***Keyboard* Sample Help**

Sample Description: Keyboard

Points of Interest

- About Keyboard Events
- Identifying Keyboard Characters
- Manually Clearing a ListBox
- Automatically Clearing a ListBox
- ANSI Code Chart

Controls

- TextBox
- ListBox
- Button
- CheckBox

For Help on Help, Press F1

-Keyboard

The Keyboard application is a visual demonstration of the various keyboard events. The application uses several event procedures to display information about the generated events in a ListBox on the form. The ListBox may quickly overflow since a great deal of information about the events are recorded. Once the ListBox overflows, a scrollbar will appear automatically for you to review the keyboard events.

A **Manual Clear** button is provided to clear the contents of the ListBox. An **Automatic Clear** CheckBox is provided to automatically clear the contents of the ListBox when it is about to overflow. Clicking the Automatic Clear CheckBox to True, will display an "X" and temporarily disable the Manual Clear button.

Event information that will be displayed includes all KeyDown, KeyPress, and KeyUp events. Each event will display information about the character code that was struck and the shift key code associated with the event.

Identifying Keyboard Characters

This application identifies keyboard characters using the following KeyPress event.

```
Sub txtKeyboardEntry_KeyPress(keyAscii As Integer)
  Dim action As String

  action = "KeyPress - keyAscii is: " & keyAscii & " Character is: [" & Chr(keyAscii) & "]"
  lstKeyboard.AddItem action
  ' Don't allow anything to come into the box
  keyAscii = 0
End Sub
```

As character keys are pressed on the keyboard, the **KeyPress** event is triggered and the program code shown above is executed. The program code converts the ANSI character code to a character with **Chr** function. This information is then added to the Listbox on the form.

Manually Clearing a ListBox

In this application, a **Manual Clear** button contains the following program code:

```
Sub btnClear_Click()
  lstKeyboard.Clear
  txtKeyboardEntry.SetFocus
End Sub
```

The ListBox's standard **Clear** method is used to clear the contents of the ListBox when the Manual Clear button is clicked. Once the list is cleared, the insertion point is placed back in the keyboard TextBox with the **SetFocus** method. This is necessary since the keyboard events are defined on this textbox.

Automatically Clearing a ListBox

The **Automatic Clear** CheckBox executes the following Click method:

```
Sub chkAutoClear_Click()
  If chkAutoClear.Value == 0 Then
    btnClear.Enabled = "True"
  Else
    btnClear.Enabled = "False"
  End If

  txtKeyboardEntry.SetFocus
End Sub
```

When the CheckBox is clicked to True, its **Value** property is set to "1" and the Manual Clear button is disabled. This prevents the user from automatically clearing the ListBox contents when the Automatic Clear method is activated. When keys on the keyboard are pressed, the following **KeyUp** event method is executed:

```
Sub txtKeyboardEntry_KeyUp(keyCode As Integer, ByVal shift As Integer)
  Dim action As String

  action = "KeyUp - keyCode is: " & keyCode & " Shift parameter is: " & shift
  lstKeyboard.AddItem action

  ' Check to see if the autoclear is on
  If chkAutoClear.Value == 1 Then
    If lstKeyboard.ListCount > 18 Then
      lstKeyboard.Clear
    End If
  End If

End Sub
```

An If test is performed to determine if the value of the CheckBox is True, or equal to "1." When the CheckBox is toggled on, the program code uses the **ListCount** method to determine the number of entries in the ListBox. If the Count is greater than 18, the **Clear** method is executed and the ListBox contents is cleared.

ANSI Code Chart

The following table lists the values that represent alphanumeric characters for the Envelop interpreter. The table shows the basis of the character code of the American National Standards Institute (ANSI). This code has been adapted by Microsoft for use with Microsoft Windows, so it contains some elements of the all-encompassing ANSI code as well as some elements of ASCII (American Standard Code for Information Interchange) on which the ANSI code is partly based.

Understanding ANSI and ASCII Character Sets

The original ASCII character set was limited to numeric values in the range of 1 to 127. Later, an extended ASCII character set added values 128 to 255. Envelop's ANSI character set is a superset of the original ASCII character set; values 1 to 127 define the same characters in both ASCII and ANSI. The characters associated with the values from 128 to 255, however, differ between ANSI and Extended ASCII.

Not all numeric values correlate to an ANSI character that can be displayed. Some values produce special effects. For example, Chr(9) produces a Tab and Chr(10), a line feed. Not all character values technically defined in the ANSI set are supported by Windows. The chart below also does not contain all Envelop supported characters.

Character	**Hex Code**	**ANSI Code**
Backspace	&H08	8
Tab	&H09	9
Line Feed	&H0A	10
Carriage Return	&HOD	13
Space	&H20	32
!	&H21	33
"	&H22	34
#	&H23	35
$	&H24	36
%	&H25	37
&	&H26	38
'	&H27	39
(	&H28	40
)	&H29	41
*	&H2A	42
+	&H2B	43
,	&H2C	44
-	&H2D	45
.	&H2E	46
/	&H2F	47
0	&H30	48
1	&H31	49
2	&H32	50
3	&H33	51
4	&H34	52
5	&H35	53
6	&H36	54
7	&H37	55
8	&H38	56
9	&H39	57
:	&H3A	58
;	&H3B	59
<	&H3C	60
=	&H3D	61
>	&H3E	62
?	&H3F	63
@	&H40	64
A	&H41	65
B	&H42	66
C	&H43	67
D	&H44	68
E	&H45	69
F	&H46	70
G	&H47	71
H	&H48	72
I	&H49	73
J	&H4A	74
K	&H4B	75
L	&H4C	76
M	&H4D	77
N	&H4E	78
O	&H4F	79

P	&H50	80
Q	&H51	81
R	&H52	82
S	&H53	83
T	&H54	84
U	&H55	85
V	&H56	86
W	&H57	87
X	&H58	88
Y	&H59	89
Z	&H5A	90
[	&H5B	91
\	&H5C	92
]	&H5D	93
^	&H5E	94
_	&H5F	95
`	&H60	96
a	&H61	97
b	&H62	98
c	&H63	99
d	&H64	100
e	&H65	101
f	&H66	102
g	&H67	103
h	&H68	104
I	&H69	105
j	&H6A	106
k	&H6B	107
l	&H6C	108
m	&H6D	109
n	&H6E	110
o	&H6F	111
p	&H70	112
q	&H71	113
r	&H72	114
s	&H73	115
t	&H74	116
u	&H75	117
v	&H76	118
w	&H77	119
x	&H78	120
y	&H79	121
z	&H7A	122
{	&H7B	123
|	&H7C	124
}	&H7D	125
~	&H7E	126

About Keyboard Events

The KeyDown, KeyPress, and KeyUp events are usually activated when a user presses a key on the keyboard.

KeyDown

For low-level keyboard handling, use the KeyDown event to report the current keyboard status when a key is pressed. The format for the KeyDown event is as follows:

```
Sub txtKeyboardEntry_KeyDown(keyCode As Integer, ByVal shift As Integer)
```

The KeyDown event occurs for the current control (the control that has focus) every time a user presses a key. This does include the "Shift," "Ctrl," or "Alt," keys. By using the KeyDown event, the program can react to function keys 1 through @, and unique key combinations like "Shift," "Ctrl" key. The program can process these keys because it can handle all keys, not just the standard character keys.

To inform the program of which keys being pressed triggered the event, this event provides the **keyCode** and **shift** arguments. The keyCode argument uniquely identifies the key that is pressed by supplying a unique number. The shift argument is an integer variable indicating the status of three special keys, "Shift", "Alt," and "Ctrl" at the time the KeyDown event was called. Each of these keys keys is assigned a unique value: 1 for "Shift," 2 for "Ctrl," and 4 for "Alt." Pressing any of these keys, causes its value to be added to the shift argument.

KeyPress

The program uses the KeyPress event to intercept ASCII keystrokes when the event's control has the focus. As the keys are intercepted, the program can review the user's input on a byte by byte basis. The KeyPress event format is as follows:

```
Sub txtKeyboardEntry_KeyPress(keyAscii As Integer)
```

The KeyPress event can be useful for validating data input and alerting the user as soon as an invalid character is entered. This event can also be used to limit the type of characters that may be entered in a box. For example, the following program code for a TextBox will accept only numbers 0-9 and the period character as input. The backspace character can also be entered. See the ANSI Code section for other character codes.

```
Sub TextBox1_KeyPress(keyAscii As Integer)
  Select Case keyAscii
    Case 8                          ' Backspace
    Case 46             ' Period
    Case 48 To 57       ' Numbers 0-9
    Case Else
      keyAscii = 0      ' No Character Input
  End Select
End Sub
```

The control with the focus receives this event every time the user presses a key that corresponds to a valid ASCII character. If the value of **keyAscii** is modified within this event, the modification is passed on to the control. In the example shown above, the keyAscii value is set to 0 (zero) if it is not picked up by one of the Case statements. The example below will automatically convert lowercase characters to uppercase.

```
Sub TextBox1_KeyPress(keyAscii As Integer)
  Dim lower_char As String
  Dim upper_char As String

  Select Case keyAscii
    Case 97 To 122 ' Lower-case characters a-z
      lower_char = Chr(keyAscii)
      upper_char = UCase(lower_char)
      keyAscii = Asc(upper_char)
  End Select
End Sub
```

KeyUp

For low-level keyboard handling, use the KeyUp event to report the current keyboard status when a key is released and this event's control has the focus. The format for the KeyUp event is as follows:

```
Sub txtKeyboardEntry_KeyUp(keyCode As Integer, ByVal shift As Integer)
```

The KeyUp event complements the KeyDown event, since each time a user releases a pressed key on the keyboard, including the "Shift," "Ctrl," and "Alt" keys, the current object (that has the focus) receives this event.