Reviewing the Stylebits Parameters

The four parameters of stylebits are AddBit, RemoveBit, AddExtendedBit, RemoveExtendedBit. For a review of these four parameters, and an introduction to Stylebits in general, please view Stylebits - Windows.

Listboxes and Colors

The native listbox is black text against a white background. To change the background color to match the default color of the user's scheme, include

```
ListboxColor$ = "Buttonface"
```

prior to defining that listbox. Change the ListboxColor$ to white (or any other valid color) for subsequent listboxes if desired. To change the color of the text, use ForegroundColor$ before opening the window. Only one color may be assigned to all controls with ForegroundColor$.

Stylebits, Listboxes and Borders

The default border is single edged. Techniques to change the border of a listbox are similar to changing the border of a window, textbox or any other control.

- ***Borderless***

```
ListboxColor$ = "Buttonface"
Stylebits #style.lb1, 0, _WS_BORDER, 0, _WS_EX_CLIENTEDGE
```

- ***Sunken Border***

```
Stylebits #style.lb2, 0, 0, _WS_EX_CLIENTEDGE or _WS_EX_STATICEDGE, 0
```

- ***Raised Border***

```
Stylebits #style.lb2, 0, 0, _WS_EX_DLGMODALFRAME, 0
```

- ***Resizable Border***

```
Stylebits #style.lb2, _WS_THICKFRAME, 0, 0, 0
```

Stylebits, Listboxes and Scrollbars

The default scrollbar lies on the right side of the listbox. When the listbox is tall enough to contain all the items, this scrollbar shows in the disabled state. Using the stylebits _WS_HSCROLL, _WS_VSCROLL, _WS_EX_LEFTSCROLLBAR, and _WS_EX_RIGHTSCROLLBAR, you can hide and even reposition the scroll bar.

- ***Add Horizontal Scrollbar and Remove Vertical Scrollbar***

```
Stylebits #style.lb1, _WS_HSCROLL, _WS_VSCROLL, 0, 0
```

- ***Add Horizontal Position Scrollbar on Left***

```
Stylebits #style.lb2, 0, 0, _WS_EX_LEFTSCROLLBAR, 0
```

- ***Remove Default Disabled Listbar***

```
Stylebits #style.lb3, 0, _LBS_DISABLENOSCROLL, 0, 0
```

Stylebits, Listboxes and Columns

The native listbox displays a single, scrolling column. The stylebits _LB_MULTICOLUMN allows the listbox to display more than one column. The number of columns is NOT determined by assigning a value. Rather, the number of columns is determined by the height of the listbox. Only vertically visible items are assigned to each column. By carefully controlling the height of the listbox, thus controlling the number of vertically visible listed items, the programmer can control the number of columns. The width of the column has no effect upon the number of columns. If not all columns will be visible, assign the stylebits _WS_HSCROLL as well.

- ***Multicolumn Listbox with a Horizontal Scrollbar***

```
Stylebits #style.lb3, _LBS_MULTICOLUMN or _WS_HSCROLL, 0,0, 0
```

Stylebits, Listboxes and Selection

A selection in the native listbox is chosen by double left - clicking. This highlights that item. Only one selection can in the selected state. That selection is retrieved using either the selection? or the selectionindex? command. Stylebits allow more than one item to be in the selected state at any given time.

- ***Multiple Single Selections*** - String selection is toggled each time the user clicks or double-clicks the string. Any number of strings can be selected.

```
#style.lb2, _LBS_MULTIPLESEL, 0, 0, 0
```

- ***Extended Selection*** - The user can select multiple items using the SHIFT key and the mouse or special key combinations. Ctrl - Mouse Click selects multiple single items. Shift - Mouse Click selects a range of items.

```
Stylebits #style.lb3, _LBS_EXTENDEDSEL, 0, 0, 0
```

- ***No Selection*** - Specifies that the list box contains items that can be viewed but not selected.

```
Stylebits #style.lb1, _LBS_NOSEL, 0, 0, 0
```

Stylebits and API Calls

Stylebits do allow multiple and extended selections in listboxes, but an API call (SendMessageA) is required to retrieve those selections. _LB_SETSEL is used to select one, multiple, or all items. By changing the state parameter, _LB_SETSEL is also used to clear one, multiple, or all items. The number of selected items is obtained with _LB_GETSELCOUNT. Because the Listbox sees the items beginning with 0, it may be necessary to decrement the item number by 1.

```
'Setting Multiple Selections and Getting Multiple Selections
'with SendMessageA API calls
'A special Thank You to Bill Beasley for getting this code started

    Dim itemArray$(20)
    For i = 1 to 20
        itemArray$(i) = "Item Number ";Space$(i < 10);Str$(i)
    Next i
    nullParameter = 0 'Used as filler when parameter irrelevant

    Nomainwin
    WindowWidth = 400
    WindowHeight = 360

    UpperLeftX = Int((DisplayWidth - WindowWidth)/2)
    UpperLeftY = Int((DisplayHeight - WindowHeight)/2)

    Listbox #main.lb, itemArray$(), [selectItem], 20, 20, 140, 200
    '_LBS_MULTIPLESEL = Multiple Single Selections using Left Click
    Stylebits #main.lb, _LBS_MULTIPLESEL, 0, 0, 0
```

```
'_LBS_EXTENDEDSEL = Multiple Single Selections using Ctrl - Left Click
 as well as
    'selection of a range of items using Shift - Left Click
    'Stylebits #main.lb, _LBS_EXTENDEDSEL, 0, 0, 0
    Statictext #main.st, "", 20, 230, 350, 100
    Button #main.b1, "Count Items Selected"
, [nItemsSelected], UL, 190, 40, 170, 32
    Button #main.b2, "List Items Selected"
, [listItemsSelected], UL, 190, 85, 170, 32
    Button #main.b3, "Select All Items"
, [selectAllItems], UL, 190, 130, 170, 32
    Button #main.b4, "Clear All Items"
, [clearAllItems], UL, 190, 175, 170, 32

    Open "Listbox Multi Select" for Window as #main
    #main "Trapclose [endDemo]"
    #main "Font Times_New_Roman 12 Bold"
    hLB = hwnd(#main.lb)

'Select some items
    For i = 1 to 20 step 3
        null = SendMessageA(hLB, _LB_SETSEL, 1, i)
    Next i
    #main.st "Every 3rd Item is Selected"

    Wait

[selectItem]
'No action necessary
    Wait

[nItemsSelected]
    nItemsSelected = SendMessageA(hLB, _LB_GETSELCOUNT, nullParameter,
 nullParameter) 'no relevance (par1), (par2)
    #main.st "There are ";nItemsSelected;" items selected."
    Wait

[listItemsSelected]
    itemsSel$ = ""
    For i = 1 to 20


  isSel
= SendMessageA(h
LB, _LB_GETSEL, i - 1, nullParam
```

```
eter) 'i - 1 = Item Number (par1), no relevance (par2)
        If isSel Then
            itemsSel$ = itemsSel$;itemArray$(i);", "
        End If
    Next i
    itemsSel$ = Left$(itemsSel$, Len(itemsSel$) - 2)
    #main.st itemsSel$
    Wait

[selectAllItems]
    allSel = SendMes
sageA(hLB, _LB_SETSEL, 1, -1)
' 1 (par1) = Flag to Set, -1 (par2) = ALL
    Wait

[clearAllItems]
    allClear = Sen
dMessageA(hLB, _LB_SETSEL, 0, -1)
'0 (par1) = Flag to Clear, -1 (par2) = ALL
    Wait

[endDemo]
    Close #main
    End

Function SendMessageA(hW, msg, par1, par2)
    CallDLL #user32, "SendMessageA",_
        hW as Ulong, _ 'Handle of the control (listbox)
        msg as Long, _ 'Stylebits (Windows Constant)
        par1 as Long, _ 'Parameter 1 (sometimes irrelevant)
        par2 as Long, _ 'Parameter 2 (sometimes irrelevant)
        SendMessageA as long 'Return Value, 1 = success
End Function
```

These are just some examples of what you can do with stylebits and listboxes. With experimentation, you may find more.

A List of Stylebits

You can get a list of all dwStyles and dwExStyles available with the Stylebits command at the MSDN Library - List Box Styles.