Contents

Using VB Messenger

Adding VB Messenger to Your Application

All Visual Basic custom controls are loaded into your project from within the Visual Basic environment. Once a custom control is added to your project and your project is saved, it will always load whenever you load that project.

To add the VB Messenger custom control to your project, choose **'Add File'** from the **'File'** menu in Visual Basic.   Then, enter the name of the design time VB Messenger custom control, and its fully qualified path if necessary.   The VB Messenger icon should now appear in you Visual Basic Toolbox.

To use VB Messenger, you must first add it to a form in your project.   When you add the VB Messenger custom control to your form, it will appear on your form as an icon similar to the one that appears in the Toolbox.   Although you can reposition the icon anywhere you like on the form, the position of the icon is not important.   When you run your application, the custom control becomes *invisible* and therefore cannot be seen by the user.

Refer to the *'Loading Custom Controls'* section in the *Visual Basic Programmer's Guide* for a detailed explanation of adding controls to your project.

How Do I Use VB Messenger? - A Guided Tour

VB Messenger is used to subclass a form or control in Visual Basic to intercept the Windows messages that are associated with the form or control.

Once added to your form, VB Messenger is ready to use.   Most of VB Messenger's properties can be set via the property window in design mode of Visual Basic.

The following instructions provide you with a sample walk through on using VB Messenger with your application.   It walks you through the creation of a sample program that illustrates the fundamental usage of the VB Messenger Custom Control.   The program uses VB Messenger to trap the **WM_MENUSELECT** message that is sent to the main form when the user highlights a menu item in a drop down menu.   Then the program will display information about the menu item selected on a message line at the bottom of the screen in a status bar.
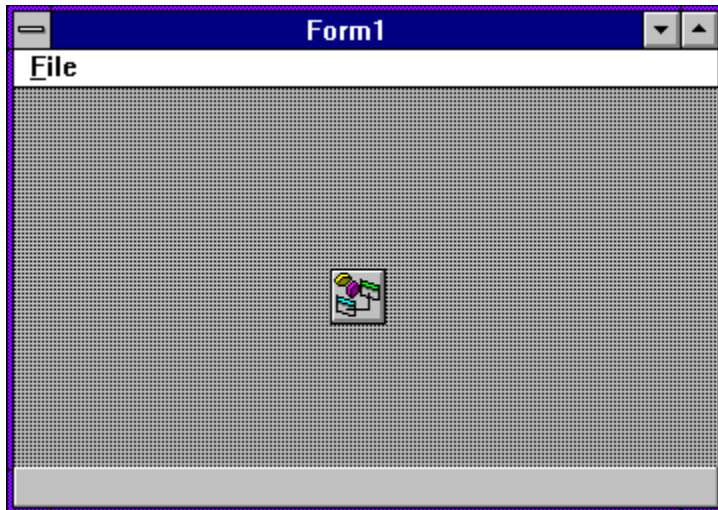
Step 1.

First create the following form with the following drop down menu.   The bottom of the form contains a 3D Panel control with the Align property set to Align Bottom.   You may substitute a standard Visual Basic Label if you wish.
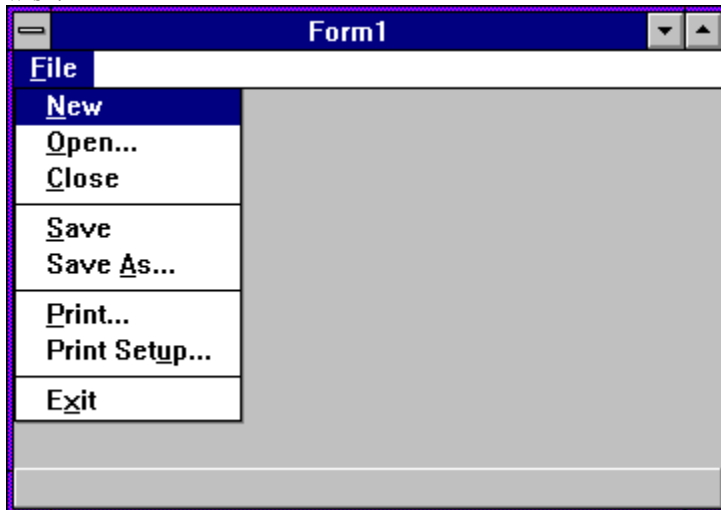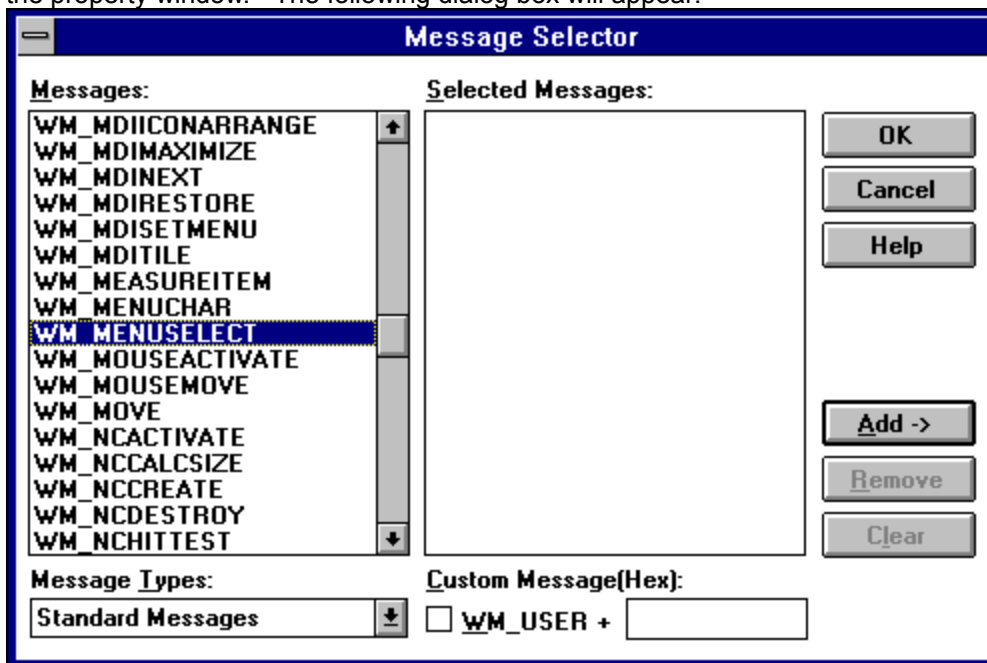

Step 2.

Add the VB Messenger Custom Control to the new form.

Step 3.

Bring up the property window for the new VB Messenger Custom Control.   Select the **(Message Selector)** property from the property window.   Click once on the ellipses ('...') button next to the text in the property window.   The following dialog box will appear:

Scroll through the **Messages** list box and search for **WM_MENUSELECT**.   Select this message by double clicking on it with the mouse or by pressing the **Add->** button.   The message will then appear in the **Selected Messages** list box.   Click on **OK** to save the selection and close the Message Selector dialog box.

Step 4.

Add the following code to the Form1's Form_Load procedure:

```
VBMsg1.SubClasshWnd = Form1.hWnd
```

When the form is loaded, this code will be executed instructing VB Messenger to subclass the main form.   VB Messenger immediately starts intercepting messages at this point.   To turn off the message

processing, just set the SubclasshWnd property to zero.

Step 5.

Now add the following code to the VBMsg1_WindowMessage procedure:
```
Panel3D1.Caption = "wParam =" & Str$(wParam)
```

wParam is passed to this procedure by VB Messenger.

Step 6.

Run the program.   Use the mouse to select through the different menu items and watch how VB Messenger intercepts the message and allows you to display the information about the menu item on the status bar.

This example detects when the user is selecting a menu item and displays the wParam parameter associated with the message (WM_MENUSELECT) at the botton of the screen.   You can use this number in your program to reference a line of text that may be used to describe the particular function on the menu.   Then you can display this text on the bottom of the screen, very similarly to other Windows programs.

See the sample project **MENU.MAK**.

VB Messenger Custom Control Reference

Description   The VB Messenger Custom Control allows you to subclass a form or control to receive and/or intercept its messages.

File Name   VBMSG.VBX

Object Type   VBMsg

Related Topics:

[Properties , Events, and Methods](#)
[Properties Reference](#)
[Events Reference](#)

Properties , Events, and Methods

All of the properties, events, and methods for VB Messenger are listed in the tables below.   All standard Visual Basic properties, events, and methods are denoted with an asterisk(*) and can be found documented in the *Visual Basic Language Reference* that comes with Visual Basic.

Properties

About	AddMessage	ClearMessages
*Height	HiWord	*hWnd
*Index	*Left	LoWord
lParam	lParam2String	MessageCount
MessageList	MessageSelector	MessageText
MessageTypes	*Name	*Parent
PostMessage	RemoveMessage	ReturnVal
SendMessage	String	SubclasshWnd
*Tag	*Top	*Width
wParam		

Events

WindowMessage	WindowDestroyed

Methods

VB Messenger does not support any methods.

Properties Reference

The following is a detailed reference of all the properties supported by VB Messenger.

Related Topics:

About Property

Description	Displays version information about the VB Messenger Custom Control.
Usage	Double click on the ellipses ('...') button next to the property text to activate the about dialog box.
Remarks	Available only at design time.
Data Type	N/A

AddMessage Property

Description	Adds a message to the MessageList property.
Usage	[*form.*]*VBMsg.***AddMessage**[= *message&*]
Remarks	To specify which messages are intercepted by VB Messenger at runtime, you must use this property. Each time you set the AddMessage property to a message, a new message gets appended to the end of the MessageList property array.

Example:

```
VBMsg.AddMessage = WM_PAINT
VBMsg.AddMessage = WM_SIZE
VBMsg.AddMessage = WM_CLOSE
```

Available only at runtime and is write only.

Data Type	Long
See Also	MessageSelector, RemoveMessage, ClearMessages

ClearMessages Property

Description	Clears all messages from the MessageList property.
Usage	[*form.*]*VBMsg.***ClearMessage** = *True*
Remarks	Setting this property to True (or any integer value) clears all the messages from the MessageList property. This is the equivilent of using RemoveItem for all messages in the MessageList.
	Only available at runtime and is write only.
Data Type	Boolean
See Also	RemoveMessage, AddMessages, MessageSelector

HiWord Property

Description	Returns or sets the high-order word of the 32-bit long integer value in the lParam property.

Usage	[*form.*]*VBMsg.***HiWord**[= *value%*]
Remarks	This property is used primarily in conjunction with the LoWord property to create the lParam property. This property is useful when you need to send the lParam parameter with a message to a window that calls for the high-order word to be a certain value.
	Setting this value to an integer causes VB Messenger to combine this value with the LoWord property and set the lParam property to the result. VB Messenger performs a concatenation of the two 2-byte integer values to produce the 4-byte integer lParam property.
	Setting the lParam property to a long integer causes VB Messenger to parse out two 2-byte values and place the results in LoWord and HiWord respectively.

Example:
```
VBMsg1.LoWord = Form1.hWnd
VBMsg1.HiWord = 100
Print "The resulting lParam is:"; VBMsg1.lParam
```

Available only at runtime.

Data Type	Integer
See Also	LoWord, lParam, SendMessage, PostMessage

LoWord Property

Description	Returns or sets the low-order word of the 32-bit long integer value in the lParam property.
Usage	[*form.*]*VBMsg.***LoWord**[= *value%*]
Remarks	This property is used primarily in conjunction with the HiWord property to create the lParam property. This property is useful when you need to send the lParam parameter with a message to a window that calls for the low-order word to be a certain value.
	Setting this value to an integer causes VB Messenger to combine this value with the HiWord property and set the lParam property to the result. VB Messenger performs a concatenation of the two 2-byte integer values to produce the 4-byte integer lParam property.
	Setting the lParam property to a long integer causes VB Messenger to parse out two 2-byte values and place the results in LoWord and HiWord respectively.

Example:
```
VBMsg1.LoWord = Form1.hWnd
VBMsg1.HiWord = 100
Print "The resulting lParam is:"; VBMsg1.lParam
```

Available only at runtime.

Data Type	Integer
See Also	HiWord, lParam, SendMessage, PostMessage

lParam Property

Description This property represents the 32-bit long value of the Windows message structure.

Usage [*form.*]*VBMsg.***lParam**[= *value&*]

Remarks This property is used primarily in conjunction with the SendMessage and PostMessage properties as a parameter for sending messages directly to a subclassed window.

This property can also be used to parse out the low-order and high-order word values of any 32-bit long integer.   The results can be found in the HiWord and LoWord properties respectively.

Example:
```
' select a range of items in a multi-select list box
VBMsg1.wParam = True
VBMsg1.LoWord = 0
VBMsg1.HiWord = List1.ListCount

'The concatenated value is now in lParam property
VBMsg1.SendMessage = LB_SELITEMRANGE
```

Available only at runtime.

Data Type Integer

See Also wParam, SendMessage, PostMessage

lParam2String Property

Description This property converts a 32-bit address to a Visual Basic string.

Usage [*form.*]*VBMsg.***lParam2String**[= *value&*]

Remarks Setting this value to a valid 32-bit far segment address (stored in a long integer) causes VB Messenger to place the data pointed to by the address into the String property.   The length of the resulting String property is determined by the first occurrance of an ASCII 0 in the data.

WARNING!   Use this property very carefully.   Setting this to an invalid pointer could result in undesirable results such as a GPF or loss of data.   Do not set this property to anything else except a valid pointer.

Generally this property is used to convert the lParam message parameter passed from within the WindowMessage event procedure to a Visual Basic string.

Available only at runtime and is write only.

Example:

```
'... from within the VBMsg1_WindowMessage proc ...
```

```
                              .
                              .
                VBMsg1.lParam2String = lParam
                Print "The resulting string is: " & VBMsg1.String
```

Data Type   Long

See Also   lParam

MessageCount Property

Description   Returns the number of messages in the MessageList property array.

Usage   [*form.*]*VBMsg.***MessageCount**

Remarks   Available only at runtime and is read only.

Data Type   Integer

See Also   MessageList

MessageList Property

Description   Contains a list of all messages to be intercepted by VB Messenger.

Usage   [*form.*]*VBMsg.***MessageList(***index)*[= *message&*]

Remarks   This property array contains all the messages set by either
                         AddMessage or at design time by the MessageSelector dialog.   Each
                         time a new message is added, the message gets appended to the end
                         of this list, increasing the count by one.   To access any message in
                         the list, you must specify the index of the array.

                         You can also change messages in the list by assigning the specific
                         element in the property array.

                         Example:

```
' changes the 3rd message (0 based) in the list
' to WM_CLOSE
VBMsg.MessageList(2) = WM_CLOSE
```

                         Available only at runtime.   The first element in the array is at index 0.

Data Type   Long

See Also   MessageCount, MessageSelector, AddMessage, RemoveMessage,
                         ClearMessage

## *MessageSelector Property*

Description   Displays a dialog box from which you can manage the list of messages
                         to be intercepted by VB Messenger.

Usage   Double click on the elipses ('...') button next to the property text to

activate the about dialog box.

Remarks	Clicking on the elipses in the property window display the Message Selector dialog box. The Message Selector allows you to add and remove standard and custom messages to the MessageList Property array at design time.
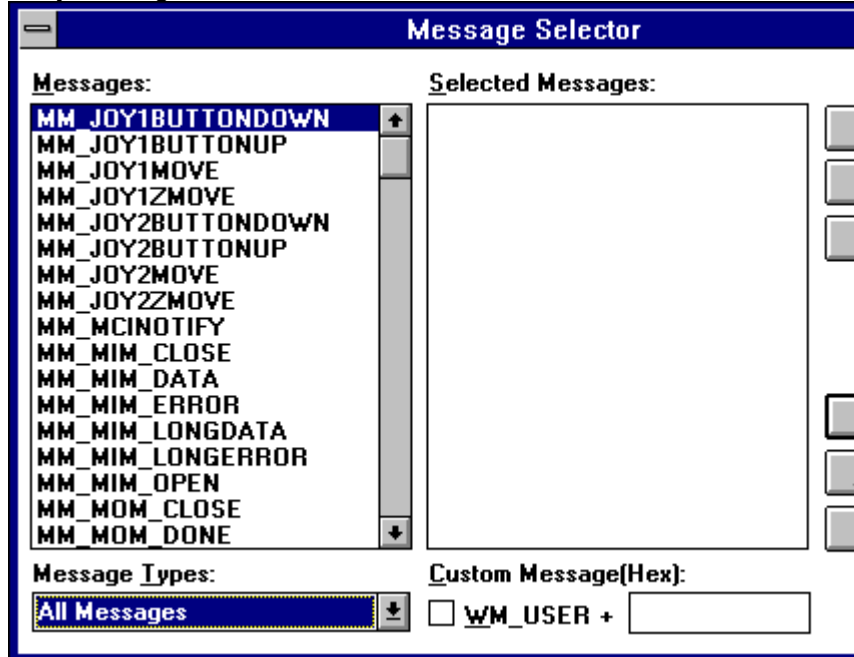
The **Messages** list box contains all available standard messages. The **Selected Messages** list box on the right contains your selected messages to be intercepted.   Clicking on the **Add**, **Remove** or **Clear** button allows you to manage the selected messages.

You can filter the types of standard messages to be displayed in the **Messages** list box by selecting from the **Message Types** drop down list.

You can specify a custom message that does not appear in the **Messages** list of standard messages by entering the message value (in hex) into the **Custom Message** edit box.   Optionally, you can check the **WM_USER +** check box to add the value of WM_USER to the entered custom message.

Available only at design time.

Data Type	N/A
See Also	MessageList, MessageCount, AddMessage, RemoveMessage, ClearMessage

MessageText Property

Description	Converts a message value to the corresponding text (i.e., "WM_PAINT") as defined by the Windows 3.1 SDK.
Usage	[*form.*]*VBMsg*.**MessageText(***message&)*
Remarks	This property array contains all the messages in literal form. You can specify the message description to retrieve by indicating the message

value as the index to the property array.

Example:

```
Const WM_CLOSE = &H10

X$ = VBMsg.MessageList(WM_CLOSE)
' X$ now equals "WM_CLOSE"
```

Available only at runtime and is read only.

Data Type   String

MessageTypes Property

Description   Allows you to specify how VB Messenger interprets the MessageList property.

Usage   [*form.*]*VBMsg.***MessageTypes**[= *setting%*]

Remarks   Use the MessageTypes property to instruct VB Messenger when to fire an event in accordance to when the messages in the MessageList property array are detected.

The MessageTypes settings are as follows:

Setting	Description
0	Intercept selected messages in the MessageList property only.
1	Intercept all messages (ignore MessageList property).
2	Do not intercept any messages.
3	Intercept all messages except those selected in the MessageList property array.

Data Type   Integer (Enumerated)

PostMessage Property

Description   Posts a message to the Windows message queue for the subclassed window.

Usage   [*form.*]*VBMsg.***PostMessage**[= *message&*]

Remarks   Setting this property will cause VB Messenger to post the specified message for the subclassed window to the Windows message queue. VB Messenger uses the properties wParam and lParam as the 16-bit word and 32-bit long parameters for the message.   The return value of the posted message can be obtained in the ReturnVal property.

If the SubclasshWnd property is not set, no message will be posted.

Example:

```
Const WM_CLOSE = &H10

VBMsg1.wParam = 0
VBMsg1.lParam = 0
```

```
VBMsg1.PostMessage = WM_CLOSE
```

Available only at runtime and is write only.

Data Type	Long

RemoveMessage Property

Description	Removes a message from the MessageList property.
Usage	[*form.*]*VBMsg.***RemoveMessage**[= *index%*]
Remarks	Setting this property to the message value of a message in the MessageList property array will remove the message from the list.

Example:

```
' Remove message from the MessageList property
VBMsg1.RemoveMessage = WM_CLOSE
```

Available only at runtime and is write only.

Data Type	Long
See Also	ClearMessages, AddMessage

SendMessage Property

Description	Sends a message directly to the subclassed window bypassing the message queue.
Usage	[*form.*]*VBMsg.***PostMessage**[= *message&*]
Remarks	Setting this property will cause VB Messenger to send the specified message directly to the subclassed window. VB Messenger uses the properties wParam and lParam as the 16-bit word and 32-bit long parameters for the message. The return value of the message can be obtained in the ReturnVal property.

If the SubclasshWnd property is not set, no message will be sent.

Example:
```
' select a range of items in a multi-select list box
VBMsg1.wParam = True
VBMsg1.LoWord = 0
VBMsg1.HiWord = List1.ListCount
'The concatenated value is now in lParam property

VBMsg1.SendMessage = LB_SELITEMRANGE
```

Available only at runtime and is write only.

Data Type	Long

ReturnValue Property

Description	This property is set with the return value of the SendMessage or PostMessage property.
Usage	[*form.*]*VBMsg.***ReturnVal**
Remarks	**Example:**

```
' This example sends an EM_GETLINECOUNT message to
' retrieve the number of lines in a multiline edit
' control and then sends an EM_LINESCROLL message
' to scroll the edit control so that the last line
' is displayed at the top of the edit control.

VBMsg1.SendMessage = EM_GETLINECOUNT

' Number of lines returned can be found in the
' ReturnVal property.
VBMsg1.wParam = 0
VBMsg1.LoWord = VBMsg1.ReturnVal - 1
VBMsg1.HiWord = 0
VBMsg1.SendMessage = EM_LINESCROLL
```

Available only at runtime and is read only.

Data Type	Long
See Also	SendMessage, PostMessage

String Property

Description	This property is set with the resulting string after setting the lParam2String property. Setting this property returns an address which can be found in the lParam property.
Usage	[*form.*]*VBMsg.***String**
Remarks	If you set this property to a string, VB Messenger will return an address of the string in the lParam property. The address of the string will be valid for as long as VB Messenger is active or the string is replaced with a new string.

Example:
```
'... from within the VBMsg1_WindowMessage proc ...
            .
            .
VBMsg1.lParam2String = lParam
Print "The resulting string is: " & VBMsg1.String
```

Available only at runtime.

Data Type	String
See Also	lParam2String, lParam

SubclasshWnd Property

Description	Set this property to the window handle (hWnd) of the form or control to subclass.
Usage	[*form.*]*VBMsg.***SubclasshWnd** = [*handle%*]
Remarks	Setting this property to a valid window handle immediately activates VB Messenger. All messages sent to the window associated with the handle from that point onward will be filtered by VB Messenger and the event WindowMessage will be fired for each.
	Setting this property to zero will automatically disable the subclassing. Upon the destruction of the window (WM_DESTROY), this property is cleared and subclassing will terminate.
	Available only at runtime.
Data Type	Integer

wParam Property

Description	This property represents the 16-bit integer value of the Windows message structure.
Usage	[*form.*]*VBMsg.**w*Param**[= *value%*]
Remarks	This property is used primarily in conjunction with the SendMessage and PostMessage properties as a parameter for sending messages directly to a subclassed window.
	Available only at runtime.
Data Type	Integer
See Also	lParam, SendMessage, PostMessage

Events Reference

The following is a detailed reference of all the properties supported by VB Messenger.

Related Topics:

WindowMessage Event
WindowDestroyed Event

WindowMessage Event

Description	VB Mesenger fires this event each time one of the selected messages is detected for the subclassed window.
Syntax	**Sub VBMsg_WindowMessage** (*hWindow* **As Integer**, *Msg* **As Integer**, *wParam* **As Integer**, *lParam* **As Long**, *RetVal* **As Long**, *CallDefProc* **As Integer**)
Remarks	When a message that you wish to intercept is detected for the subclassed form, VB Messenger fires this event passing the message's parameters to the event procedure.

Parameter	Description
hWindow	Identifies the subclassed window.
Msg	The message that was detected.
wParam	The 16-bit word value associated with the message.
lParam	The 32-bit long value associated with the message.
RetVal	After you process the message, use this parameter if you wish to return a value to VB Messenger and bypass the default windows procedure. VB Messenger will then use this value as the return value.
CallDefProc	If this value is True, VB Messenger will call the default windows procedure for the subclassed control. If it is False, VB Messenger will not call the default procedure and return the specified return value in RetVal.

See Also	WindowDestroyed Event

WindowDestroyed Event

Description	VB Mesenger fires this event unconditionally if the subclassed window is sent a WM_DESTROY message.
Syntax	**Sub WinMsg1_WindowDestroyed** (*hWindow* **As Integer**)
Remarks	This event is fired when the subclassed window is destroyed. This is useful for code to clean up memory that you may have associated with the subclassed window. The window is automatically unhooked after this event is fired.

Parameter	Description
hWindow	Identifies the subclassed window.

See Also	WindowDestroyed Event

VB Messenger API

Overview

VB Messenger comes with a set of API functions that you will   need to process certain messages.

Several Windows messages require the programmer to be able to access data via pointers.   Although in languages like C it is possible to provide pointers, it is not possible using Visual Basic.   These API functions allow you to access data while "faking" pointers.   VB Messenger uses long integers to represent the pointers.   Since pointers are actually just 32-bit numbers (i.e., long integers), you can actually "fake" Windows by sending certain WIndows API functions the long integer equivilent of a pointer as supplied by the VB Messenger API.

CAUTION: These routines require the use of pointers.   Take care when using such routines as they may cause unpredictable results if used improperly.   Do not pass invalid addresses to these routines.   Doing so may cause a GPF or loss of data.

Special Note: You may notice that the Lib in the Declare statements below refer VBMSG.VBX.   Why is this not referencing a DLL?   Since a custom control (VBX) is really a DLL with special routines in so that Visual Basic can access it, functions can be called externally from them.   So rather than supplying a separate DLL that you would need to include with your distribution, VB Messenger comes with a full set of functions built right into itself.   All of the following functions can be called directly from the file VBMSG.VBX.

As with all DLLs and VBXs, the executable file must be either in the path, the current directory, or the Windows SYSTEM directory in order for Visual Basic to find and load them.   See the Chapter 22, "*Calling Procedures in DLLs*" in the *Microsoft Visual Basic Programmer's Guide* for a further description on calling external procedures.

API Reference

The following section details the API functions available within VB Messenger.

Related Topics:

ptGetIntegerAddress, ptGetLongAddress, ptGetStringAddress

Description	Returns the address of a Visual Basic variable with stricter type checking.
Declarations	Declare Function **ptGetIntegerAddress** Lib "VBMSG.VBX" Alias "ptGetVariableAddress" (ByVal var As Integer) As Long
	Declare Function **ptGetLongAddress** Lib "VBMSG.VBX" Alias "ptGetVariableAddress" (ByVal var As Long) As Long
	Declare Function **ptGetStringAddress** Lib "VBMSG.VBX" Alias "ptGetVariableAddress" (ByVal var As String) As Long
Remarks	Each of these functions return a 32 bit address of a variable. These function declarations are provided to give better parameter checking when using this function.

Parameter	Description
var	The variable to get the address of. Can either be an integer, long, or string depending upon the declaration used. For Types, use the ptGetVariableAddress API.

Return Value	These functions returns a 32-bit address of the variable.

ptGetIntegerFromAddress

Description	Returns the 16-bit integer value from the data at the address specified.
Declaration	Declare Function **ptGetIntegerFromAddress** Lib "VBMSG.VBX" (ByVal address As Long) As Integer
Remarks	

Parameter	Description
address	The 32-bit far address of the integer.

Return Value	Returns the integer value associated with the address.

ptGetLongFromAddress

Description	Returns the 32-bit long integer value from the data at the address specified.
Declaration	Declare Function **ptGetLongFromAddress** Lib "VBMSG.VBX" (ByVal address As Long) As Long
Remarks	

Parameter	Description
address	The 32-bit far address of the long integer.

Return Value	Returns the long integer value associated with the address.

ptGetStringFromAddress

Description	Returns a string from the data at the address specified.
Declaration	Declare Function **ptGetStringFromAddress** Lib "VBMSG.VBX"

(ByVal address As Long) As String

Remarks	The string located at the address specified must end with a terminating zero.

Parameter	Description
address	The 32-bit far address of the string.

Return Value	Returns the string associated with the address.

ptGetTypeFromAddress

Description	Returns a Type structure from the data at the address specified.
Declaration	Declare Sub **ptGetTypeFromAddress** Lib "VBMSG.VBX" (ByVal address As Long, typevar As Any, cbBytes As Integer)
Remarks	The data located at the address specified will be copied into the Type structure variable defined by the calling program. Only the number of bytes specified will be copied. Do not specify more bytes then are actually allocated. Doing so may produce unpredictable results such as a GPF or loss of data.

Parameter	Description
address	The 32-bit far address of the data.
typevar	The user defined Type variable to copy the data into.
cbBytes	The number of bytes to copy.

ptGetVariableAddress

Description	Returns the address of a Visual Basic variable.
Declaration	Declare Function **ptGetVariableAddress** Lib "VBMSG.VBX" (variable As Any) As Long
Remarks	This function returns a 32 bit address of any variable or Type. Any type of variable can be used.

Parameter	Description
variable	The variable or Type to get the address of.

Return Value	This function returns a 32-bit address of the variable or Type.

ptHiWord

Description	This function parses out the high-order 16-bit word value of a 32-bit long integer.
Declaration	Declare Sub **ptHiWord** Lib "VBMSG.VBX" (ByVal lParam As Long) As Integer
Remarks	This API provides the same functionality as the property HiWord.

Parameter	Description

	lParam The 32-bit long integer value to parse.
Return Value	Returns a 16-bit integer representing the high-order of the 32-bit long value.

ptLoWord

Description	This function parses out the low-order 16-bit word value of a 32-bit long integer.
Declaration	Declare Sub **ptLoWord** Lib "VBMSG.VBX" (ByVal lParam As Long) As Integer
Remarks	This API provides the same functionality as the property LoWord.

Parameter	Description
lParam	The 32-bit long integer value to parse.

Return Value	Returns a 16-bit integer representing the low-order of the 32-bit long value.

ptMakelParam

Description	This functions creates an unsigned long integer for use as an lParam parameter in a message by concatenating two integer values, specified by the wLow and wHigh parameters.
Declaration	Declare Sub **ptMakelParam** Lib "VBMSG.VBX" (ByVal wLow As Integer, wHigh As Integer) As Long
Remarks	

Parameter	Description
wLow	Specifies the low-order word of the new long value.
wHigh	Specifies the high-order word of the new long value.
address	The 32-bit far address of the data.

Return Value	The return value specifies a long-integer value.

ptMakeUShort

Description	This function converts a signed integer value to an unsigned integer value and returns it as a long integer.
Declaration	Declare Sub **ptMakeUShort** Lib "VBMSG.VBX" (ByVal ushortVal As Integer) As Long
Remarks	

Parameter	Description
ushortVal	The signed integer value to convert.

Return Value	The function returns a long integer value representing the unsigned integer.

ptMessageToText

Description	Returns the literal description of a message number as define by the Windowss SDK.

Declaration	Declare Function **ptMessageToText** Lib "VBMSG.VBX" (ByVal message As Integer) As String
Remarks	This routine translates the message number to the literal text description of the message. This function is useful in developing a diagnostic program that detects all messages for a specific window and displays the message as text (i.e., WM_PAINT instead of &H10).

Parameter Description

Parameter	Description
message	The message number.

Return Value	The message string.