Using Option Button and Checkbox Control

1- Option Button Controls:

Option Button controls are also known as radio buttons because of their shape. You always use Option Button controls in a group of two or more because their purpose is to offer a number of mutually exclusive choices. Anytime you click on a button in the group, it switches to a selected state and all the other controls in the group become unselected.

You set an Option Button control's Caption property to a meaningful string, and if you want you can change its Alignment property to make the control right aligned. If the control is the one in its group that's in the selected state, you also set its Value property to True. (The Option Button’s Value property is a Boolean value because only two states are possible.) Value is the default property for this control. At run time, you typically query the control's Value property to learn which button in its group has been selected. Let's say you have three Option Button controls, named opt10, opt100, and opt1000. You can test which one has been selected by the user as follows:

```
If opt10.Value=True Then
Y=X*10
Else If opt100.Value=True Then
Y=X*100
Else If opt1000.Value=True Then
Y=X*1000
End If
```

Strictly speaking, you can avoid the test for the last OptionButton control in its group because all choices are supposed to be mutually exclusive. But the approach I just showed you increases the code's readability.

A group of Option Button controls is often hosted in a Frame control. This is necessary when there are other groups of Option Button controls on the form. As far as Visual Basic is concerned, all the Option Button controls on a form's surface belong to the same group of mutually exclusive selections, even if the controls are placed at the opposite corners of the window. Actually, you can group your controls within any control that can work as a container—Picture Box, for example—but Frame controls are often the most reasonable choice.

2- CheckBox Control

The Checkbox control is similar to the Option Boutton, except that a list of choices can be made using check boxes where you cannot choose more than one selection using an Option Button. By ticking the Checkbox the value is set to True. This control can also be grayed when the state of the Checkbox is unavailable, but you must manage that state through code.

The only important event for Checkbox controls is the Click event, which fires when either the user or the code changes the state of the control. In many cases, you don't need to write code to handle this event. Instead, you just query the control's Value property when your code needs to process user choices. You usually write code in a Checkbox control's Click event when it affects the state of other controls. For example, if the user clears a check box, you might need to disable one or more controls on the form and re enable them when the user clicks on the check box again.

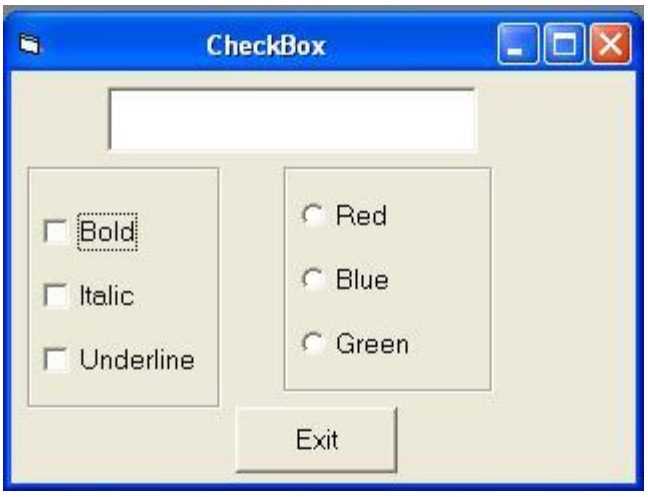
Example 1 The following example illustrates the use of CheckBox control and Option Button.

* Open a new Project and save the Form as CheckBox.frm and save the Project as CheckBox.vbp

* Design the Form as shown below

*

Object	Property	Setting
Form	Caption Name	CheckBox Frm
CheckBox	Caption Name	Bold chkBold
CheckBox	Caption Name	Italic chkItalic
CheckBox	Caption Name	Underline chkUnderline
OptionButton	Caption Name	Red optRed
OptionButton	Caption Name	Blue optBlue
OptionButton	Caption Name	Green optGreen
TextBox	Name Text	Txt1 (empty)
CommandButton	Caption Name	Exit cmdExit
Frame1	Caption	Empty
Frame2	Caption	Empty



Solution: Following code is typed in the Click() events of the CheckBoxes

```
Private Sub chkBold_Click()
If chkBold.Value = 1 Then
  Txt1.FontBold = True
Else
  Txt1.FontBold = False
End If
End Sub
```

```
Private Sub chkItalic_Click()
If chkItalic.Value = 1 Then
  Txt1.FontItalic = True
Else
  Txt1.FontItalic = False
End If
End Sub
```

```
Private Sub chkUnderline_Click()
If chkUnderline.Value = 1 Then
  Txt1.FontUnderline = True
Else
  Txt1.FontUnderline = False
End If
End Sub
```

Following code is typed in the Click() events of the OptionButtons

```
Private Sub optRed_Click()
  Txt1.ForeColor = vbRed
End Sub
```

```
Private Sub optBlue_Click()
  Txt1.ForeColor = vbBlue
End Sub
```

```
Private Sub optGreen_Click()
  Txt1.ForeColor = vbGreen
End Sub
```

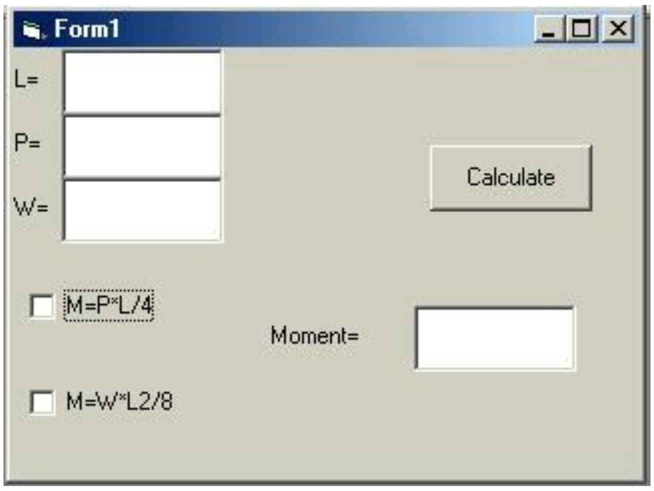
To terminate the program following code is typed in the Click() event of the Exit button

```
Private Sub cmdExit_Click()
  End
End Sub
```

Example 2: For a simply supported beam subjected to a uniform load (W) on the length of span (L) and a concentrated load (P) on a mid span (L/2). When the user click checkbox1, enter the value of (P) and display the value of bending moment (Mom) in a separate text box, when the user click on the checkbox2, enter the value of (W) and display the value of bending moment (Mom) in a separate textbox. Write a program in a separate command button (Calculate) to find the value of (Mom) at mid span of beam subjected to (W) or (P) or both of them.

Solution:

```
Private Sub Command1_Click()
Private Sub Command1_Click()
Dim l, p, w, mom1, mom2,
mom l = Val(Text1.Text)
If Check1.Value = 1 Then
p = Val(Text2.Text)
mom = p * l /
4:mom=mom1 Text4.Text =
Str(mom) Else
Text4.Text = ""
End If
If Check2.Value = 1 Then
w = Val(Text3.Text)
mom = w * l ^ 2 / 8: mom2=mom
Text4.Text = Str(mom2)
Else
End If
If Check1.Value = 1 And Check2.Value = 1 Then
mom = mom1 + mom2
Text4.Text =
Str(mom) End If
End Sub
```


Form1
L=
P=
W=
Calculate
M=P*L/4
Moment=
M=W*L2/8