CLACKER.VBX Custom Control

Description

This control allows the developer to provide the user with a dynamic menu help system. The control provides a custom event that defines when the user selects a menu item. The control can be used for displaying a help phrase for all menus in a program or for any combination of menus.

Place a CLACKER Control on the form to be monitored for menu selection events. For a Multiple Document Interface (MDI) program, place the CLACKER control on the menu status label of the MDI form, not a child form. You cannot place the CLACKER control directly on the MDI form.

A ClackerClick() event will be issued for each menu that is highlighted. Only ONE control should be used with each form that menu events are to be monitored. A separate help text string array will be needed for each menu within an application to be monitored. See Tips and Techniques for more information.

CLACKER.VBX has several ways of being used:

In each usage hook up the control with the appropriate command

A simple approach is to match the returned property "MenuCaption" from the ClackerClick() event, to the corresponding MenuCaption property in the application. A Select Case Statement block is used for this.

A second technique is to store the help text to display in each menu item's Tag property. This Tag text is returned in the TagText value in the ClackerClick() event. When the ClackerClick() event is triggered, display the TagText in the appropriate form control. See the ClackerClick() event for an example.

A more advanced usage of CLACKER.VBX is to use the Windows API parameters hMenu and MenuID returned in the ClackerClick() event. This usage will monitor all menu items in the application, and allow some more advanced menu techniques to be used. Using these parameters imposes the demand that the application programmer understand the menu system of windows and how/when the menu positions change and when to reload the position arrays. See the demo program provided and the Windows SDK manuals for more information.

The control usage variation options are dependent on the way the application wants to convey help status text to the user. Depending on this choice, the way the application stores the help text and associates this text with the returned menu parameters may change. The CLACKER.VBX control will always return the proper menu parameters in the ClackerClick() event for all menu items selected. What the application does with these returned parameters and how the help text is presented to the user is up to the programmer.

File Name

CLACKER.VBX

Remarks

When you create and distribute applications that use the CLACKER control you should install the file CLACKER.VBX in the customer's Microsoft Windows \SYSTEM sub directory. All of the properties, events, and methods for this control are listed below. Properties and events that apply only to this control, or require special considerations when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or on-line Help for documentation of the remaining properties, events, and methods.

Properties

Action

CtlName

HwndForm

Index

Left

MenuIDArray

RefreshHwnd

SystemMenu

Top

Tag

Events

ClackerClick

ClearStatusClick

RefreshHwndClick

Methods

The Action property is used as a pseudo method.

Tips and Techniques

Tips and Techniques

1. Simple menu status

To supply menu help use a code construct as follows:

```
FOR VB2 and VB3 only, use the following code
If MenuName = "" And MenuCaption = "" Then
        ' it is a system menu or a separator bar
        ' clear the status text if you want to
        Label1.Caption = ""
ElseIf MenuName = "" Then
        ' detects a system menu item
        Label1.Caption = TXT_Result.Text
Else
        ' regular VB menu item
        Label1.Caption = TagText
End If

FOR VB1 use the following code:
If MenuCaption != "" Then
        Select Case MenuCaption
                Case FirstMenuCaption
                        Label1.Caption = "Text to display"
                Case SecondMenuCAption
                        Label1.Caption = "Text to display"
                ...
        End Select
End If
```

The example program supplied as part of the package explains the code more fully.

Note:

1. The Visual Basics caption property does not give the complete caption string, it only contains the menu text, not the accelerator text. The string returned from the CLACKER control is in the same format as the VB menu caption property.

2. VB1 does not return the MenuName and TagText parameters in the ClackerClick() event. Only the MenuCaption, hMenu and MenuID are available in VB1. The other parameters are NULL.

2. Debugging:

The debugging version of CLACKER will most likely resolve the errors encountered when developing your application. In some circumstances the following faults may result during development. Some precautions are necessary when using a system modifying control like CLACKER.

3. Setting Debug BreakPoints:

If the VISUAL BASIC application under test has a break point set in the ClackerClick() return event, further menu operation is prevented since the CLACKER.VBX is stopped at the break point in the VISUAL BASIC application. The VISUAL BASIC development environment and the application under test are behaving as one application during debugging. Plan the debug strategy of the ClackerClick() event code accordingly. Using Debug.Print statements is the best way to debug the ClackerClick event subroutine.

4. Running In The Design Mode:

Another application cannot use the CLACKER.VBX when the VISUAL BASIC development environment is being used with an application using CLACKER.VBX. Close all applications making use of CLACKER.VBX before starting the development of a VISUAL BASIC application using CLACKER.

5. MDI Applications:

In MDI applications menus can be changed when the state of the active form changes. When using the hMenu or MenuID parameters returned from the ClackerClick() event, make use of the CLACKER.VBX command RefreshHwnd to reload the menu arrays as required.

Use a separate control for the MDI form and for each Child form that is to be monitored. The control cannot be placed directly on a MDI form. Place the MDI form's control on a toolbar or status bar.

6. Getting Menus For Variable "Windows Menu":

VISUAL BASIC has a feature to use a "WINDOWS" menu item with a MDI form's menu. This menu item is a toggle for switching to open child windows. To include the window items in the status text displayed take these steps. The example uses a Case Select statement block. In the Case Select statement block every menu item should display the status text and then EXIT the ClackerClick() subroutine. The Case Else statement handles the extra expanding menu items. Use a statement of the form

```
StatusText.Text = "Switch to the Window" + MenuCaption
```

in the Case Else statement to supply status text for these items. This points the user to the menu item with the proper caption.

7. Getting Menu IDs

Use the Internal RefreshHwnd functions to set up menu IDs. Unless the menu system is thoroughly understood, incorrect results may result. The Microsoft SDK documentation has more details on this subject.

8. Menu IDs

Each application has unique MenuIDs for all of its menu items. You must build an array of MenuIDs for each window with a menu that you want to display help text for. All menu items, including unused separators and hidden menus must have a MenuID entry and a corresponding entry for help text in the help text array. Otherwise the indexing will not be proper, and you will display the wrong help text for the menu item.

9. MDI applications

Finding the appropriate point in an applications init cycle to load CLACKER, and fill the MenuID array is critical for successful use of dynamic menu help. MDI applications present a real challenge in this regard because the menu system switching can be confusing to the designer and the end user...might explain why users find menu help prompting an invaluable feature. When MDI applications run, the currently

open child window with focus, substitutes it's menu for the MDI window's menu. CLACKER transparently handles this switching. When that window is closed, the MDI frame places the next open child window's menu on the menu bar. The menus can be the same, or they can be unique to each child window. If there are no open child windows, the MDI frame places the default MDI menu on the menu bar.

Usage of the property "RefreshHwnd" to generate a new array of position values at appropriate times is a way to minimize the code in the VB application to accommodate this menu switching.

IMPORTANT: Be aware of all the menu switching that is going on. Each child window has its own unique set of MenuIDs for that child instance. Therefore each MDI child window with a menu must have its own array of MenuIDs for matching. It may be necessary to experiment with the MenuID array loading to find a point in the initiation sequence for your application where the menu is loaded and the MenuIDs are stable. Use the VB DoEvents() function to clear out the Windows event queue if necessary before loading the MenuID array. Additionally, you must obtain the MDI MenuIDs when only the MDI menu is loaded, that is, when no child windows are open.

10. Help Text Arrays

You need only one text array for each child window type. The main MDI window will also need a MenuID array and a unique help text array. You can determine the active window type, and therefore the proper help text array to use by using the following pseudo code;

```
If "NoChildWindowsOpen" Then
      ...process the MDI menu help text array
elseif TypeOf frmMDI.ActiveForm Is "MyForm" Then
      ...do search in help text array type1
elseif TypeOf frmMDI.ActiveForm Is "MyForm2
      ...do search in help text array type1
...
```

The switching of the menu and returning the proper MenuID is automatic from CLACKER. Reloading the local VB arrays and indexing the text strings must be handled by the VB application.

HwndForm Property, CLACKER Control

Description

Sets the Hwnd of the form to monitor menu selections. Once hooked up, the menu monitoring should be left installed for the duration of the application.

Usage

[*form*.]*CLACKER1*.**HwndForm**[= *setting* %]

Settings

The HwndForm Property settings are:

Setting	Description
0	(Default) Control does not do anything. Setting the Action property has no effect if a proper Hwnd is not registered first.
Form.hWnd	The Hwnd of the form whose menu is to be monitored for selections.

Data Type

Integer (Hwnd)

RefrsehHwnd Property, CLACKER Control

Sets the HWND property of the form to use to generate the MenuIDArray. Setting this property will result in the CLACKER.VBX control enumerating the menu items in the form, and returning the event RefreshHwndClick() when complete. The event returns a parameter "MenuItemCount," the size of the MenuIDArray. A local VB array should be loaded with the MenuIDArray data in the Sub RefreshHwndClick(). The local VB array should be dimensioned to contain LONG variables.

These values are provided for advanced usage of the CLACKER control. See the Windows SDK documentation for more information.

Description

Sets the HWND to notify the CLACKER Control of the calling form to return the menu MenuIDArray for.

Usage

[*form*.]*CLACKER1***.**RefrsehHwnd [= *setting* %]

Settings

The RefrsehHwnd Property settings are:

Setting	Description
Form.hWnd	The Hwnd of the form whose menu is to be enumerated for MenuIDs.

Data Type

Integer (Hwnd)

MenuIDArray Property Array, CLACKER Control

DESCRIPTION

The MenuIDArray is filled in response to the setting of the property RefreshHwnd.

The values returned in the array are the resource IDs of the menu items. These values correspond to the MenuID parameter returned in the ClackerClick() event.

The length of the array is the MenuItemCount value returned in the event RefreshHwndClick().

The MenuIDArray values are valid only during the event RefreshHwndClick().A single array is returned for each setting of the RefreshHwnd property. The values in the property array must be transferred to a local VB array for storage during the return event RefreshHwndClick().

Array values start at position 1 and run through MenuItemCount.

Usage

ElementValue = [*form*.]*CLACKER1*.MenuIDArray[ElementNumber]

Code fragment to fill the local VB array:

```
Dim LocalVBArray() As Long

Sub Clacker1_RefreshHwndClick( MenuItemCount As Integer )
Redim LocalVBArray( 1 To MenuItemCount )
For ndex = 1 To MenuItemCount
        LocalVBArray( ndex ) = Clacker1.MenuIDArray( ndex )
Next ndex
```

Settings

Read only data.

Data Type

LONG (MenuID).

The data returned is a USHORT data type but since VB has no Unsigned Integer data type the return data is converted to a long and stored in the LocalVBArray().

Action Property, CLACKER Control

Description

Setting the Action property cause the control to attach or detach from the forms menu. The control must be attached (hooked) to the forms menu before the ClackerClick event can be provided.

The Form's Hwnd that the control is to monitor is sent immediately before the Action call.

Usage

[*form*.]*CLACKER1*.**Action**[= *setting* %]

Settings

The Action property settings are:

Setting		Description
ID_START	= 1	Attach to the form's menu.
ID_STOP	= 2	Detach from a form's menu.

Remarks

The code fragment to begin monitoring the menu for selection events is

```
Form.Clacker.HwndForm = Form.hWnd
Form.Clacker.Action = ID_START
```

The code fragment to stop monitoring the menu is,

```
Form.Clacker.HwndForm = Form.hWnd
Form.Clacker.Action = ID_STOP
```

Initiate the menu monitoring action in the start up code of the program for each form menu to be monitored. Terminate the menu monitoring action for each form monitored in the form's unload event or when the instance of the program terminates. When the program terminates, each CLACKER control must be terminated separately. Failure to start and stop the monitoring action properly of each CLACKER control used in an application instance can result in unpredictable behavior, and may result in an unstable Window's session.

Data Type

Integer

SystemMenu Property, CLACKER Control

Description

Allows selection of the inclusion of system menus in the MenuIDArray. If it is desired to provide help text for system menu(s), set this property to TRUE. Otherwise set it to FALSE.

Setting the SystemMenu property to TRUE includes the system menu(s) in the RefreshHwnd menu searching process. The MenuIDArray property array returned in the event RefreshHwnd() includes all system menu items found.

Setting the MenuIDArray property FALSE excludes the system menu(s) from the MenuIDArray returned in the event RefreshHwnd()

Usage

[*form*.]*CLACKER1*.SystemMenu [= *setting* %]

Settings

BOOLEAN, TRUE or FALSE.

Remarks

Data Type

Integer

ClackerClick Event, CLACKER Control

Description

This event is generated when the user selects a menu from the system menu or the form's menus. The event is generated prior to the actual menu choice and does not interfere with the menu event generation.

A normal application menu event is passed to the VB application when the actual menu item is selected for execution.

Syntax

Sub *CLACKER1*_**ClackerClick** (MenuName **As** String, TagText **As** String, MenuCaption **As** String, MenuID **As** Integer, hMenu **As** Integer)

Remarks

The MenuName is the Name property of the menu item selected.

The TagText is the Tag property of the menu item selected.

The MenuCaption is the Caption property of the menu item selected.

The hMenu is the SDK referenced menu handle of the menu item selected.

The MenuID is the SDK referenced menu item ID of the menu item selected.

In all cases when a menu is selected, a unique MenuID is returned from CLACKER. Windows uses a linked list for storing menu items, so it is advised that you read the SDK manuals on menus before making use of the hMenu and MenuID parameters.

Usage

Normal Usage of returned parameters:

To supply menu help use a code construct as follows:

```
FOR VB2 and VB3 only, use the following code
If MenuName = "" And MenuCaption = "" Then
        ' it is a system menu or a separator bar
        ' clear the status text if you want to
        Label1.Caption = ""
ElseIf MenuName = "" Then
        ' detects a system menu item
        Label1.Caption = TXT_Result.Text
Else
        ' regular VB menu item
        Label1.Caption = TagText
End If

FOR VB1 use the following code:
If MenuCaption != "" Then
        Select Case MenuCaption
                Case FirstMenuCaption
```

```
                Label1.Caption = "Text to display"
            Case SecondMenuCAption
                Label1.Caption = "Text to display"
            ...
        End Select
    End If
```

Advanced Usage of returned parameters:

The MenuID parameter is normally used as the "key" for searching an array of help text strings. By storing the MenuID at start-up of the form and then searching the array for a matching MenuID when the ClackerClick event is generated, the help text can be displayed in the appropriate way. The MenuID must be converted to an Unsigned Integer before using. The following code fragment does this.

```
If ( MenuID < -1 ) Then
    LongMenuID = MenuID + 65536
Else
    LongMenuID = MenuID
```

The value -1 is used instead of 0 because -1 has special meaning as the identifier of a sub menu item.

See the provided demo program provided for more information on how to set up and retrieve the help text. Pay particular attention to the way in which the menu system is interrogated for the initial MenuID parameters. A recursive lookup function is used to parse the menu linked list into a linear array of MenuIDs. This function is encapsulated in the command call to the CLACKER property RefreshHwnd.

RefreshHwndClick Event, CLACKER Control

This event is generated by the CLACKER.VBX in response to the setting of the RefreshHwnd property with a valid form Hwnd. The parameter returned corresponds to the size of the array to be used to store the position parameters.

The local form's position array should be filled during this event.

IMPORTANT: SINCE VB HAS NO UNSIGNED SHORT VARIABLE TYPE USE A LONG INTEGER FOR THE LOCAL VB ARRAY VARIABLE TYPE.

Description

Returns the menu position count. The CLACKER.VBX property array "MenuIDArray" contains the position data to store locally. The array is accessed sequentially to transfer the data from the CLACKER.VBX array to local storage.

It is not necessary to hook up the menu before using the Clacker1.RefreshHwnd command. This event will produce the same results in both cases.

The Clacker1.RefreshHwnd command can be used for the same form as many times as necessary. It may be necessary to re-issue the Clacker1.RefreshHwnd command and reload the arrays in the program as the menu structures may be changed, added to or deleted.

Syntax

Sub *CLACKER1_***RefreshHwndClick**(MenuItemCount As Integer)

Remarks

Using this event is optional.

Usage

Event is returned in response to the setting of the RefreshHwnd property with a valid form Hwnd. If the "RefreshHwnd" property is not valid, no event will be returned.

The value of MenuItemCount is the number of menu items found if the Hwnd has a menu, it is zero if the Hwnd was invalid or the Hwnd did not have a menu. Test the value of MenuItemCount before using to determine the appropriate action.

ClearStatusClick Event, CLACKER Control

Description

This event can be used by the VB application to clear the status line when a menu operation is actually initiated, or the pending menu action is canceled. Removing the text from the status line can give an effective visual queue to the user that the actual menu command has been initiated.

Each menu command or cancel action will generate this event.

Syntax

Sub *CLACKER1*_ClearStatusClick ( )

Remarks

Using this event is optional.

Usage

Zero out the status text by doing a instruction like, StatusLine.Text = "" in this Sub().