MLIST.VBX

Multi-Purpose List Box Control for Visual Basic 3.0

Introduction

MLIST is a custom control for managing the way your data appears in a list box. **MLIST** is a property-for-property replacement of the standard List Box control which comes with Visual Basic 3.0.   In addition, **MLIST** has many more features than the normal List Box.   Some of these features are:

- Set up owner draw columnar data in the list box
- Make a drawing region a bitmap or a checkbox
- Control individual line colors in the list box
- Total control over the checked and unchecked states of lines
- Aligned bitmaps
- True MUTLIPLE COLUMN list boxes, with bitmaps and check boxes
- Checking and Selecting a range of line items
- Properties to find closest match and find exact matches in the list box
- 3-D effects
- Vertical and horizontal grids
- Virtual List Boxes
- Properties to aide you in drag and drop
- ScrollMessage event for drawing column headers over scrolling list boxes
- Handle entire drawing of items in your code
- Set individual fonts for individual lines
- Resort entire lists
- Resize list box to fit windows
- Improved string searching
- Variable height lines in the list box

New Features

ZIP File Contents

This ZIP file contains:

File Name	Description
MLIST.VBX	VB version of the MLIST custom control.
PROJECT1.MAK	Sample project file for MLIST.
FORM1.FRM	Form with some simple information.
READ.ME	ASCII text representation of this file.
MLIST.HLP	On-line Help file.
MLIST.BAS	Contains cool reusable procedures that utilizes features of MLIST.
PLUS.BMP	Bitmap used in example
PAGE.BMP	Bitmap used in example
FINDSTR.MAK	FindString and FindStringExact demo
FIND1.FRM	Form for FindString demo
VIRTUAL.MAK	Virtual List Box example
VIRTUAL.FRM	Form for virtual list box example
SCROLL.MAK	Scrolling column header examples
SCROLL.FRM	Form for scrolling example
ITEMPIC.MAK	Project file for ItemPicture example.
ITEMPIC.FRM	Form file for ItemPicture example.
ITMHITE.MAK	Project file for ItemHeight example.
ITMHITE.FRM	Form file for ItemHeight example.

Registration

This Multi-Column List Box is Shareware. It is NOT crippled in anyway. When you download this custom control, you have the same custom control that I am using in my everyday VB programming. As a Shareware contributor, I am counting on the honor and moral fiber of every person who downloads this custom control to do the right thing.

If you like and appreciate (and more importantly, use) this custom control...

1) Send $25.00 to the address listed in the address topic, or
2) Register in the CompuServe SWREG Forum, ID 7286

Registered users may purchase the source...

1) Send $50.00 to the address listed in the address topic, or
2) Register in the SWREG ID TBD.

If you register via CompuServe, the source will be e-mailed to you. Please remember, you may not purchase source unless you are a registered user.

This control is written in C++ and has been compiled using Visual C++ 1.52.

You may now register this control in the SWREG on CompuServe.   Type GO SWREG, ID 7286.

Warranty

Klasman Quality Consulting, Inc. disclaims all warranties, either express or implied, including, but not limited to, implied warranties of merchantibility and fitness for a particular purpose, with regard to the SOFTWARE, the sample applications and the accompanying written materials.

Reaching The Author

Klasman Quality Consulting, Inc can be reached via US Mail at...

13 Vespa Lane

Nashua, NH 03060

You can reach us via E-Mail at the following locations...

CompuServe:

Kevin M. Klasman

70233,1476

INTERNET:

klasman@kqc.mv.com

Bug Fixes

Bug fixes and work-arounds:

v4.60

Feature	**Description**
FindString	MList won't search backwards all the way to 1st (0th) item **Answer:** This has been fixed. The FINDSTR.MAK sample project has been enhanced to demonstrate this fix.
FindString	MList doesn't include the current item in this search. **Answer:** This has been fixed. The FINDSTR.MAK sample project has been enhanced to demonstrate this fix.
PageDown key	MList with ListBoxStyle = 1 - Variable processes the first PageDown keystroke but ignores subsequent PageDown keystrokes. **Answer:** This has been fixed. This appears to be a bug in Windows itself. There is sample on the MSDN CD (search for ODVHLB or LBS_OWNERDRAWVARIABLE) that has the same problem. I have coded a workaround to make MList behave exactly as VB's listbox control.

v4.58

Feature	**Description**

ListRegion	I'm getting GPFs when I have multiple draw regions, but I only add the first region, then go back later and fill in all regions...
	`MList1.AddItem "A"`
	`MList1.ActiveRegion = 2`
	`MList1.ListRegion(3) = "Column 3"`
	Answer: This has been fixed.

Previous versions

Feature	**Description**
ListIndex	When assigning a value to ListIndex, there is no Click or SelChange event triggered as in a regular list box. Also the starting value is 0, not -1. **Answer:** In versions 4.20 and greater, setting ListIndex will generate a SelChange event. Note that in Multiple-Select list boxes, the starting index is 0, where the focus rectangle is, and not -1, like in a normal list box.
SetFocus	When I SetFocus to a list configured with both grid lines on, the first line doesn't redraw its line correctly. **Answer:** Make sure that your column lengths take up the entire list box, even if you are not using the extra space at the end. MList assumes that you are using the entire width of the list box.
ListRegion	When loading information using ListRegion and TextRegion , ListIndex is reset to -1 after first assignment. If multiple regions have to be loaded, I have to cache the ListIndex property in a temporary variable throughout

	the operation and reassign it to ListIndex in order to get back to the same line. **Answer:** This has been fixed.
SendMessage	I'm getting General Protection Faults when sending messages directly to an MList via SendMessage. **Warning:** MList does not have LBS_HASSTRINGS style, which Windows expects for some messages. Do not use SendMessage with an MList.
MultiSelect	MList always reports ListIndex = 0 when a search is conducted with MultiSelect is set to true. **Answer:** This is a related Multiple-Select problem. I appreciate both the input and your patience. It has been fixed.
Sorted	MList seems to have a problem finding things when Sorted is true. **Answer:** Searches, whether sorted or not, always search from the current item in the direction specified by FindDirection property. It will not wrap around.
Click	The Click event works differently from the standard list box which comes with VB **Answer:** I haven't figured out this one. Microsoft called their main notification event, Click, which in all other controls responds to mouse events. Yet, the one in the list box, responds to all events which change the currently selected item, whether that is ListIndex, a mouse, or a key press. Humph. I used the SelChange event to do the same thing, because is more indicative of what is happening to the list box. If someone asks me to, I will provide a property to force the Click event to behave like the SelChange event.
ItemLength	This property returns space in pixels when it is set with twips. **Answer:** OOPS! I do convert it to Pixels, but I return it NOW in twips. There is a small conversion difference, so don't be surprised if you set ItemLength(0) = 1000 and get back 997.
DrawRegions	The more columns I have, the more the clipping region diminished. **Answer:** When setting the number of DrawRegions (columns), MList will evenly divide the current area of the list box into the number of DrawRegions you specify. If you want to use the horizontal scroll bar, then you must reset the ItemLength of the desired columns to extend past the edge of the list box.
DblClick	Double clicking with the right mouse button acts as if I double clicked the currently selected item with the left mouse button. **Answer:** This has been corrected.

Revision History

Revision	Comments
4.60	Change in ownership from McKean Consulting to Klasman Quality Consulting, Inc.; update help; bug fixes.
4.58	Added SortType property; bug fixes.
4.56	Added (About), AutoCheck properties
4.55	Added FindResult, FindIndex, SearchCompare, FindPattern, FindPatternColumn, SortOrder, OwnerCompare, ItemFont and ClickRegion properties.
4.50	Added font support, complete OwnerDraw, DisableDrawing, and NoIntegralHeight properties.
	Added InString, FindResult, FindColumn, and FindColumnString properties.
	Added icon support.
4.30	Changed the way that virtual list boxes determine when they are in the virtual region. I used to use the TopIndex standard property, now I use the thumb position on the scroll bar.
	Added SortColumn and StringCompare properties. These properties allow you sort on different columns as well as specify if the sort is case sensitive or case insensitive.
	Added new properties for Drag and Drop. The first, SelectMode, doesn't allow items to become "de-selected" when dragging. ItemX, ItemY, and Item help you determine which line item is being dragged over or dropped upon.
	Fixed dynamic loading of an MLIST, and calling AddItem "MyString", 0 when there are no items in the list.
	Added ScrollMessage event for scrolling column headers.
4.20	Tightened up the difference between a multi-select list box and a normal list box.
	Added GridColor property; VirtualMessage event, check marks in check boxes,
	Added SelChange event when ListIndex is changed,
	Added SelCheck event for when an item is checked or unchecked.
4.11	Hilited lines will default to the system colors as controlled by the Control Panel. The user may still override using the HiliteForeColor and HiliteBackColor.
	Added the ExtendedSelect property.
4.10	Added 3-D RiseColor and FallColor properties, and SelChange event.
4.02	Removed a bug which caused a GP fault when more than 16 drawing regions were requested by the user.
	Reworked VB2.0 compatibility strategy.
4.01	Removed the 64k limit. MLIST will now allow more than 64k worth of data to be added to it.

Also, quit handling default methods, such as Move (which didn't work), Refresh, and others.

Fixed problem with RemoveItem not deleting lines data. Changing the way MLIST handles strings did the job.

4.00 Added the MultiColumn property and item width.

Added the FindString and FindStringExact properties for searching the list box.

Added the range properties for checking and selecting multiple line items.

3.04 Fixed the bug with setting the Checked property, and then not having the user be able to double click it or click the box to change it back.

Added the AlignBitmap image type and the Alignment property.

3.03 Fixed the index problem with the List default property. If anything, I am guilty of being over enthusiastic. I apologize to those who got the previous version and thought it sucked. Probably because it did.

Added bitmaps and checkboxes as well as the ability to change the color of a specific line (ItemBkColor and ItemForeColor properties.) You can also change the bitmap of a specific line item.

3.02 Fixed a ton of problems. Namely, that page down didn't work. Delete current controls and rebuild to fix this problem.

Added ALL default properties for the standard list box. Tested all of them before uploading.

3.01 Fixed "Bad Index" error message when setting Selected property to True or False.

Added ItemData default property.

Added ListIndex default property.

Fixed an apparent bug in the VB API that was passing right mouse button clicks to the Click event, but not the left mouse button.

3.00 Initial release of the MLIST.VBX Custom Control

Acknowledgments

I would like to thank Robin McKean, for without his vision, skills and determination, this software would not exist. And also for giving me the opportunity and responsibility for moving this software into the future.

Properties

All of the properties supported by MLIST are listed below. Properties that apply *only* to this control, or require special consideration when used with it, are underlined. The other properties are documented in Visual Basic's on-line help and in the Visual Basic *Custom Control Reference* (see the section "Standard Properties, Events, and Methods" Appendix A). MLIST now implements all standard list box properties

(About)	GridColor	MousePointer
ActiveRegion	GridStyle	MultiColumn
AddItemHeight	Height	MultiSelect
Alignment	HelpContextID	Name
AllowFocusRect	HiliteBackColor	NewIndex
AutoCheck	HiliteForeColor	NoIntegralHeight
BackColor	HorizontalGrids	OwnerCompare
BorderStyle	hWnd	OwnerDraw
CheckColor	ImageRegion	Parent
Checked	ImageType	RangeChecked
CheckStyle	Index	RangeEnd
ClickRegion	IndItemHeight	RangeSelected
Columns	InString	RangeStart
DefPicture	Item	Resort
DisableDrawing	ItemBkColor	RiseColor
DragIcon	ItemData	SearchCompare
DragMode	ItemFont	SelCount
DrawFlags	ItemFontBold	Selected
DrawRegions	ItemFontItalic	SelectMode
Enabled	ItemFontName	SetHzScroll
EnableVirtualMsgs	ItemFontSize	SortColumn
ExtendedSelect	ItemFontStrikeThru	Sorted
FallColor	ItemFontUnderline	SortOrder
FindColumn	ItemForeColor	SortType
FindColumnString	ItemHeight	StringCompare
FindDirection	ItemHiliteBackColor	TabIndex

FindIndex	ItemHiliteForeColor	TabStop
FindPattern	ItemLength	Tag
FindPatternColumn	ItemPicture	Text
FindResult	ItemWidth	TextRegion
FindString	ItemX	Top
FindStringExact	ItemY	TopIndex
FontBold	Left	Version
FontItalic	List	VerticalGrids
FontName	ListBoxStyle	VirtualMsgZone
FontSize	ListCount	Visible
FontStrikethru	ListIndex	Width
FontUnderline	ListRegion	
ForeColor	MaskingColor	

Events

All of the events supported by MLIST are listed below. Events that apply *only* to this control, or require special consideration when used with it, are underlined. The other events are documented in Visual Basic's on-line help and in the Visual Basic *Custom Control Reference* (see the section "Standard Properties, Events, and Methods" Appendix A).

Click	KeyUp
CompareItem	LostFocus
DblClick	MouseDown
DragDrop	MouseMove
DragOver	MouseUp
DrawItem	ScrollMessage
GotFocus	SelChange
KeyDown	SelCheck
KeyPress	VirtualMessage

Klasman Quality Consulting, Inc.

With over 17 years combined in the computer industry, ***Klasman Quality Consulting, Inc.,*** brings a wealth of experience in all aspects of software development to the delivery of high quality, user-friendly solutions to customer requirements in a Microsoft Windows environment. We can help you with:

- Custom programming in Visual Basic, including database access using SQL
- Windows Help systems, or other applications of the WinHelp hypertext engine
- Software test automation strategies and tools

With expertise in today's advanced communications technologies, we can tele-commute anywhere!

Call or e-mail us with your needs and let us propose a high quality, cost-effective solution.

Klasman Quality Consulting, Inc.

Kevin M. and Lisa S. Klasman, Owners

13 Vespa Lane

Nashua, NH 03060

Phone:	**(603) 880-7801**
Fax:	**(603) 886-6152**
CompuServe:	**70233,1476**
Internet:	**70233,1476@compuserve.com**

pseudo-property

A pseudo-property is one that appears in the property list of a control, but is used to access some feature instead of actually storing data related to the control.

grayed

The bitmap colors are reversed

range properties

RangeChecked

RangeEnd

RangeSelected

RangeStart

(About) Property

Description

Displays an informative dialog box identifying the current version of the MLIST control at design time. Not available at runtime.

Remarks

The (About) property is a pseudo-property.

Data Type

String

ItemHeight Property

Description

Controls the height of all lines in the list box.

Remarks

It defaults to 195 Twips, the height of the font used by the standard list box. You should adjust this height if you change the font, font size, etc.

This property will not change the height of the lines after the list box has already been loaded. It was not designed to have the height of the items changed dynamically. Set the height of this item at design time, and the font, and your list box will be fine.

Example

You can resize the height of every item by unloading the list box (saving them to a local array), setting this property, then reloading the Mlist. See the sample project ITMHITE.MAK.

Data Type

Long

DrawRegions Property

Description

Determines the number of drawing regions on each line.

Remarks

This is basically equivalent to the number of COLUMNS, but this list box will still scroll like a normal list box.

Setting

The DrawRegions property settings are:

Setting	Description
0	(Default) No regions are specified, thus MList acts like a normal list box.
1	1 region specified, but in essence this is equivalent to 0.
2 to N	The number of regions.

Note When setting this property, the control clears the flags for the draw regions and sets them to left aligned, single line, and centered vertically. Consult the Windows API for DrawText to see exactly what these flags mean.

Data Type

Integer

ItemLength Property

Description

Controls the length of each drawing region. Not available at design time.

Remarks

If you want a drawing region to be a specific length, set that drawing region's length to the desired number of twips.

Usage

```
MList1.DrawRegions = 2
MList.ItemLength(1) = 500
MList.ItemLength(2) = 500

' Both of the following statements
' generate an out of index error
MList1.ItemLength(0) = XXX
MList1.ItemLength(3) = XXX
```

Note It is probably a good idea to set the lengths of all drawing regions when you are changing the defaults.

Data Type

Array of Integers

DrawFlags Property

Description

An indexed property which corresponds to the flags of each drawing region. Not available at design time.

Remarks

If you want a drawing region to be drawn in a specific manner, set that drawing region's flags to the desired settings. For specific flags, look up the DrawText API function in the Windows API Reference.

Usage

In the following example, I set up two drawing regions and set the second drawing region's flags to right justify the text. In this example, the second drawing region displays currency values.

```
Const DT_RIGHT = 2
Const DT_VCENTER = 4
Const DT_SINGLELINE = 32

MList1.DrawRegions = 2
MList1.DrawFlags(2) = DT_RIGHT + DT_VCENTER + DT_SINGLELINE
MList.ItemLength(1) = 500
MList.ItemLength(2) = 500
```

Note If you are planning to include the '&' character in your strings, you should include the DT_NOPREFIX flag to avoid the underline.   I do NOT automatically add this flag for you, as I want you to have total control over the flags.

Data Type

Integer

ActiveRegion Property

Description

Defines the drawing region (or index or column) that the TextRegion and ListRegion properties return. Not available at design time.

Usage

An MList is divided up into three columns; name, account number, and dollar amount. I want to get the account number from both the currently selected line as well as the first item.

```
MList1.ActiveRegion = 2
' Gets 2nd column of currently selected item
aString$ = MList1.TextRegion
' Gets 2nd column of item at index 0
bString$ = MList1.ListRegion(0)
```

Data Type

Integer

TextRegion Property

Description

Gets or sets the value of the ActiveRegion in the currently selected item. Not available at design time.

Remarks

This property corresponds to a one of the DrawRegions of the Text standard property.

Usage

```
MList1.ActiveRegion = 2
' Gets 2nd column of currently selected item
aString$ = MList1.TextRegion
```

Data Type

String

ListRegion Property

Description

This property returns the contents of the ActiveRegion in the item at the specific index. Not available at design time.

Remarks

This property corresponds to one of the DrawRegions of the List standard property.

Usage

```
MList1.ActiveRegion = 2
' Gets 2nd column of item at index 0
bString$ = MList1.ListRegion(0)
```

Data Type

String

DefPicture Property

Description

Determines the default bitmap to be displayed in the ImageRegion when a specific bitmap has not been assigned to that item's ItemPicture property

Remarks

To set this property at run time, you must use the LoadPicture function. For example

```
Mlist1.Picture = LoadPicture(c:\page.bmp)
```

Note See the demo on setting your own DefPicture and ItemPicture properties.

Data Type

Picture

ImageType Property

Description

Determines what type of image is contained in an ImageRegion.

Setting

The ImageType property settings are:

Setting	Description
0	(Default) None
1	Bitmap
2	CheckBox
3	Aligned Bitmap

Remarks

If this property is set to 0, then the ImageRegion property is ignored and only text is displayed. If this image type is valid, then a checkbox or bitmap/icon is displayed in the ImageRegion.

Note When the image type is set to AlignBitmap, the normal drawing regions no longer apply.   Anybody got a problem with that?

Data Type

Integer (enum)

ImageRegion Property

Description

Specifies which region contains the image region.

Remarks

This value should be any number between 1 and the number of DrawRegions. You may not set this property to a value of less than zero or greater then DrawRegions.

Usage

When setting up your DrawRegions, the following is a good example:

```
MList1.DrawRegions = 3
MList1.ImageRegion = 1
' Bitmap or Icon is in here
MList1.ItemLength(1) = 100
MList1.ItemLength(2) = 500
MList1.ItemLength(3) = 750

' Don't leave a space for the ImageRegion
' in your strings
MList1.AddItem "Region2" + Chr$(9) + "Region3"
MList1.AddItem "Region22" + Chr$(9) + "Region33"
```

Note The ImageRegion is ignored when calculating the "piece" of text that goes in that region.

Data Type

Integer

Checked Property

Description

Determines if a Check mark is displayed in an MList with ImageType set to 2 (CheckBox). Not available at design time.

Setting

The Checked property settings are:

Setting	Description
True	(default) The CheckBox is checked.
False	The CheckBox is not checked.

Remarks

This property is very similar to the Selected standard property. In a Multi-Column list box with the ImageType set to 2 (CheckBox), this property will return to you whether or not that line item is checked.

Usage

```
For X% = 0 To MList1.ListCount - 1
    If MList1.Checked(X%) Then
          Debug.Print "I am Checked! "; X%
    Else
          Debug.Print "I am not checked! "; X%
    End If
Next X%
' Check the first item
MList1.Checked(0) = True
```

Data Type

Integer (boolean)

ItemPicture Property

Description

Controls the bitmap for each line item in the list box. Not available at design time.

Remarks

If this item is not set by you, then the MList uses the DefPicture property.

Usage

```
 ' Change the first items picture in the list box
MList1.ItemPicture(0) = Image1.Picture
```

To set this property at run time, you must use the LoadPicture function. For example

```
Mlist1.ItemPicture(0) = LoadPicture(c:\page.bmp)
```

Example

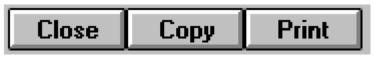
Data Type

Array of Pictures

```
' ItemPicture Example
MList1.DrawRegions = 2
MList1.ImageRegion = 1
MList1.ImageType = 1   ' bitmap
' In reality, you would probably set these picture properties in the
' properties window to be more efficient!
MList1.DefPicture = LoadPicture(app.Path & "\page.bmp")
Image1.Picture = LoadPicture(app.Path & "\plus.bmp")

' set height to display entire picture
'  (Mlist1.ListBoxStyle is set to 1 - Variable in Properties window)
Mlist1.AddItemHeight = Image1.Height

' Bitmap or Icon is in here
MList1.ItemLength(1) = 400
MList1.ItemLength(2) = MList1.Width - MList1.ItemLength(1)
For i = 1 To 20
    MList1.AddItem "MList1" & Str$(i)
Next I
```



```
Sub MList1_Click ()
    MList1.ItemPicture(MList1.ListIndex) = Image1.Picture
End Sub
```

ItemForeColor Property

Description

Controls the foreground color for each line item in the list box. Not available at design time.

Setting

The ItemForeColor property settings are:

Setting	Description
Normal RGB colors	Colors specified by using the RGB or QBColor functions.

Remarks

If this item is not set by you, then the MList uses the default foreground color property.

Usage

```
' Change the first items foreground color to white
MList1.ItemForeColor(0) = RGB(255,255,255)
```

Data Type

Array of Long (color)

ItemBkColor Property

Description

This property is an array of colors which correspond to the background color for each line item in the list box. Not available at design time.

Setting

The ItemBkColor property settings are:

Setting	Description
Normal RGB colors	Colors specified by using the RGB or QBColor functions.

Remarks

If this item is not set by you, then the MList uses the default background color property.

Usage

```
 ' Change the first item's background color to black
MList1.ItemBkColor(0) = RGB(0,0,0)
```

Data Type

Long (color)

Alignment Property

Description

Controls the placement of the bitmap when ImageType is set to 4, AlignBitmap. See the demo for details.

Setting

The Alignment property settings are:

Setting	Description
0	(Default) None
1	Left. Align the bitmap or icon to the left, centered, then the text, centered vertically and left justified.
2	Top. Align the bitmap or icon on top, centered, then the text, centered horizontally and vertically.
3	Right. Align the bitmap or icon to the right, centered, then the text, centered vertically and left justified.
4	Bottom. Align the bitmap or icon on bottom, centered, then the text, centered horizontally and vertically.

Note When ImageType is set to AlignBitmap, the normal drawing regions no longer apply. Anybody got a problem with that?

Data Type

Integer (enum)

MultiColumn Property

Description

Determines whether an MList will scroll horizontally or vertically. Read-only at run time.

Setting

The MultiColumn property settings are:

Setting	Description
True	Items are arranged in snaking columns, filling the first column, then the second column, and so on. The MList scrolls horizontally.
False	(Default) The MList scrolls vertically.

Remarks

A default ItemWidth is provided, but you, the developer, should override this property

Note This property is analogous to the standard list box Columns property, in that it determines which way the control scrolls. It does not, however, set the number of Columns (use the DrawRegions property instead.)

Data Type

Integer (boolean)

ItemWidth Property

Description

The property specifies the width of each line in a multiple column list box. Read-only at run time.

Remarks

The width is set in Twips. The normal drawing regions still apply, but I can't think of a reason why someone would want to divide line items in a column list box into more columns.

Data Type

Integer

FindString Property

Description

Setting this property causes the MList to search, from the current ListIndex, for an item with the closest match . Not available at design time. Write-only at run time.

Remarks

If a match is found, that item is set to the current ListIndex. This is useful for moving the selection through a list box while typing the string in an edit control, like the Search dialog in Help.

This property is the same as FindStringExact, except that it will search for the closest match.

Note The direction of the search is determined by the FindDirection property.

Data Type

String

FindStringExact Property

Description

Setting this property causes the MList to search, from the current ListIndex, for an exact match. Not available at design time. Write-only at run time.

Remarks

If a match is found, that item is set to the current ListIndex. This is useful for moving the selection through a list box while typing the string in an edit control, like the Search dialog in Help.

This property is the same as FindString, except that it will search for an exact match.

Note The direction of the search is determined by the FindDirection property.

Data Type

String

RangeStart Property

Description

This property marks the beginning line item of a range for use with the RangeSelected or RangeChecked. properties. Not available at design time.

Remarks

As is the case with all list boxes and combo boxes in Visual Basic, the offset is 0 based, so the first item is 0.

Usage

The following example selects the first 4 items in an MList.

```
MList1.RangeStart = 0
MList1.RangeEnd = 3
MList1.RangeSelected = True
```

Data Type

Integer

RangeEnd Property

Description

Marks the ending line item for RangeSelected or RangeChecked. Not available at design time.

Remarks

As is the case with all list boxes and combo boxes in Visual Basic, the offset is 0 based, so the first item is 0.

Usage

The following example checks the first 4 items in an MList.

```
MList1.RangeStart = 0
MList1.RangeEnd = 3
MList1.RangeChecked = True
```

Data Type

Integer

RangeSelected Property

Description

Determines if all items in the range RangeStart to RangeEnd are marked as selected in the list box. Not available at design time. Write-only at run time.

Setting

The RangeSelected property settings are:

Setting	Description
True	All items in the range are selected
False	(Default) All items in the range are not selected.

Remarks

As is the case with all list boxes and combo boxes in Visual Basic, the offset is 0 based, so the first item is 0.

Usage

The following example selects the first 4 items in an MList.

```
MList1.RangeStart = 0
MList1.RangeEnd = 3
MList1.RangeSelected = True
```

Note If the list box is not MultiSelect, then this property is ignored.

Data Type

Integer (boolean)

RangeChecked Property

Description

Determines if all items in the range RangeStart to RangeEnd are marked as checked. Not available at design time. Write-only at run time.

Setting

The RangeChecked property settings are:

Setting	Description
True	All items in the range are checked.
False	(Default) All items in the range are not checked.

Remarks

As is the case with all list boxes and combo boxes in Visual Basic, the offset is 0 based, so the first item is 0.

Usage

The following example checks the first 4 items in an MList.

```
MList1.RangeStart = 0
MList1.RangeEnd = 3
MList1.RangeChecked = True
```

Data Type

Integer (boolean)

SetHzScroll Property

Description

Adds a horizontal scroll bar to the MList if a column extends past the right edge of the control. Not available at design time. Write-only at run time.

Setting

The SetHzScroll property settings are:

Setting	Description
False	(Default) No horizontal scroll bar.
True	Displays a horizontal scroll bar, if required.

Remarks

When all is said and done, and you are through setting up your list box, and your columns extend past the displayable area of the list box, setting this property to **True** will cause the MLIST control to add up all of the drawing regions and add a horizontal scroll bar to the list box if one is necessary.

Data Type

Integer (boolean)

HiliteForeColor Property

Description

Determines the default foreground color to use when a line in the list box is hilited.

Setting

The HiliteForeColor property settings are:

Setting	Description
&H00000000&	(Default) Black.
Normal RGB colors	Colors specified by using the Color palette, or by using the RGB or QBColor functions in code.

Remarks

If these colors are set, then the normal colors are ignored, and the HiliteForeColor is used to draw the text, and HiliteBackColor is used to draw the background.

You can force MList to use the default colors by setting this color equal to HiliteBackColor.

Data Type

Long (color)

HiliteBackColor Property

Description

Determines the default background color to use when a line in the list box is hilited.

Setting

The HiliteBackColor property settings are:

Setting	Description
&H00000000&	(Default) Black.
Normal RGB colors	Colors specified by using the Color palette, or by using the RGB or QBColor functions in code.

Remarks

If these colors are set, then the normal colors are ignored, and the HiliteForeColor is used to draw the text, and HiliteBackColor is used to draw the background.

Note You can force MList to use the default colors by setting this color equal to HiliteForeColor.

Data Type

Long (color)

ItemHiliteForeColor Property

Description

Controls the foreground color of hilited individual line items. Not available at design time.

Setting

The ItemHiliteForeColor property settings are:

Setting	Description
Normal RGB colors	Colors specified by using the RGB or QBColor functions.

Remarks

If this item is not set by you, then the MList uses the default hilite foreground color property.

Usage

```
MList1.ItemHiliteForeColor(0) = RGB(0,0,0)
MList1.ItemHiliteBackColor(0) = RGB(255,255,255)
```

Data Type

Array of Longs (color)

ItemHiliteBackColor Property

Description

Controls the background color of hilited individual line items. Not available at design time.

Setting

The ItemHiliteBackColor property settings are:

Setting	Description
Normal RGB colors	Colors specified by using the RGB or QBColor functions.

Remarks

If this item is not set by you, then the MList uses the default hilite background color property.

Usage

```
MList1.ItemHiliteForeColor(0) = RGB(0,0,0)
MList1.ItemHiliteBackColor(0) = RGB(255,255,255)
```

Data Type

Array of Longs (color)

MaskingColor Property

Description

This color is used to mask out colors in your bitmaps when they are included in your MList Box.

Setting

The MaskingColor property settings are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette, or by using the RGB or QBColor functions in code.

Remarks

For example, lets say that you know you are not going to use white in your bitmaps or list boxes. If you set to white all the pixels in your bitmap that you want to be transparent, the natural color of the list box will show through your bitmap in these places. Think of this as the transparent color, like in Icons, except you get to determine what it is.

Data Type

Long (color)

HorizontalGrids Property

Description

Determines if horizontal grid lines are drawn on the MList.

Setting

The HorizontalGrids property settings are:

Setting	Description
True	Horizontal grid lines are drawn.
False	(Default) Horizontal grid lines are not drawn.

Data Type

Integer (boolean)

VerticalGrids Property

Description

This property determines if vertical grid lines are drawn on the MList.

Setting

The VerticalGrids property settings are:

Setting	Description
True	Vertical grid lines are drawn.
False	(Default) Vertical grid lines are not drawn.

Data Type

Integer (boolean)

GridStyle Property

Description

Determines what line style is used to draw the grid lines.

Setting

The GridStyle property settings are:

Setting	Description
0	(Default) Solid
1	Dash
2	Dot
3	Dash Dot

Remarks

This didn't quite turn out the way I wanted, but experiment if you like, and give me some suggestions.

Note This property is ignored if both HorizontalGrids and VerticalGrids are false.

Data Type

Integer (enum)

BorderStyle Property

Description

Determines the border style of the MList.

Setting

The BorderStyle property settings are:

Setting	Description
0	(Default) Normal
1	Raised. The control appears to be above the form (embossed)
2	Inset. The control appears to be sunken into the form.

Data Type

Integer (enum)

RiseColor Property

Description

This is the color of the rising edge of a 3-D Style list box.

Setting

The RiseColor property settings are:

Setting	Description
&H00FFFFFF&	(Default) White.
Normal RGB colors	Colors specified by using the Color palette, or by using the RGB or QBColor functions in code.

Remarks

Visual Basic uses the Microsoft Windows environment RGB scheme for colors.

Data Type

Long (color)

3D Style

The control has BorderStyle set to 1 (Raised) or 2 (Inset).

FallColor Property

Description

Determines the color of the falling edge of the 3-D Style list box.

Setting

The FallColor property settings are:

Setting	Description
&H00808080&	(Default) Dark gray.
Normal RGB colors	Colors specified by using the Color palette, or by using the RGB or QBColor functions in code.

Remarks

Visual Basic uses the Microsoft Windows environment RGB scheme for colors.

Data Type

Long (color)

Version Property

Description

This property was included to insure backwards compatibility. You don't need to be concerned with its use, as it is used internally for different things. Read-only at design time.

ExtendedSelect Property

Description

Determines if multiple items can be selected by using the Ctrl and/or Shift keys and the mouse. Read-only at run time.

Setting

The ExtendedSelect property settings are:

Setting	Description
True	(Default) Enables multiple select.
False	Enables single select.

Remarks

This property causes the MList to be created with the LBS_EXTENDEDSEL style. Shift-clicking an item selects a range of items running from the previous value of ListIndex to the new value of ListIndex. Ctrl-clicking an item selects or deselects it without affecting any others. You can mix Shift-clicks and Ctrl-clicks in the same Extended Selection. See the MS-Windows SDK help for more on extended select.

Notes If ExtendedSelect is **True**, the default value for ListIndex is **0**, not **-1**, as is the case with the standard list box. Otherwise, it is **-1**. This property is ignored if MultiSelect is **False**.

Data Type

Integer (boolean)

GridColor Property

Description

Determines the color of the horizontal and vertical grids.

Setting

The GridColor property settings are:

Setting	Description
&H00000000&	(Default) Black.
&H00C0C0C0&	I prefer Light gray.
Normal RGB colors	Colors specified by using the Color palette, or by using the RGB or QBColor functions in code.

Remarks

Visual Basic uses the Microsoft Windows environment RGB scheme for colors.

Note This property is ignored if both HorizontalGrids and VerticalGrids are false.

Data Type

Long (color)

FindDirection Property

Description

Determines which direction a FindString, FindPattern, FindPatternColumn, or FindStringExact will search the list.

Setting

The FindDirection property settings are:

Setting	Description
0	(Default) Forward. The search will begin at the currently active ListIndex and search to the bottom of the list.
1	Backwards. MList will begin at the current ListIndex and search to the top of the list.

Data Type

Integer (enum)

EnableVirtualMsgs Property

Description

Determines if the MList is a virtual list box.

Setting

The EnableVirtualMsgs property settings are:

Setting	Description
True	(Default) Makes the MList a virtual list box.
False	Makes the MList a normal list box.

Remarks

Setting this property to **True** causes a message to be sent to the VirtualMessage event whenever the user approaches the beginning or end of the list. These messages are sent whether caused by a mouse or keyboard event.

Data Type

Integer (boolean)

VirtualMsgZone Property

Description

Determines when a message is sent to the VirtualMessage event.

Setting

The VirtualMsgZone property settings are:

Setting	Description
0	(Default) Virtual messages are not sent.
>0	Indicates the "zone" at the beginning and end of the list that triggers the event.

Remarks

It indicates the "zone" at the beginning and end of the list that triggers the event. For example, setting this property to 100, causes a VIRTUAL_END message to be sent to VirtualMessage event when the user gets within 100 lines of the end of the list box, and a VIRTUAL_BEGIN message when the user gets within 100 lines of the beginning of the list box. The event is only triggered if the user is heading towards the respective end of the list box.

Note This property is ignored if the EnableVirtualMsgs property is False.

Data Type

Integer

CheckStyle Property

Description

Determines the type of "checkmark" displayed by a checkbox style MList, with the ImageType property set to 2 (CheckBox).

Setting

The CheckStyle property settings are:

Setting	Description
0	(default) The normal "cross/diagonal" check box is used.
1	A check mark is used.

Data Type

Integer (enum)

CheckColor Property

Description

Determines the color of the check mark for an MList with the ImageType property set to 2 (CheckBox).

Setting

The CheckColor property settings are:

Setting	Description
&H00000000&	(Default) Black
Normal RGB colors	Colors specified by using the Color palette, or by using the RGB or QBColor functions in code.

Remarks

Visual Basic uses the Microsoft Windows environment RGB scheme for colors.

Data Type

Long (color)

SortColumn Property

Description

Determines which column is sorted, when the DrawRegions property is greater than 1 and the Sorted property is **True**.

Setting

The SortColumn property settings are:

Setting	Description
0	(Default) Sorts the entire string.
1 to N	Sorts on the specified column. (N = DrawRegions)

Usage

The following example sorts the MList by priority.

```
MList1.DrawRegions = 2
MList1.SortColumn = 2
MList1.AddItem "100" + Chr$(9) + "High"
MList1.AddItem "101" + Chr$(9) + "Critical"
```

Note Only strings added after this property is set will be affected. If this property is changed after the strings are added, you will have to remove all the strings and then re-add them. I thought about doing this myself, but the issues over preserving pictures, drawing information, and all that other stuff made it a little more difficult than I care to tackle at this moment.

Data Type

Integer

SelectMode Property

Description

Determines how selected items in an ExtendedSelect or MultiSelect MList respond to left mouse clicks.

Setting

The SelectMode property settings are:

Setting	Description
0	(Default) Normal. All other selected items (if any) are deselected when the left mouse button is clicked on a selected item.
1	Drag Mode. A selected item ignores a left mouse button click, thus allowing the MouseMove event to fire.

Remarks

This property will help you when you want to drag and drop with a ExtendedSelect or MultiSelect list box. When this property is set to 0 (Normal), after selecting X number of items, if you select a hilited item with the mouse, all the other items are de-selected. This is a problem when trying to drag and drop multiple items from the list. Setting this property to 1 (Drag Mode) causes the left mouse click to be ignored when the item being selected is already hilited. This allows the user to start the drag (MouseMove), and you, the developer, can then set MList1.Drag 1, so dragging can begin. If the user releases the drag over the same list box, nothing changes. If the user moves over other controls, you will get the appropriate messages.

Clicking on a hilited item, and not moving the mouse, then releasing the mouse over the same spot, will de-select all other items in the list box, except for the one over which the mouse was released.

All in all, this property causes MList to behave like the File Manager program which comes with Windows. See the PROJECT1.MAK and File Manager for examples of the behavior.

Note This property is ignored if both ExtendedSelect and MultiSelect are **False.**

Data Type

Integer (enum)

ItemX Property

Description

This property, in conjunction with ItemY, determines which item in the list is currently processing mouse events. Not available at design time.

Usage

Here is an example that determines over which item a control was dropped:

```
Sub MList1_DragDrop (Source As Control, X As Single, Y As Single)


    MList1.ItemX = X
    MList1.ItemY = Y


    Debug.Print "Control was dropped over item at Index"; Str$(MList1.Item)


End Sub
```

Data Type

Integer

ItemY Property

Description

This property, in conjunction with ItemX, determines which item in the list is currently processing mouse events. Not available at design time.

Usage

Here is an example that determines over which item a control was dropped:

```
Sub MList1_DragDrop (Source As Control, X As Single, Y As Single)


    MList1.ItemX = X
    MList1.ItemY = Y


    Debug.Print "Control was dropped over item at Index"; Str$(MList1.Item)


End Sub
```

Data Type

Integer

Item Property

Description

Returns the Index of the string at ItemX and ItemY. Not available at design time. Read-only at run time.

Usage

Here is an example that determines over which item a control was dropped:

```
Sub MList1_DragDrop (Source As Control, X As Single, Y As Single)
    MList1.ItemX = X
    MList1.ItemY = Y
    Debug.Print "Control was dropped over item at Index"; Str$(MList1.Item)
End Sub
```

Note This property starts the search at TopIndex, since it assumes that the item you are looking for is currently displayed in the list box.

Data Type

Long

StringCompare Property

Description

Determines whether compares are case sensitive or case insensitive

Setting

The StringCompare property settings are:

Setting	Description
0	(Default) Case Sensitive.
1	Case Insensitive.

Remarks

This only affects the compare when determining placement in a Sorted list box.

Note This property does not affect the various searching properties.

Data Type

Integer (enum)

Resort Property

Description

Setting this property to 1 causes the list box to resort its contents. Not available at design time.

Setting

The Resort property settings are:

Setting	Description
False	(Default) The property is ignored.
True	Forces the MList to resort its contents.

Remarks

This property is ignored if the MList is not Sorted.

Warning The sort may fail if there is not enough disk space for resorting.

Data Type

Integer (boolean)

NoIntegralHeight Property

Description

Determines if the MList is sized to show only complete items. Read-only at run time.

Setting

The NoIntegralHeight property settings are:

Setting	Description
True	The MList can be set to any arbitrary size, and may partially obscure the last visible item.
False	(Default) The MList is sized to be an exact fit for the height of each line item.

Remarks

Set this property to **True** if you want to be able to size the list box to fit inside the entire window.   Normally, Windows forces the list box to be an exact fit for the height of each line item. For example: If you have room for ten line items, and each line item is 195 twips high, Windows will resize the list box to 1950 even if you tell it to set the list box to 2015.

Note Setting the ListBoxStyle property to 1 causes the MList to be created with the LBS_OWNERDRAWVARIABLE style. For list boxes with this style, the LBS_NOINTEGRALHEIGHT style is always enforced, and thus the setting of this property (NoIntegralHeight) is ignored. (See the *List Box Controls* topic in MSDN CD 7/95 for more information.)

Data Type

Integer (boolean)

DisableDrawing Property

Description

Suspends drawing until font changes, position changes, etc., are completed. Not available at design time.

Setting

The DisableDrawing property settings are:

Setting	Description
True	(Default) Suspends drawing (redrawing) of the control
False	Resumes drawing of the control.

Remarks

You might want to set this property to **True** while you are changing the font characteristics of a line that is visible in the list box. Before you set the last font attribute, set this value to **False**, and the line will redisplay itself with the correct font.

Data Type

Integer (boolean)

ItemFontBold Property

Description

Controls the **bold** style of an individual line item. Not available at design time.

Setting

The ItemFontBold property settings are:

Setting	Description
True	The item's text has the **bold** style.
False	(Default) The item's text does not have the bold style.

Caution

Setting the individual font attributes of each line will cause a font for that line to be created.   Windows will only support so many fonts. If you are going to be setting individual fonts, consider using the Unimplented property which will accept an actual font handle.

Usage

```
MList1.ItemFontBold = True
```

Note You should set ItemFontName before setting this property.

Data Type

Array of Integers (boolean)

ItemFontItalic Property

Description

Controls the *italics* style of the font for a specific line. Not available at design time.

Setting

The ItemFontItalic property settings are:

Setting	Description
True	The item's text has the *italics* style.
False	(Default) The item's text does not have the italics style.

Caution

Setting the individual font attributes of each line will cause a font for that line to be created.   Windows will only support so many fonts. If you are going to be setting individual fonts, consider using the Unimplented property which will accept an actual font handle.

Usage

```
MList1.ItemFontItalic = True
```

Note You should set ItemFontName before setting this property.

Data Type

Array of Integers (boolean)

ItemFontName Property

Description

Controls the font name for a specific line of text. Not available at design time.

Remarks

You should set this property first and then set the other font properties. See VB Help for details on the property "FontName".

Usage

```
MList1.ItemFontName(0) = "MS Sans Serif"
```

Data Type

Array of String

ItemFontSize Property

Description

Controls the individual size for a specific line of text. Not available at design time.

Usage

```
MList1.ItemFontSize(0) = 7.8
```

Note You should set ItemFontName before setting this property.

Data Type

Array of Integers

ItemFontStrikeThru Property

Description

Controls the ~~strikethru~~ style of the font for a specific line. Not available at design time.

Setting

The ItemFontStrikeThru property settings are:

Setting	Description
True	The item's text has the ~~strikethru~~ style.
False	(Default) The item's text does not have the strikethru style.

Caution

Setting the individual font attributes of each line will cause a font for that line to be created.   Windows will only support so many fonts. If you are going to be setting individual fonts, consider using the Unimplented property which will accept an actual font handle.

Usage

```
MList1.ItemFontStrikeThru(0) = True
```

Note You should set ItemFontName before setting this property.

Data Type

Array of Integers (boolean)

ItemFontUnderline Property

Description

Controls the underline style of the font for a specific line. Not available at design time.

Setting

The ItemFontUnderline property settings are:

Setting	Description
True	The item's text has the underline style.
False	(Default) The item's text does not have the underline style.

Caution

Setting the individual font attributes of each line will cause a font for that line to be created.   Windows will only support so many fonts. If you are going to be setting individual fonts, consider using the Unimplented property which will accept an actual font handle.

Usage

```
MList1.ItemFontUnderline(0) = True
```

Note You should set ItemFontName before setting this property.

Data Type

Array of Integers (boolean)

OwnerDraw Property

Description

Determines whether the program controls all drawing of an item in the list.

Setting

The OwnerDraw property settings are:

Setting	Description
True	The program controls all aspects of drawing an item in the MList.
False	(Default) Windows is responsible for drawing the item.

Remarks

See the DrawItem event for handling drawing of a line.

Data Type

Integer (boolean)

ListBoxStyle Property

Description

This property determines whether the lines in a list box are a fixed height or a variable height.

Setting

The ListBoxStyle property settings are:

Setting	Description
0	(Default) Fixed. All lines in the list box are a fixed height.
1	Variable. All lines in the list box are variable height.

Remarks

The property AddItemHeight should be set before adding an item to the list. After that, you can use IndItemHeight to adjust the height of a line. The reason for this is because I need the height of the line before the line is displayed, otherwise, the entire list must be redrawn when you change the height of that line with IndItemHeight.

Note Setting this property to 1 causes the MList to be created with the LBS_OWNERDRAWVARIABLE style. For list boxes with this style, the LBS_NOINTEGRALHEIGHT style is always enforced, and thus the setting of the NoIntegralHeight property is ignored. (See the *List Box Controls* topic in MSDN CD 7/95 for more information.)

Data Type

Integer (enum)

AddItemHeight Property

Description

Determines the height of the next line to be added when the MList is a variable height list box. (ListBoxStyle = 1).

Usage

```
 ' Default line height
MList1.AddItemHeight = 195
```

Data Type

Integer

IndItemHeight Property

Description

Adjusts the height of a line item after it has been added to a variable height MList. Not available at design time.

Remarks

Be aware that setting this property causes the entire list box to be redrawn. You might want to disable redrawing (DisableDrawing) while you adjust the height of a lot of lines.

Usage

```
MList1.IndItemHeight(0) = 395
MList1.IndItemHeight(2) = 595
```

Note This property only affects an Mlist with the ListBoxStyle property set to 1.

Data Type

Integer

InString Property

Description

Setting this property will cause the list box to search, from the current ListIndex, for an item that contains the property's value . If one is found, that item is set to the current ListIndex. Not available at design time. Write-only at run time.

Remarks

This property is like FindString, except this one looks for a string within another string.

Note The direction of the search is determined by the FindDirection property.

Data Type

String

FindColumn Property

Description

Determines the column searched by the FindColumnString property. Not available at design time.

Remarks

The direction of the search is determined by the FindDirection property.

Data Type

Integer

FindColumnString Property

Description

Specifies the string to search for in the FindColumn. Not available at design time. Write-only at run time.

Remarks

The direction of the search is determined by the FindDirection property.

Data Type

String

FindResult Property

Description

Controls the action of MList after it finds a string via FindString, FindStringExact, FindColumnString, or InString.

Setting

The FindResult property settings are:

Setting	Description
0	(Default) Update ListIndex. The ListIndex is updated, and the image of the list box is updated as well.
1	Store in FindIndex. The list box is not updated.

Remarks

Regardless of the setting, FindIndex is updated with the ListIndex of the found item, when an item is successfully found.

Data Type

Integer (enum)

FindIndex Property

Description

Contains the ListIndex of the last successful find of an item. Not available at design time.

Remarks

This property is updated whether FindResult is set to Update List Index or Update FindIndex. Compare this to ListIndex or previously set FindIndex to see if a find is successful.

Note In version 4.51, FindIndex will be set to -1 if the search should fail.

Data Type

Integer

OwnerCompare Property

Description

Determines whether the internal compare function or the CompareItem event is used to compare strings during sorting.

Setting

The OwnerCompare property settings are:

Setting	Description
True	The CompareItem event is called, thus allowing a user-defined compare algorithm to be executed.
False	(Default) The internal compare function is used.

Data Type

Integer (boolean)

FindPattern Property

Description

Searches the MList for the specified pattern. The FindIndex property contains the result. Not available at design time. Write-only at run time.

Remarks

You can use this property to find a pattern in the list. This property behaves exactly as FindString except that you may use wild cards.

Note Use the FindPatternColumn property to perform a wild-card search in a particular column.

Usage

The following example would find yours truly if it appears anywhere (in any column) in the list box. The starting point is determined by the FindDirection and SearchCompare determines the case sensitivity of the search.

```
MList1.FindPattern = "Robin W. Mc*"
```

Note See FINDSTR.MAK for a demo of this property.

Data Type

String

FindPatternColumn Property

Description

Searches the MList for the specified pattern in a particular column. The FindIndex property contains the result. Not available at design time. Write-only at run time.

Remarks

Set FindColumn to the column you want to search, then use this property to search for the pattern of the string in that column.

Note Use the FindPattern property to perform a wild-card search in all columns.

Usage

The following example would find the strings "Bits", "Bite" or "Bitstream" only if they appeared in column 2.

```
MList1.FindColumn = 2
MList1.FindPatternColumn = "Bit*"
```

Data Type

String

searching properties

FindColumnString

FindPattern

FindPatternColumn

FindString

FindStringExact

SearchCompare Property

Description

Determines the case sensitivity of the searching used by MList.

Setting

The SearchCompare property settings are:

Setting	Description
0	Case Sensitive
1	(Default) Case Insensitive

Remarks

This property is set apart from StringCompare so that you can have different case sensitivities for the sort order and the searches.

Note: This property does NOT affect InString, which is always case sensitive.

Data Type

Integer (enum)

ItemFont Property

Description

Sets the font property of one or more lines with the same font. Not available at design time.

Remarks

You can create a font by calling CreateFont or CreateFontIndirect. You can also get a system font by calling GetSystemObject. You can then use this font to assign a font property to different lines in your list box.

```
Dim font As Integer


font = CreateFont(...)


MList1.ItemFont(1) = font
MList1.ItemFont(20) = font
MList1.ItemFont(21) = MList1.ItemFont(20)
```

It is your responsibility to destroy the font. If you use the standard VB font properties, *FontName*, *FontBold*, etc., VB will destroy the font for you. MList assumes that the font you have assigned to an individual line item through this property is spread out all over the galaxy. Therefore, you need to clean it up.

Note You can also use the font property of a line whose font was created with the regular ItemFont properties. For example:

```
MList1.ItemFontName(0) = "Arial"
MList1.ItemFontBold(0) = TRUE
MList1.ItemFont(1) = MList1.ItemFont(0)
```

MList will clean up the font used by ItemFont(0), since it was created with the normal VB properties. Since the font is being cleaned up automatically, you do not have to clean up the same font which was assigned to ItemFont(1).

Data Type

Array of Integers

SelChange Event

Description

Occurs when a selection in the list box is changed either through the keyboard or a mouse event.

Syntax

Sub *ctlname*_**SelChange()**

Remarks

The event uses no arguments. You might want to use this event instead of or in addition to the Click event if you want to respond to selection changes when the user uses the arrow keys instead of the mouse.

Note This event mimics the standard list box's Click event. To respond only to mouse clicks, use the Click event.

SelCheck Event

Description

Occurs when the Checked property of a line in the list box is changed.

Syntax

```
Sub ctlname_SelCheck(Index As Integer, State As Integer)
```

Remarks

The SelCheck event uses these arguments:

Argument	Description
Index	The line item that has been affected
State	The state of the checked box. It is either **True** or **False**.

VirtualMessage Event

Description

Occurs when the user moves within VirtualMsgZone items of each the top or the bottom of the MList.

Syntax

Sub *ctlname*_**VirtualMessage(***Message* **As Integer)**

Remarks

The VirtualMessage event uses these arguments:

Argument	Description
Message	The message sent to this event.

This event provides the means to notify you when more data is needed. For now, you will have to keep track of your own data.

The following Message(s) are sent to this event:

Message	Description
VIRTUAL_UP	This message will be sent when you need to add items to the end of the list, because it is scrolling up.
VIRTUAL_DOWN	This message will be sent when you need to add items to the beginning of the list, because it is scrolling down.
VIRTUAL_END	This is sent when the user is moving to the end of the list. You should load whatever number of lines you need, then MList will position ListIndex to the end of the list.
VIRTUAL_HOME	This is sent when the user is moving to the beginning of the list. You should load whatever number of lines you need, then MList will position ListIndex to the start of the list

MList will move the current ListIndex as items are added to and removed from the list. You should try to cache all the items you will load in memory, then add them to the list one at a time. Save TopIndex and ListIndex, so that you can restore them. You will need to add or subtract the number of new items added or removed depending on the direction in which you are moving. See the example for details.

Note There are no properties like *VirtualListIndex*, *VirtualTopIndex*, *VirtualCount*, etc.   I am thinking hard about these, and hope to come up with an elegant solution. Any suggestions? I guess what I have done is provide virtual capabilities while leaving most of the implementation in your court.

VIRTUAL_UP 1

VIRTUAL_DOWN   2

VIRTUAL_END   3

ScrollMessage Event

Description

Occurs when the list box is scrolling ***horizontally***.

Syntax

Sub *ctlname*_**ScrollMessage(***Offset* **As Integer)**

Remarks

This event allows you to scroll your own column headers over the list box. The ScrollMessage event uses the following arguments:

Argument	Description
Offset	The number of ***pixels*** the window has been scrolled. See the scroll demo for details.

Known Problem

Due to Twips To Pixels and vice versa, there is no exact relationship between the number of pixels and the number of twips. If you look at the example, you will see that the more columns to the right you go, the further out of line the column header is. Manually placing each column header to start with solves this problem. Once this is done, just copy the code and go!

DrawItem Event

Description

Occurs when OwnerDraw is set to True and any change is made to the item at *ListIndex*.

Syntax

Sub *ctlname*_**DrawItem(***ListIndex* **As Integer**, *ItemAction* **As Integer**, *ItemState* **As Integer**, *ItemDC* **As Integer**, *ItemLeft* **As Integer**, *ItemTop* **As Integer**, *ItemRight* **As Integer**, *ItemBottom* **As Integer**, *ItemText* **As String)**

Remarks

OwnerDraw basically forces you to be responsible for drawing each and every line item.

The DrawItem event uses these arguments (See DRAWITEMSTRUCT in WinAPI Help for more details):

Argument	Description
ListIndex	0 based index of item being drawn.
ItemAction	Action being performed. (See DRAWITEMSTRUCT)
ItemState	State of line. (See DRAWITEMSTRUCT)
ItemDC	DC to do the drawing with. **DON'T** use MList1.hDC.
ItemLeft	Left coordinate of rectangle.
ItemTop	Top coordinate of rectangle.
ItemRight	Right coordinate of rectangle.
ItemBottom	Bottom coordinate of rectangle.
ItemText	The text that needs to be drawn.

Note See the commented out code in SCROLL.MAK for details on how you might get started using this feature (Set OwnerDraw to True!).

Formatting Strings

Description

You should place the Tab character between each column in your string. The following example formats a string for the DrawFlags example:

```
MList1.AddItem "Robin W. McKean" + Chr$(9) + "$100.00"
```

Examples

The following string is for three columns:

```
MList1.AddItem "Robin W. McKean" + Chr$(9) + "$100.00" + Chr$(9) + "True"
```

Remember, you do NOT have to include a column for the ImageRegion. If the ImageRegion property was 1 and the DrawRegions property was 3, then the following line would work fine (3-1=2)

```
MList1.AddItem "Robin W. McKean" + Chr$(9) + "$100.00"
```

Click Event

Description

Occurs when the user clicks the mouse on an item in the MList.

Syntax

```
Sub ctlname_Click()
```

Remarks

The Click event has no arguments.

Note This event responds only to mouse clicks, unlike the standard list box's Click event. Use the SelChange event instead to mimic the standard Click event.

CompareItem Event

Description

Occurs during a sorting operation when OwnerCompare is set to True.

Syntax

Sub *ctlname*_**CompareItem(***ListItem1* **As String,** *ListItem2* **As String,** *Result* **As Long)**

Remarks

You may parse the string out any way you like. The CompareItem event uses these arguments:

Argument	Description
ListItem1	The first string to be compared.
ListItem2	The second string to be compared.
Result	The result of the compare, returned to the calling routine (internal to the MList.) The return values for *Result* follow:

Value	Meaning
-1	ListItem1 is less then ListItem2.
1	ListItem1 is greater than ListItem2.
0	ListItem1 and ListItem2 are equal.

WARNING: `Result` will still be affected by SortOrder. If you plan to determine your own sort order, then set SortOrder to Ascending and leave it. Your sort order will not be affected. A SortOrder of Descending will effectively negate `Result` so it is the opposite of what you set it to.

AllowFocusRect Property

Description

Determines if the focus rectangle is drawn on the MList when it gets the focus.

Setting

The AllowFocusRect property settings are:

Setting	Description
True	(Default) The focus rectangle is drawn.
False	The focus rectangle is not drawn.

Data Type

Integer (boolean)

SortOrder Property

Description

Determines the sort order.

Setting

The SortOrder property settings are:

Setting	Description
0	(Default) Ascending. Sorts list items alphabetically, from A to Z
1	Descending. Sorts list items alphabetically, from Z to A

Data Type

Integer (enum)

MultiSelect Property

Description

Enables or disables the multiple-select feature of the list box. Read-only at run time.

Setting

The MultiSelect property settings are:

Setting	Description
True	Enables multiple-select.
False	(Default) Disables multiple-select.

Remarks

When enabled, this property allows the user to select multiple items by simply single-clicking them, without the need to hold down the Ctrl key. Clicking on an item that is already selected deselects that item (and only that item.)

Known Problem

When deselecting an item in an Mlist with MultiSelect set to True, subsequently retrieving the Text property returns the value of the just deselected item. This is due to ListIndex being set to the last item clicked. If this is not the desired behavior, it is up to you to set ListIndex to one of the currently selected items.

Note If MultiSelect is **True**, the default value for ListIndex is **0**, not **-1**, as is the case with the standard list box. Otherwise, it is **-1**.

Data Type

Integer (boolean)

Sorted Property

Description

Specifies whether the elements of the MList are automatically sorted alphabetically. Read-only at run time.

Setting

The Sorted property settings are:

Setting	Description
True	List items are sorted alphabetically
False	(Default) List items are not sorted alphabetically

Remarks

When set to **True**, the control handles almost all necessary string processing to maintain alphabetical order, including changing the index numbers for items as required by the addition or removal of items.

Note Using the AddItem method to add an element to a specific location in the list may violate the sort order, and subsequent additions may not be correctly sorted.

Data Type

Integer (boolean)

sorting operation

Occurs when the AddItem method is invoked for a sorted MList, or when the Resort property is set to true.

ItemPicture Example

Description

This example demonstrates the interrelated properties required to make the ItemPicture property work.

Steps

Open the ItemPic sample project (ITEMPIC.MAK)

Copy the code in the Form_Load topic to the Form_Load procedure.

Copy the code in the Declarations topic to the Declarations section.

Examples

Here is a list of the example projects documented in this Help file and supplied with MList.

Example	Description
ItemPicture	Demonstrates the use of this and other related properties
ItemHeight	Demonstrates the dynamic resizing of ItemHeight when changing the FontSize property.

Text Property

Description

Returns the text from all DrawRegions (columns) of the item pointed to by ListIndex. Not available at design time. Read-only at run time.

Remarks

If MultiSelect is True and multiple items are selected, the text from the last item selected is returned, since that is the item pointed to by ListIndex. (If no items are selected, ListIndex = 0.)

If ExtendedSelect is True and multiple items are selected, the text from the list item selected (at one end of the range) is returned, since that is the item pointed to by ListIndex. (If no items are selected, ListIndex = 0.)

Note This property differs from the standard list box's Text property in that it is unavailable at design time and read-only at run-time.

Data Type

String

ClickRegion Property

Description

Returns the DrawRegions (column) in which the last mouse click occurred. Read only at runtime.

Remarks

This property is set during the mouse down event.

Data Type

Integer

AutoCheck Property

Description

Controls the action taken when a user double clicks a line in the list box

Setting

The AutoCheck property settings are:

Setting	Description
True	(default) Double clicking a line will toggle the Checked state of the line item.
False	The checked state of the line item is not changed.

Remarks

The user may still change the checked state by clicking in the box.

Data Type

Integer (boolean)

SortType Property

Description

Determines the type of data in the SortColumn.

Setting

The SortType property settings are:

Setting	Description
0	(Default) String
1	Number, assumed of type float, but any number will do!
2	Date in the order MM/DD/YY or MM/DD/YYYY

Remarks

This isn't fair to users of other countries, so look for an enhancement to the date format.

Data Type

Integer (enum)

New Features Prior to v4.55

- SelCheck Event is triggered by code and events.
- FindIndex property is set to -1 when a search fails.
- Case sensitive finds and searches.
- Added FindPattern and FindPatternColumn properties for searching strings by pattern.
- Added SortOrder property for ascending and descending sort order.
- Added CompareItem function for implementing your own compare function.
- Added SelectMode property to allow you to use Ctrl Key and/or Mouse press to deselect items in a multiple select (ExtendedSelect) list box. This behavior imitates File Manager to a large degree.
- Added ItemFont property to allow to set the font properties of many lines with the same font.

New Features v4.56 to v4.58

- Added AutoCheck property
- Added SortType property

New Features v4.6

- Added ItemHgt.MAK demo project to illustrate how to resize the item height of an Mlist dynamically in response to changes in FontSize the property.

ItemHeight Example

Description

Demonstrates the dynamic resizing of ItemHeight when changing the FontSize property. The example is self-contained, so just run it and experiment. Try the following:

Steps

1. Load ITMHITE.MAK into VB and run it
2. Enter 15 in the FontSize textbox, then click Set Size. The font will get larger (assuming that 8.25 is the default FontSize in your environment.) But the space between each line has not increased.
3. Now scroll to the bottom of the list and click Add Items, and notice that the new items are spaced better, but the earlier items remain the same height.
4. Now click Reload Mlist....voila! All the items are the same, right height!
5. Now uncheck Skip Reload from the Options menu
6. Enter 8.25 in the FontSize textbox, then click Set Size. This time the entire contents of the MList are resized, as you would expect.

All the magic occurs in MListItemsResize... check it out!

You can also set the height directly by entering values in the ItemHeight textbox and clicking Set Height.

Remarks

You probably noticed that when you click Set Size after entering a FontSize, the value in the ItemHeight changed. Where did I get this value?

From a hidden Label control. The Label has an AutoSize property, which does exactly that. I'd seen it change size to fit the length of text, but I'd never tried changing the Height. So I tried that, and saw that it did resize the height of the label, automatically. I also noticed that the default Label.Height was 195, exactly that of the MList, when using the same Font and FontSize. So by setting the Label.FontSize to the same value as the MList.FontSize, I can get the appropriate value for MList.ItemHeight from Label.Height, which I then pass to MListItemsResize. (You can watch the Label change size by selecting Show Label Control from the Options menu.)

Notice that the value of ListBoxStyle is set to **1 - Variable** at design time. If you change it to **0 - Fixed** and rerun this example, the font size will change but the item height will not.