VBTOOLS - by MicroHelp Inc.

Custom Controls

Overview of Custom Controls

Custom Control Reference

MhGauge
MhState
MhTag

MicroHelp Products for Visual Basic

MicroHelp has more add-on products for Visual Basic than any other vendor. A DEMO of these products is included with these free controls. Before you buy any add-ons for Visual Basic, ask yourself these important questions:

1) Who has been in the Basic add-on business longer than any other company?
2) Who has MORE products and the most COMPLETE line of products for Visual Basic?
3) Who has shown the committment to Visual Basic since the day it shipped with more products available for Visual Basic from the very FIRST day?

VBTools version 2- Contains more than 30 custom controls, video "special effects", examples of how to use some of the more popular Windows API functions, access to the Windows 3.1 "Common Dialog Routines" and much MORE!

MicroHelp Muscle - The only Visual Basic add-on to win Computer Language Magazine's 1991 Productivity Award. Muscle contains over 430 assembly language routines that provide you with speed and functionality that is not available anywhere else. DOS and Windows low level services, string and array manipulation, date and time functions, bit manipulation and routines that will allow you to create arrays that can use all available Windows memory (including temporary and permanent swap files).

MicroHelp Communications Library - Since Visual Basic does not have any native communications support, this library provides you with everything you need, including source code for a working Windows terminal program. Seven different file transfer protocols that work even when your application is minimized, ANSI terminal emulation, and the ability to have upto eight simultaneous communications sessions at ONE TIME!

MicroHelp Network Library - 100% assembly language routines that support Novell, Lantastic and NetBIOS API functions. now you can access and manipulate all objects in the bindery, add, change and delete items in the print queues as well as redirect all shared resources.

MicroHelp VBXRef - The COMPLETE project management tool for Visual Basic. We can read the binary .FRM (form) files and generate reports that detail all Form and control properties, their current values and even show you which ones have been changed from their default settings. VBXRef can also generate a cross-reference report listing that shows all Procedures and variables, including their scope and even the line numbers where the variable's value is changed.

MicroHelp DATABasic - COMING in AUGUST 1992! B-Tree Database management routines. Automatically updates indexes and provides file, record and field locking. Does not require TSRs. Unlimited indexes, variable and fixed-length records. All source included.

For pricing or to order call:

1-800-922-3383

In Georgia or Outside the U.S. (404) 516-0899
FAX (404) 516-1099

Introduction

VBTools version 2 includes a wide selection of 36 different"Custom Controls". These are controls you add to your programs when you want to go beyond the capabilities of the controls that are a part of Visual Basic. This FREE DLL is a sample of three of the controls that are in VBTools. You can distribute the MHFR200.VBX free of charge, even give it to your friends. Please make sure you include this Windows Help file whenever you give someone the VBX so they too can use these controls to their fullest.

Why are we GIVING AWAY three custom controls and all the other Windows utilities? Because MicroHelp believes that once you get a chance to actually use our products, there will no doubt in your mind which Visual Basic add-on vendor has the BEST products on the market and gives you the MOST VALUE for your dollar.

Using Custom Controls in your applications is a simple matter of adding (Alt-F-Add) one of the five VBTools design mode DLL's to your application. After you do so, the icons for the Custom Controls stored in the DLL appear in the Toolbox Window and are available for use.

> **Tip**: When VB starts, it displays the standard control icons in the Toolbox using two columns and eight rows. When you add a custom control DLL to a project, VB places the new icons at the bottom of the Toolbox. When you have a large number of custom controls, the Toolbox can become quite large. If you prefer a square Toolbox, simply position the Toolbox at the bottom of the screen exposing only the number of rows you want the Toolbox to use. Now, when you add custom controls to your project, VB extends the Toolbox horizontally, with new columns, instead of vertically.

Technical Support: All of the information you need to use these custom controls is contained within this help file. Free technical support on these controls is available ONLY to registered users of VBTools.

Organization of the Libraries

All three of these FREE custom controls are in one DLL named MHFR200.VBX. Since these controls are identical to the ones in VBTools version 2, any programs you create using these controls will work unmodified with VBTools. To convert a project from the FREE controls to VBTools version 2:

1) Load the project that uses the FREE controls.
2) Use File-Remove File to remove MHFR200.VBX from the project.
3) Use File-Add File to add the commerical VBX that contains the control(s) used in your project.
4) Save your project.

This flexibility however, brings with it one drawback: You will **NOT** be able to run a Visual Basic application that uses the commercial version of any of these three controls at the same time as another program that is using the FREE version of the same control. If you run into this conflict, you should convert your program to use the commercial version of the controls as outlined above.

The MHFR200.VBX will work in both design and runtime modes. If you create an executable program using these controls don't forget to include the MHFR200.VBX with your program!

MHFR200.VBX - Contains all 3 of the Free Controls

MhGauge	MhState	MhTag

Accessing Help

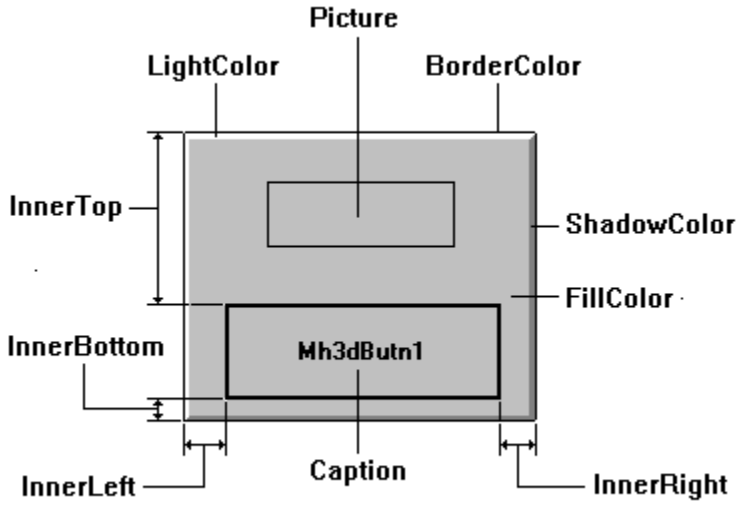
To make it easier for you to use the Custom Controls, VBTools includes a Windows Help file containing all the information necessary to use these controls. The simplest way to access the information for an individual control is to place a copy of the control on a form then position the cursor over the control and double click the right mouse button. When you do this, the help system will display the reference information about the selected control. Note, the directory containing the help file must be either in your path or in your Windows or Windows\System directory.

You can also load the help file by opening (Alt-F-Open) it from within the Windows Help program. The name of the help file is VBT200.HLP and, if you followed the installation instructions, it can be found in your Windows directory.

MicroHelp Standard Properties

Many of the VBTools Custom Controls have custom properties in common. We refer to the common properties as the MicroHelp Standard Properties. These properties fulfill the same function in all the VBTools Custom Controls in which they appear. This section summarizes these properties. Figure A identifies the MicroHelp Standard Properties as they appear on an Mh3dButn control.

Figure A: MicroHelp Standard Properties



The MicroHelp Standard Color Properties 
(The palette icon, shown above, is used to indicate that a control has "custom colors".)

Many VBTools Custom Controls have color properties that give you total control over each part of the control. For these controls, either or both of BackColor and ForeColor may appear in the list of properties, but they are there simply as placeholders, so that we can make use of VB's color palette.

All the properties listed below (and shown in Figure A) can take values within the range of VGA colors. The MicroHelp Standard Color Properties are:

Name	Description
BorderColor	Sets or returns the color of the outer border.
FillColor	Sets or returns the color displayed between the caption area and the outer border.
LightColor	Sets or returns the "light"color, as opposed to the "shadow" color of the control.
ShadowColor	Sets or returns the color used for "shadows".
TextColor	Sets or returns the foreground color of the caption.
TextFillColor	Sets or returns the background color of the caption.

The MicroHelp Standard Position Properties
You can place pictures and captions, with great precision, on many of the VBTools Custom Controls using the MicroHelp Standard Position Properties. We refer to these properties, in general, as the Inner*xxxx* properties, where *xxxx* represents Left, Right, Top and Bottom.

The values of the Inner*xxxx* properties are measured from the edge of the control. While you can specify the Inner*xxxx* property values in the edit box on the property bar, it's easier to use a mouse to resize the inner area of the control while you're in the VB Design mode. The next subsection describes the process.

Mouse Sizing 

(The mouse icon shown above, indicates that a control has special mouse sizing capabilities.)

You can use the right mouse button to change the values of the Inner*xxxx* properties in design mode. The right button lets you drag one of the borders of the inner area or it allows you to reposition the entire inner area when the appropriate cursor is visible.

Notice that when the cursor passes over an inner area, its appearance changes to indicate that sizing or movement is available. We have made these sizing cursors visually different than the Visual Basic cursors so that you can easily distinguish them. The left mouse button still operates normally, even when the right button sizing or movement cursor appears.

The two most common inner areas are the Caption*xxxx* and Picture*xxxx* areas. When a Custom Control contains a Caption and Picture inner area, the picture area is on "top" of the Caption area by default.

Methods

All the VBTools Custom Controls support the Drag, Move, Refresh and SetFocus methods. In addition, all List Box controls support the AddItem and RemoveItem methods. No other methods are supported.

Using Graphics with Custom Controls

The MhGauge Custom Control has the ability to display pictures. All custom properties with "picture" in the name can be either bitmaps (2-, 4- or 16-color) or icons.

In all cases, these properties operate exactly like the standard Visual Basic Picture property. You *set* these properties to tell the control which image should be displayed. At runtime, this property returns a "handle" to the picture.

When you use pictures in the controls and you specify those pictures at design time, the pictures become a part of your program. This means you do not have to distribute the bitmaps/icons separately with your programs. On the other hand, if you use the LoadPicture function to load pictures at runtime, then the bitmap files must be present at runtime.

Multiple Document Interface (MDI) 

(The icon shown above indicates that a control has MDI features.)

The Multiple Document Interface (MDI) is a Windows standard that specifies the way in which applications can provide access to multiple forms within a form. Many Windows word processing and spreadsheet programs use MDI to let you edit multiple files at the same time. From a programmer's perspective, MDI lets you manipulate multiple documents using a single set of code.

VBTools 2 has several controls that have MDI capabilites. Only the MhGauge control that is included with this free DLL has MDI properties.

MhGauge 

The MhGauge Custom Control has no counterpart in Visual Basic. You can use this control any time you want to show an analog-style gauge. For example, you can use it as a thermometer, fuel gauge, percent complete gauge, etc.

One advantage of analog gauges is that they give users pictures they can "process" quickly. Industrial psychology studies have shown that when you display numeric values, users must spend more time reading the screen and decoding the numbers than they do processing the information from well-designed analog gauges. The tradeoff is that analog gauges don't present values as precisely as their digital equivalents. In some cases, you may want to use an analog gauge and also unobtrusively display the exact value.

There are three basic styles of gauges:

- Linear gauges--horizontal and vertical varieties that use a "fill" area to display the gauge's value.
- Semi-circular gauges that use a needle to indicate the value.
- Circular gauges that use a needle to indicate the value.

When you design custom gauges, be aware that you must provide an "inner" area for the MhGauge control to display the current value of your control. For thermometer-like gauges, the inner area is similar to the bar of a bar chart. For both fuel and percent complete gauge controls, the inner area defines the area in which the needle displayed.

To set the value of an MhGauge control, you use the .Value property. Each time the .Value property changes, the Change event is fired.

Since you can set MhThreeD controls to fill their caption area with a "percent complete" bar, you might consider using MhThreeD controls as gauges.

Please see the MHGAUGE project for examples of how this control can be used.

MhState 

The MhState custom control is used to display and/or modify the state of the Caps Lock, Num Lock, Insert and Scroll Lock keys. All you have to do to use this control is place it on your form and set the .Style property to the keyboard state you want the control to display.

You don't have to write any additional code to support this control unless you want to *get* the state of the keyboard. The MhState control automatically handles switching keyboard states on or off when the user clicks the control.

The obvious benefit to this control is that it gives an LED indication to users who don't have LED's on their keyboards and allows "heavy" mouse users to avoid the keyboard when those keys need pressing.

See the MHSTATE project for a demonstration of this control.

MhTag

MhTag is similar to a standard VB List Box, but it has several enhancements:

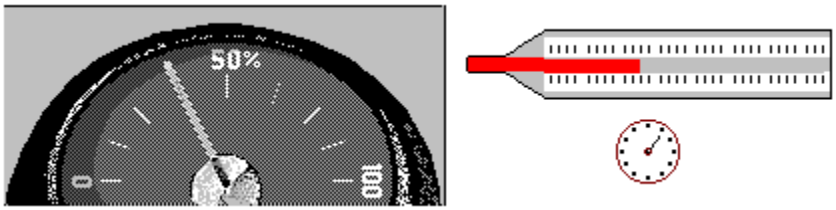
- The user can select multiple items from the list. You can determine not only how many items are selected, but specifically which ones.
- You can display data in multiple columns with a minimum of effort.
- You get an *immediate* reaction to movement of the scrollbar "thumb".
- We provide a built-in search capability.
- You can clear the contents of the list box with a single instruction.
- You can control whether the box contents are updated with every change to the contents. This allows you to fill a box before the display is updated, thus eliminating flickering.
- You can scroll the contents of the list box under program control.

This type of control is ideal in file management applications where the user wants to perform an operation on multiple files.

The MHTAG project demonstrates the usage of the MhTag control. Please refer to *MhIconTag* on page 75 for another enhanced list box control.

MhGauge

Description: The MhGauge control displays your choice of linear (filled), needle or bitmap style gauges. MhGauge can also use custom bitmaps of gauges that you create, insteadof the bitmaps we supply. Typical MhGauge controls are shown below.



Class Name: MhGauge

Remarks: MhGauge controls are useful for percent-complete indicators, fuel gauges, thermometers or any other type of analog gauge.

In the directory containing the MHGAUGE project, you'll find the bitmaps we used to create the gauges shown above. When you use bitmaps (or icons) in the MhGauge control, and you specify them in the .Picture and .PictureFill properties at design time, the bitmaps become a part of your program. This means you don't have to distribute the bitmaps/icons separately with your programs. On the other hand, if you use the LoadPicture function to add bitmaps/icons at runtime, then the files must be present at runtime.

You can easily control the range in which the needles are displayed by setting a wide difference between .Min and .Max and a narrower range for the actual numbers placed into .Value. For example, to have a semi-circular gauge where the needle moves only in the 90-degree quadrant surrounding "straight up", you would set:

```
.Min = 0
.Max = 100
LowestValueYouPass = 25
HighestValueYouPass = 75
```

Properties

AutoSize	*BackColor	Caption	ControlBox
CtlName	DragIcon	DragMode	Enabled
*ForeColor	Height	HWND	Icon
Index	*InnerBottom	*InnerLeft	*InnerRight
*InnerTop	Left	*Max	MaxButton
*MDI	*Min	MinButton	MousePointer
*NeedleWidth	Parent	Picture	*PictureFill
*ReverseFill	*ScaleMode	*Sizeable	*Style
TabIndex	TabStop	Tag	Top
*Value	Visible	Width	WindowState

Events

*Change	Click	*Close	DblClick
DragDrop	DragOver	GotFocus	KeyDown
KeyPress	KeyUp	LostFocus	MouseDown
MouseMove	MouseUp	*Move	

Custom Properties

MhGauge uses three standard VB properties, marked with asterisks above, to set the range and value of the gauge's indicator. The properties function in the same fashion as described in the ***Microsoft Visual Basic Language Reference Manual.*** *The default values for the properties are:*

Min - 0
Max - 100
Value - 0

BackColor *(Long Integer)*

Description: The .BackColor property sets the color used to fill an "empty" linear gauge.

Usage: [*form*.]*MhGauge*.**BackColor** = *colorvalue&*

Remarks: Think of this as the background color of the fill area. The default value is &H80000005& (White). The .BackColor property doesn't effect the color of needles when the .Style property is two or three.

ForeColor *(Long Integer)*

Description: The .ForeColor property sets the color used to paint the "percent complete" portion of the gauge.

Usage: [*form*.]*MhGauge*.**ForeColor** = *colorvalue&*

Remarks: Think of this as the foreground color of the fill area. The default value is &H80000008& (Black). The .ForeColor property does not affect the color of needles when .Style is two or three.

InnerTop, InnerLeft, InnerBottomand InnerRight *(Integer)*

Description: These properties specify the inner dimensions of a gauge control, relative to the borders of the control

Usage: [*form*.]*MhGauge*.**InnerTop** = *topposition%*
[*form*.]*MhGauge*.**InnerLeft** = *leftposition%*
[*form*.]*MhGauge*.**InnerBottom** = *bottomposition%*
[*form*.]*MhGauge*.**InnerRight** = *rightposition%*

Remarks: These values must be positive and expressed using values in the control's ScaleMode. They specify the coordinates of the area of the control in which the "percent complete" value is displayed (see the .Style property). The values for the .InnerTop and .InnerLeft properties are relative to the top left borders of the control. In other words, the top left corner of the inner area is .InnerTop units below the top border and .InnerLeft units to the right of the left border. Similarly, the values for the .InnerBottom and .InnerRight properties are relative to the bottom right edge of the control.

Needle gauges display a needle within these coordinates. See remarks.

MDI *(Boolean)*

Description: The .MDI property determines whether the control has MDI features.

Usage: [*form*.]*MhGauge*.**MDI** = *boolean%*

Remarks: If the .MDI property is FALSE, the control doesn't display the Control Box, the Minimize Button or the Maximize Button, even if those properties are TRUE. The default value is FALSE. The valid settings are:

Setting	Description
TRUE (-1)	Display the MDI features.
FALSE (0)	Do not display the MDI features.

NeedleWidth *(Integer)*

Description: The .NeedleWidth property sets the width of the needle, in the control's .ScaleMode, when the .Style property is two or three

Usage: [*form*.]*MhGauge*.**NeedleWidth** = *integer%*

Remarks: The default value is one pixel or 15 Twips..

PictureFill *(Picture)*

Description: The .PictureFill property sets or returns the fill picture used for MhGauge controls when the .Style property is five or six. This property is set the same way as a standard Visual Basic picture property.

Usage: [*form*.]*MhGauge*.**PictureFill** = *picture*

Remarks: The .PictureFill property lets you use bitmaps to display the "percent complete" value instead of a linear or needle indicator. You must set this property for the bitmap fill operation to work.

The bitmap (or icon) you load with the .PictureFill property should be an exact copy of the .Picture bitmap (or icon) except for the fill area. The .PictureFill bitmap should show a completely full gauge and the .Picture bitmap should show a completely empty gauge. When you specify an amount for the .Value property, the MhGauge control determines how much of the bitmap to display in the .Picture bitmap.

Even though most gauges will not use the full length (or height) of the picture you supply, it is still important to set the Inner*xxxx* properties so they accurately reflect the area that "appears" to be filled by the gauge. When you set these properties, the control scales the .Min, .Max and .Value properties to that area and the gauge appears to reflect the value setting properly. If the Inner*xxxx* properties are not set properly, the .Value property might appear to be "out of sync" with the gauge.

The easiest way to size the Inner*xxxx* properties is to first set the gauge style to one of the "normal" linear styles. Next, set the Inner*xxxx* properties (using the right mouse button is easiest), then switch the gauge back to the bitmap fill style.

The default value is zero (no bitmap).

ReverseFill *(Boolean)*

Description: The .ReverseFill property sets or returns the direction in which the fill takes place.

Usage: [*form*.]*MhGauge*.**ReverseFill** = *boolean%*

Remarks: The default value is FALSE. The valid settings are:

Setting	Description
TRUE (-1)	Linear horizontal gauges fill from right to left and vertical linear gauges fill from top to bottom.
FALSE (0)	Linear horizontal gauges fill from left to right and linear vertical gauges fill from top to bottom.

ScaleMode *(Integer)*

Description: The .ScaleMode property sets or returns the scale of measurement for the control.

Usage: [*form*.]*MhGauge*.**ScaleMode** = *integer%*

Remarks: The default value is three (pixels). The valid settings are:

Setting	Description
1	Twips
3	Pixels

Sizeable *(Boolean)*

Description: The .Sizeable property sets or returns whether the list box is sizeable at runtime.

Usage: [*form*.]*MhGauge*.**Sizeable** = *boolean%*

Remarks: The default value is FALSE. The valid settings are:

Setting	Description
TRUE (-1)	Allows the control to be resized at runtime.
FALSE (0)	Prevents the control from being resized at runtime.

Style *(Integer)*

Description: The .Style property determines the style of the MhGauge control.

Usage: [*form*.]*MhGauge*.**Style** = *setting%*

Remarks: The default value is zero. The possible values are:

Setting	Description
0	Horizontal Bar, a horizontal linear gauge with "fill".
1	Vertical Bar, a vertical linear gauge with "fill".
2	'Semi' Needle, a semi-circular gauge with "needle". The needle base is centered at the bottom of the "inner" rectangle. When .Value = .Min, needle points 90 degrees to left. When .Value = .Max, needle points 90 degrees to the right. When .Value = (.Min + .Max) / 2, the needle points straight up.
3	'Full' Needle, a circular gauge with a "needle". The base of the needle is placed at the center of the inner rectangle. When .Value = .Min or .Value = .Max, the needle points 90 degrees to the left. Setting the .Value property in between .Min and .Max points the needle to a proportionate point on the

circle, moving clockwise.

4 Horizontal fill with bitmap. This style works the same as style zero, but uses a programmer-supplied bitmap to fill the control. See the .PictureFill property for details.

5 Vertical fill with bitmap. This style works the same as style one, but uses a programmer-supplied bitmap to fill the control. See the .PictureFill property for details.

Custom Events

Change

Syntax: *CtlName*_Change()

Description: The Change event occurs whenever the .Value property changes. The only way to change the .Value property is from code.

Close

Syntax: *CtlName*_Close()

Description: The Close event is fired when the Close option is selected from the control box menu. This event can occur only when .MDI is TRUE and .ControlBox is TRUE.

Move

Syntax: *CtlName*_Move()

Description: The Move event is fired when the control is moved (by dragging it with the mouse or moving it via the control box menu). This event can occur only when .MDI is TRUE and .ControlBox is TRUE.

Example: The MHGAUGE project

MhState 

Description: The MhState custom control lets you display and/or modify the Caps Lock, Num Lock, Insert and Scroll Lock keyboard states. It uses the bitmaps shown below to represent each of the modifier keys.

Class Name: MhKeyState

Remarks: You use the .Style property to specify which modifier key you want the MhState control to represent. MhState controls let you reflect the current state and let users change the state of the keyboard modifier keys without writing any code. If you want to maintain a particular keyboard state, you can place code in the Change event procedure to trap changes in the keyboard state.

If you need to determine the current state of a keyboard modifier key but don't want to place an MhState control on the form, consider using the MhGetKeyState function. That function returns the state of the same four modifier keys.

Properties

*AutoSize	BackColor	CtlName	Enabled
Height	Index	Interval	Left
MousePointer	Parent	*Style	TabIndex
TabStop	Tag	Top	*Value
Visible	Width		

Events

*Change	Click	GotFocus	KeyDown
KeyPress	KeyUp	LostFocus	

Custom Properties

AutoSize *(Boolean)*

Description: The .AutoSize property sets or returns whether the control automatically sizes itself to the bitmaps.

Usage: [*form*.]*MhState*.**AutoSize** = *boolean%*

Remarks: The default is TRUE (-1). Valid settings are:

Settings	Description
TRUE (-1)	The control sizes itself to the bitmaps.
FALSE (0)	The control uses StretchBlt to make the bitmaps fill the control.

Style *(Integer)*

Description: The .Style property sets or returns which of the keyboard states you want the control to display:

Usage: [*form*.]*MhState*.**Style** = *setting%*

Remarks: The default value is zero. The valid settings are

Setting	Description
0	Caps Lock
1	Num Lock
2	Insert State
3	Scroll Lock

Value *(Integer)*

Description: The .Value property sets or returns the current state of the control.

Usage: [*form*.]*MhState*.**Value** = *setting%*

Remarks: The default value is zero. Not available at design time. The possible values are:

Setting	Description
0	Off
1	On

Custom Events

Change

Syntax: *CtlName*_Change()

Description: The Change event occurs when the .Value property is changed. That happens when the user presses the associated keyboard key, clicks on the control or when you change the .Value property from code.

Example: The KEYSTATE project

MhTag 

Description: The MhTag control is similar to the standard VB List Box control, but has several enhancements.

Class Name: MhTagBox

Remarks: Since the MhTag control includes all the functionality of the standard VB List Box control, you can use it in place of any VB List Box control. Among its custom features are the ability to select multiple items at one time, clear the control with a single instruction, search for strings in the list box and display items in multiple columns.

Properties

BackColor	*ClearBox	*ColumnWidth	CtlName	
DragIcon	DragMode	Enabled		
*ExtendedSelect		*FindString	FontBold	FontItalic
FontName	FontSize	FontStrikeThru	FontUnderLine	
ForeColor	*FoundString	Height	*HorizScroll	
Index	Left	List	ListCount	
ListIndex	MousePointer	*MultiColumn	Parent	
*ScreenUpdate	*SelectedCount	*SingleSelect	Sorted	
TabIndex	TabStop	Tag	*Tagged()	
Top	*TopIndex	Visible	Width	

Events

Change	Click	DblClick	DragDrop
DragOver	GotFocus	KeyDown	KeyPress
KeyUp	LostFocus	MouseDown	MouseMove
MouseUp			

Custom Properties

ClearBox *(Integer)*

Description: Anytime you set this property, regardless of the value, the control clears the contents of the List Box.

Usage: [*form*.]*MhTag*.**ClearBox** = *integer%*

Remarks: The value can be any integer. Not available at design time. Write-only at runtime.

ColumnWidth *(Integer Array)*

Description: The .ColumnWidth property sets or returns the width of a "column" used to display one of the elements of information.

Usage: [*form*.]*MhTag*.**ColumnWidth(column%)** = *integer%*

Remarks: The value of the .ColumnWidth property is expressed in characters. The columns are separated by "embedded" tabs within the text of a list item. The columns are numbered from zero to 30. Not available in design mode.

ExtendedSelect *(Boolean)*

Description: The .ExtendedSelect property sets or returns whether the mouse can be

used for extended selection. The term "extended selection" refers to the way multiple items can be selected in File Manager, i.e., a *range* of items can be selected..

Usage: [*form*.]*MhTag*.**ExtendedSelect =** *boolean%*

Remarks: If you set this property to TRUE, the .SingleSelect property is automatically set to FALSE. The default value is FALSE. Read-only at runtime. The valid settings are:

Setting	Description
TRUE (-1)	Allows use of the mouse for extended selection.
FALSE (0)	Does not allow extended selection.

The .ExtendedSelect property allows the selection of multiple *contiguous* items. The .SingleSelect property allows the user to select multiple items, whether or not they are contiguous.

FindString *(String)*

Description: The .FindString property specifies a string to search for in the List Box. MhTag returns the result of the search in the FoundString property.

Usage: [*form*.]*MhTag*.**FindString** = *string$*

Remarks: The search takes place on the first *n* characters of each line in the control, where *n* is the length of *string$*. The result of the search is returned in the .FoundString property. Not available at design time. Write-only at runtime. Please see the .FoundString property for an example of using .FindString and .FoundString.

FoundString *(Integer)*

Description: The .FoundString property returns the result of the most recent search performed with the .FindString property.

Usage: *integer%* = [*form*.]*MhTag*.**FoundString**

Remarks: Not available at design time. Read-only at runtime. The possible return values are:

Return Value	Description
TRUE (-1)	The string was not found.
All other values	The line (element) where the string was found.

HorizScroll *(Boolean)*

Description: The .HorizScroll property sets or returns whether the control displays a horizontal scroll bar, if appropriate.

Usage: [*form*.]*MhTag*.**HorizScroll** = *boolean%*

Remarks: The default value is FALSE. Read-only at runtime. The valid settings are:

Setting	Description
TRUE (-1)	The control displays a horizontal scroll bar when required.
FALSE (0)	If the width of the control is less than the width of the information being displayed, the control clips the right side of the list box information.

MultiColumn *(Boolean)*

Description: The .MultiColumn property sets or returns whether the MhTag control displays items in a normal (single-column) display or if it formats the contents into multiple columns (to maximize the use of all available space).

Usage: [*form.*]*MhTag*.**MultiColumn** = *boolean%*

Remarks: The default value is FALSE (0). Read-only at runtime. Valid settings are:

Setting	Description
TRUE (-1)	The control supports multiple columns.
FALSE(0)	The controls supports a single column.

ScreenUpdate *(Boolean)*

Description: The .ScreenUpdate property sets or returns whether changes to the list box are displayed at the time the contents change.

Usage: [*form.*]*MhTag*.**ScreenUpdate** = *boolean%*

Remarks: This property is ideal for situations where you want to fill a control with data and then display all the contents at once. Note that setting the property to TRUE does not cause an immediate refresh, but the .Refresh method can be used to cause the display to update. The default value is TRUE. The valid settings are:

Setting	Description
TRUE (-1)	Causes the display to be updated whenever you change the contents of the list box.
FALSE (0)	Changes in the list box contents are not visible on the display.

SelectedCount *(Integer)*

Description: The .SelectedCount property returns the number of items that are currently selected (i.e., have their .Tagged property set to TRUE).

Usage: *count%* = [*form.*]*MhTag*.**SelectedCount**

Remarks: Not available at design time. Read-only at runtime.

SingleSelect *(Boolean)*

Description: The .SingleSelect property sets or returns whether the control can be used to select multiple, non-contiguous items.

Usage: [*form.*]*MhTag*.**SingleSelect =** *boolean%*

Remarks: If you set this property to TRUE, the .ExtendedSelect property is automatically set to FALSE. The default value is FALSE. Read-only at runtime. The valid settings are:

Setting	Description
TRUE (-1)	Restricts the user to selecting one item.
FALSE (0)	Allows the user to select multiple, non-contiguous items.

Setting .SingleSelect to FALSE allows the user to select multiple items, whether or not they are contiguous. The .ExtendedSelect property allows the selection of multiple *contiguous* items.

Tagged *(Boolean Array)*

Description: The .Tagged property array returns whether the specified element is currently selected.

Usage: *boolean%* = [*form*.]*MhTag*.**Tagged(element%)**

Remarks: Each element corresponds to a line in the list box. The first line is zero. Not available at design time.

Setting	Description
TRUE (-1)	The element is selected.
FALSE (0)	The element is not currently selected.

Example: The following code checks to see if element three is selected.

```
If MhFileList1.Tagged(3) Then
  ' Element 3 has been selected
End If
```

TopIndex *(Integer)*

Description: The .TopIndex property sets or returns the index number of the element that appears at the top of the selection window.

Usage: *index%* = [*form*.]*MhTag*.**TopIndex**

Remarks: Remember that the first item in a list box is number zero and the last item is *ListCount-1*. Not available at design time.

Example: The MHTAG project