Extended Control **Sample Help**

Sample Description: <u>Extended Control</u>

Points of Interest
> **<u>How Objects are Extended</u>**
> **<u>Restricting Number Input in TextBoxes</u>**
> **<u>Automatically Converting the Case of Input Characters</u>**

Control
> TextBox

For Help on Help, Press F1

Extended Control

The Extended control sample illustrates how new objects may be copied from an existing object, such as TextBox, and have their functionality specialized through the addition of new properties, methods, and events.   Controls such as this may be thought of as "Extended" controls.   The technique for extending the functionality of base objects may also be referred to as "Copy & Specialize."

How Objects are Extended

There are two techniques for extending the functionality of an object.   The first technique that may be used is to create a standard TextBox on a form, then select it and click the "Abstract Object" icon on Envelop's main tool bar.   A dialog will appear and prompt you to enter a name for the extended control.   What actually happens is that a copy of the base object TextBox   is made.   To create additional copies of this control on your form, simply select the control then press the Ctrl+C key combination or select the Object|Quick Copy menu command.

The second technique for abstracting an object is to click on the object in the Object Viewer, then select the Object|Copy Object menu item from Envelop's menubar.   A dialog will be displayed for you to name the new object to be copied.   After clicking the OK button, the new object will appear below the object from which it was copied.   The newly created object is said to have been "derived" from the parent object.   Another way to think of this is that the new object is a "child" of the parent object and will inherit all the characteristics (methods, properties, events) from the parent object.

Once a new object has been created in the Object Viewer, you must drag and drop this object onto the control palette over the reuse icon, then click the icon and drag an instance of the new object onto your form.

Restricting Number Input in TextBoxes

The three textboxes in the sample have specialized KeyPress event handlers.   An event handler may also be referred to as a "method."   This is basically program code that is triggered when its corresponding event occurs.

Each time a key is pressed, the KeyPress event is triggered and a character key value (i.e., keyAscii) is passed to this event.   In the Sub procedure below, if the Backspace key is pressed, the procedure is automatically passed through, since the Exit Sub prevents the code to filter keyboard from ever being encountered.

```
Sub KeyPress(keyAscii As Integer)
    ' If we are entering a backspace, allow it
    If keyAscii = 8 Then Exit Sub
    ' Check to see if we have reached the maximum number of characters
    If Len(Text) = MaxLength Then keyAscii = 0
    ' Check to see if a valid integer or a space has been entered
    Select Case keyAscii
        Case 48 To 57
        Case 32
        Case Else
          keyAscii = 0
    End Select
End Sub
```

The next test is to see if the number of character in the box exceeds the maximum allowed characters specified by the MaxLength property. In the sample, only a   maximum of 10 characters is allowed.

Finally, if the keyboard key that was pressed was a number   from 0-9 (keyAscii 48-57), the input is allowed.   A space character (keyAscii 32) is also permitted. Any other keys are filtered out by setting the keyAscii value to 0.

Automatically Converting the Case of Input Characters

Two textboxes in this sample are for alpha-character input.   Their KeyPress event is used as a filter to make sure that characters a-z or A-Z are entered.   In the sample code below, you can see how the case conversion is handled.   In this method, if the characters are lower case (keyAscii 97-122), they are automatically converted to upper case.

```
Sub KeyPress(keyAscii As Integer)
     ' If we are entering a backspace, allow it
     If keyAscii = 8 Then Exit Sub
     ' Check to see if we have reached the maximum number of characters
     If Len(Text) = MaxChars Then keyAscii = 0
     ' Check to see if a valid integer or a space has been entered
     Select Case keyAscii
         Case 65 To 90 ' upper case characters
         Case 97 To 122 ' lower case characters
           ' Need to convert keyAscii to Upper Case
           Dim char As String
           char = UCase(Chr(keyAscii))
           keyAscii = Asc(char)
         Case 32 ' backspace
         Case Else
           keyAscii = 0
     End Select
End Sub
```

The LCase, Chr, and Asc commands are used to handle the conversion of lower case characters to upper case characters.   First, the keyAscii value is converted to a character value with the Chr command.   The UCase command is used to convert the character to upper case, and the Asc command is used to convert it back to ANSI code.