operating system resources.
- **PictureBox**
  While it can display pictures, it also acts as an area on which you can print text and graphics. Use it for home-grown graphics or print previews.
- **Label**
  As the name suggests, this is used to label other controls. It's pretty passive and you'll seldom use its items other than the .CAPTION property.
- **Option Button**
  If you use it, you'll use it in groups. VB handles the feature that only 1 option button can be selected at a time.
- **ListBox**
  This is the first of the intrinsic controls to introduce methods common to some of the more complex controls. The use of a ListIndex which starts at 0 (not 1) is a confusing factor that you must watch in your code.
- **ComboBox**
  Whereas a listbox takes up space on the form, the combobox control minimizes the use of valuable form real estate. It has 3 modes of operation some of which allow you to keep your users from entering bad data.
- **ADO Data Control**
  If you're not accessing a database, then you don't need this one. If you are accessing a database you have to have this control to act as the interface to any other databound control. The exception is that VB offers ways to access databases directly from code, but for modest display/edit applications the ADO Data Control is very effective.
- **DirListBox / DriveListBox / FileListBox**
  You'll almost always use these in combination with each other. Read the HELP file for

how to synchronize them to work together. Often, however, you will use the CommonDialog Control instead of these.

- **Line**
  It can be used as a static display, or you can animate it with the .VISIBLE property and the .MOVE method.
- **Shape**
  When the line is not enough, this one supports rectangles and ellipses/circles. As with the line control, use its items to create animation.
- **OLEContainer**
  If you want to put objects on your VB application which come from other applications already on your machine (such as Word, Excel, ...) then this control is very useful. For my needs, I don't like making the assumption that my users have the application (in a specific version) on their machine to make distribution of my application go smoothly. I avoid this one whenever possible!
- **Frame**
  It's just a container - it can hold other controls. There are two very good reasons to use it. If you want multiple groups of option buttons then place each group in a frame and each group will operate independently. If you want to manipulate controls as a group (i.e., positioning) then put them in a frame and you can handle them all at one time.
- **Horizontal/Vertical ScrollBar Controls**
  Basically you let use the slider value of a scroll bar as the input for other code that you write. These are normally used in conjunction with other controls.
- **Timer**
  This is the most unusual of the intrinsic controls. By setting the .INTERVAL property this control will automatically create an event on a regular basis. No other control does this! You can use it to create an action at a certain time and then turn the control off to prevent repeats.

Control Array

In Visual Basic, a control array is a group of related controls in a Visual Basic form that share the same event handlers. Control arrays are always single-dimensional arrays, and controls can be added or deleted from control arrays at runtime. One application of control arrays is to hold menu items, as the shared event handler can be used for code common to all of the menu items in the control array.

Control arrays are a convenient way to handle groups of controls that perform a similar function. All of the events available to the single control are still available to the array of controls, the only difference being an argument indicating the index of the selected array element is passed to the event. Hence, instead of writing individual procedures for each control (i.e. not using control arrays), you only have to write one procedure for each array. Control arrays are no longer supported in Visual Basic 2008, as "changes to the event model" made them unnecessary.The Visual Basic Upgrade Wizard can convert code that uses control arrays into Visual Basic 2008 code that uses more recent structures.

REFERENCE:

1. Programming Microsoft Visual Basic- Francesco Balenda, WP Publications and Distributors. 2. Visual Basic 6-Gary Cronell, TataMcGraw Hill Publishing Compnay Ltd.

3. Visual Basic 6 – How to Program, H.M.Deitel., P.J .Deital and T.R.Nieto

Prepared by Dr.N.SHANMUGAVADIVU