Clacker Control

Description

This control allows the user to a menu help system. The control provides a custom event which 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 application. A separate help text string array will be needed for each menu within an application to be monitored. See Hints and Tips for more information.

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 consideration 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
hwndForm
CtlName
Index
Left
Top
Tag

Events

ClackerClick

Methods

The Action property is used as a pseudo method.

Typical Problems

Hints and Tips

Typical Problems

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 systems modifying control like CLACKER.

1. The Visual Basic development environment does not return after terminating the program under development is generally caused by not properly unhooking the menu after use and before program unload. Since the development environment and the application share functions during debug, the system may hang if this condition occurs. A "hardware" re-boot will be necessary to clear the fault. Check to make sure that when you terminate the application, either by proper user action or by fault handling the CLACKER control is ended. If you are using the debug version, a dialog box will display showing proper termination action.

2. Program under development crashes, leaving the system in an unstable state. After a GPF a "hardware" re-boot is the only proper way to restore normal operation if the CLACKER control is being used. See above.

3. When selecting "Debug-End" while in the VISUAL BASIC development environment, the application under test is not terminated properly, i.e. an abrupt end is issued terminating all further processing without allowing the application to clean-up. This means that DLLs and VBXs will be left in an indeterminate state. To reset the DLL/VBX you should either terminate the application properly, or do a "Hardware-re-boot" after the forced development environment Debug-End. This will insure the DLL/VBX is reset properly.

Hints and Tips

1. Getting Menu IDs

Most complications with CLACKER are with getting the proper MenuIDs loaded in a MenuID array to use as indexing into the help text array. Since you want to translate a CLACKER returned MenuID to a help text position, complications can arise. The key thing to remember is that the menus are stored sequentially in a linked list structure in Windows. To get all of the entries, make sure you walk the entire menu list for your application. It is best to run the MenuSetup() sub in the demo application to see how this works. port this code to your application and run it to see how many menu items are present. Build your help text array the size and in the sequence that the MenuSetup() sub returns items. This way keeping track, searching and matching MenuIDs to help text is easier.

As an aid in helping you set up the MenuID array, experiment with no help text and print the CLACKER ClackerClick variables to the Menu Status bar. This will allow you to see what the underlying menu structure for you application looks like.

2. 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.

3. 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.

IMPORTANT: Be aware of all the menu switching that is going on. Each child window has it's own unique set of MenuIDs for that child instance. Therefore each MDI child window with a menu must have it's 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.

4. 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.

hwndForm Property, Clacker Control

Description

Sets the hWnd to notify the Clacker Control of the calling form.

Usage

[*form*.]*CLACKER*.**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)

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*.]*CALCKER*.**Action**[= *setting* %]

Settings

The Action property settings are:

Setting	Description
ID_START = 1	Attach to the forms 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

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 menu event is generated when the actual menu item is selected.

Syntax

Sub *CLACKER*_**ClackerClick** (hMenu **as Long**, MenuID **as Long**, MenuCaption **As String**)

Remarks

The hMenu is the menu handle of the selected menu. The MenuID is the menu item's ID. The MenuCaption is the caption of the selected menu item.

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 MenuCaption parameters.

Usage

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.

See the provided demo program 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.