Microsoft Access Multiselect List Box Control
Version 1.0

Andrew R. Miller (andrewm@microsoft.com)
April 28, 1994

MSLB.OCX is an OLE Custom Control designed to be used with Microsoft Access 2.0. The control provides a multiselect list box much like the one found in Visual Basic. This control provides the following features:

* Methods, Properties and Events similar to VB List box control
* List box can be single select, simple multiselect, or extended multiselect.
* AfterStartup event similar to the Data Outline Control

Installation

NOTE: You must have OC1016.DLL on your system for this control to work properly.   This file is included in the ADT (Access Developer's Toolkit).

1) Copy MSLB.OCX to your Access directory.
2) Copy OC1016.DLL to your \WINDOWS\SYSTEM directory.
3) From ProgMan, run MSAREG.EXE C:\WINDOWS\SYSTEM\OC1016.DLL
4) Start Microsoft Access
5) New Form.
6) Choose EDIT / INSERT OBJECT from the menu.
7) Click on Controls
8) Click Add New
9) Select MSLB.OCX and hit OK.

NOTE: Steps 2 and 3 are not necessary if you have recently installed the ADT.   The ADT installs OC1016.DLL for you.

Using the Control

The multiselect list box was designed to be as close to a VB list box as possible. If youve ever used a VB list box, then this list box should be no problem for you. If youre only familiar with Access listboxes, you may run into a few problems.

The biggest difference between a standard VB list box and an Access list box is that an Access list box is data drivenin VB you have to add each row to the list box through code. The second biggest difference is that Access list boxes are typically multicolumn, whereas a VB list box is not multicolumn without a bit of work. See Creating a Multicolumn Multiselect List Box for more information.

One other note: you will not be able to initialize the control in the Form_Load event, since the control is not completely constructed when this event is fired. Instead, initialize the control in the controls AfterStartup event, which should fired right after the Form_Load event. For you Data Outline Control users, this AfterStartup event functions exactly like the Data Outline Controls AfterStartup event.

Properties

BackColor - Back color

Columns - Number of columns in multicolumn (snaking) listbox

Font - Font of the listbox

hWnd - Handle to the listbox

ItemData(nIndex) - Array of 32 bit values assoc. with items in list box

List(nIndex) - Array of items in the list box

ListCount - Count of items in the listbox

ListIndex - Currently selected item

MultiSelect - MultipleSelect behavior

NewIndex - The index of the item that was just added

Selected(nIndex) - Select/Unselect status for each item in list (multiselect only)

Sorted - If true, added items will be inserted sorted

Text - The text of the currently selected item

TopIndex - Index of item that is at the top of the list

Events

AfterStartup - Fired after the list box has been initialized

Click - Fired when the user clicks the list box

DblClick - Fired when the user dblclicks the list box

KeyDown - Fired when the user presses a key

KeyUp - Fired when the user releases a key

KeyPress - Fired in between KeyDown and KeyUp

MouseDown - Fired when the user presses the mouse button

MouseMove - Fired when the user moves the mouse button

MouseUp - Fired when the user releases the mouse button

Methods

AddItem stItem$, nIndex% - Add/Insert an item to the listbox.

RemoveItem nIndex% - Remove an item from the listbox.

Clear - Clear all items from the listbox.

Creating a Multicolumn Multiselect List Box

Windows does not support multicolumn list boxes in the sense that Access does. A multicolumn list box in Windows is a snaking list box. Since this control is based on a standard Windows list box, this is the behavior that you will see when you set Columns > 1. This is also the behavior that you'll get in VB.

Still, there is a way to simulate an Access multicolumn listbox, but its a not as flexible as the Access model. You have to set tab stops within the control, and then separate the data in each row with Chr$(9) (tabs). See How to Set Tab Stops in a List Box in Visual Basic for more information. If you cant get it working, send me some email.

Why havent I implemented the Access-type multicolumn listbox? One, its quite a bit of work. Two, doing so would create a control that is different from both VB and Access in some way. Right now, the control is very, very close to VBs implementation of a listbox. It would be nice to stay that way, if possible. If you have ideas on maintaining consistency and adding the multicolumn functionality, let me know.

Quickly Determining the Selected Items in a Multiselect List Box

If you want to determine the selected items in a multi-select listbox, you can walk the .Selected() property array. However, on a large listbox, this could take some time. To get around this problem, you can use the following code:

```
Declare Function SendMessage Lib "User" (ByVal hWnd As Integer, ByVal wMsg As Integer, ByVal
wParam As Integer, lParam As Any) As Long

Global Const WM_USER = &H400
Global Const LB_GETSELCOUNT = (WM_USER + 17)
Global Const LB_GETSELITEMS = (WM_USER + 18)


'-----------------------------------------------------------------------
' QuickMultiSelectCheck
'
'   Quickly figures out what items in a multi-select listbox are
'   actually checked.  This is done through the Windows API.
'
'   Parameters:
'
'       lst - The multi-select listbox control you wish to inspect
'-----------------------------------------------------------------------
Sub QuickMultiSelectCheck (lst As Control)

    Dim obj As Object
    Dim cItems As Integer
    Static rgiSel() As Integer
    Dim st As String
    Dim i As Integer

    ' Figure out how many items are actually selected.

    Set obj = lst.Object
    cItems = CInt(SendMessage(obj.Hwnd, LB_GETSELCOUNT, 0, ByVal 0&))

    If (cItems <= 0) Then
        MsgBox "Sorry, no items selected"
        Exit Sub
    End If

    ' Pull those items into an array.

    ReDim rgiSel(0 To cItems - 1)
    cItems = CInt(SendMessage(obj.Hwnd, LB_GETSELITEMS, cItems, rgiSel(0)))
    For i = 0 To cItems - 1
        st = st & "Item #" & rgiSel(i) & Chr$(13) & Chr$(10)
    Next i

    ' Display a message box containing the selected items.

    MsgBox Format$(cItems) & " Items:" & Chr$(13) & Chr$(10) & Chr$(13) & Chr$(10) & st

End Sub
```

Samples

If youre looking for some ideas on how to use the control, check out MSLB.MDB.

Notes

This control is functional, but there may be problems.   If you run into any snags, Id like to know about them.

How to Set Tab Stops in a List Box in Visual Basic

PSS ID Number: Q71067
Article last modified on 02-24-1994

2.00 3.00

WINDOWS

The information in this article applies to:

 - Standard and Professional Editions of Microsoft Visual Basic for
   Windows, versions 2.0 and 3.0

SUMMARY

Visual Basic does not have any intrinsic function for creating multiple-column list boxes. To create multiple-column list boxes, you must call a Windows API function to set tab stops within the control. The tab stops create the multiple-column effect.

For this technique to work, the values in the tab stop array must be cumulative. That is, for three successive tabs to occur, you need to load the tab stop array with 100, 150, 200.

MORE INFORMATION

To create the multiple-column effect in list boxes, call the Windows API SendMessage function. After you set the focus to the list box, you must send a message to the window's message queue that will reset the tab stops of the list box. Using the argument LB_SETTABSTOPS as the second parameter to SendMessage will set the desired tab stops for the multicolumn effect based on other arguments to the function. The SendMessage function requires the following parameters to set tab stops:

```
   SendMessage (hWnd%,LB_SETTABSTOPS, wParam%, lparam)
```

where

```
   wParam%   is an integer that specifies the number of tab stops.

   lParam    is a long pointer to the first member of an array
             of integers containing the tab stop position in
             dialog units.
```

A dialog unit is a horizontal or vertical distance. One horizontal dialog unit is equal to 1/4 of the current dialog base-width unit. The dialog base units are computed based on the height and the width of the current system font. The GetDialogBaseUnits function returns the current dialog base units in pixels.) The tab stops must be sorted in increasing order; back tabs are not allowed.

After setting the tab stops with the SendMessage function, return the focus to the control that had the focus before the procedure call. PutFocus is the Alias for the Windows API SetFocus function. The Windows API SetFocus needs to be redefined using the "Alias" keyword because SetFocus is a reserved word within Visual Basic.

Example Code to Create Multicolumn List Box

For example, to create a multiple-column list box in Visual Basic:

1. Start a new project in Visual Basic, and add a list box (List1) to
   Form1.

2. Declare the following Windows API function at the module level or in
   the Global section of your code as follows:

```
   ' Enter the Declare statement on one, single line:
   Declare Function SendMessage Lib "user" (ByVal hwnd As Integer,
      ByVal wMsg As Integer, ByVal wp As Integer, lp As Any) As Long
```

3. Declare the following constants:

```
   Const WM_USER = &H400
   Const LB_SETTABSTOPS = WM_USER + 19
```

4. Add the following code to the Form_Load Sub procedure:

```
   Sub Form_Load ()
      Const LB_SETTABSTOPS = &H400 + 19
      Static tabs(1 To 3) As Integer



      'Set up the array of defined tab stops.
      tabs(1) = 100
      tabs(2) = 50
      tabs(3) = 90



      'Send a message to the message queue.
      retVal& = SendMessage(List1.hWnd, LB_SETTABSTOPS, 3, tabs(1))
   End Sub
```

REFERENCES
==========

"Programming Windows: the Microsoft Guide to Writing Applications for
Windows 3," Charles Petzold, Microsoft Press, 1990

"Microsoft Windows Software Development Kit: Reference Volume 1,"
version 3.0

WINSDK.HLP file shipped with Microsoft Windows 3.0 Software
Development Kit

Additional reference words: 2.00 3.00
KBCategory:
KBSubcategory: PrgCtrlsStd