



SubClass-EZ Help Contents

Introduction

Properties

HwndParam

Other

Events

AfterDefProcess

MsgQueue

Functions

LoWord

HiWord

MakeLong

GetControlHwnd

HwndParam Property

Description

Sets the form or control to be subclassed. *Read-Write at runtime only. Read is of last window subclassed by the control.*

Usage

[form.]SubClass1.HwndParam = [form.][control.]hWnd

Remarks

Any number of forms and controls (up to 500) can be subclassed with a single SubClass-EZ control.

Data Type

Integer

Example

The example subclasses a form, adds an 'About' menu item to the system menu, then traps the WM_SYSCOMMAND message when that item is selected. **Note:** This is the recommended way to use a MsgBox or InputBox with this control. *Neither should be used in a MsgQueue event procedure.*

```
Sub Form_Load()
    Dim hSysMenu%, n%
    hSysMenu = GetSystemMenu(hWnd, 0)
    n = AppendMenu(hSysMenu, MF_SEPARATOR, 0, "")
    n = AppendMenu(hSysMenu, MF_STRING, IDM_ABOUT, "&About...")
    SubClass1.HwndParam = hWnd
End Sub

Sub SubClass1_MsgQueue (wnd As Integer, msg As Integer, wp As Integer, lp As Long, retval As Long, doDef As Integer)
    If msg = WM_SYSCOMMAND and wp = IDM_ABOUT Then
        doDef = True
    End If
End Sub

Sub SubClass1_AfterDefProcess (wnd As Integer, msg As Integer)
    MsgBox "This is an about box", 0, "About"
End Sub

'Declares and Constants
Declare Function GetSystemMenu% Lib "User" (ByVal hWnd%, ByVal bRevert%)
Declare Function AppendMenu% Lib "User" (ByVal hmenu%, ByVal wFlags%, ByVal wIDNewItem%, ByVal lpNewItem As Any)
Const WM_SYSCOMMAND = &H112
Const MF_SEPARATOR = &H800
Const MF_STRING = &H0
Const IDM_ABOUT = 15
```

AfterDefProcess Event

Description

Event triggered by changing the 'doDef' value in a MsgQueue event. Fired after default processing of a message.

Syntax

Sub **SubClass1_AfterDefProcess** (*wnd* As Integer, *msg* As Integer)

Remarks

The AfterProcess event uses these arguments:

Argument	Description
<i>wnd</i>	hWnd of window that processed message.
<i>msg</i>	message processed.

Example

The example subclasses a form and changes the system menu bitmap after default processing of the WM_NCACTIVATE and WM_NCPAINT messages. To setup place a picture control on a form, load an icon into its picture property, then set its autoredraw = true, autosize = true, borderstyle = none, scalemode = pixels, visible = false.

```
Sub Form_Load
    SubClass1.HwndParam = hWnd
End Sub

Sub SubClass1_MsgQueue (wnd As Integer, msg As Integer, wp As Integer, lp As Long, retval As Long, doDef As Integer)
    Select Case msg
        Case WM_NCACTIVATE, WM_NCPAINT
            doDef = True
    End Select
End Sub

Sub SubClass1_AfterDefProcess (wnd As Integer, msg As Integer)
    Dim dc%
    dc = GetWindowDC(hWnd)
    StretchBlt dc, GetSystemMetrics(SM_CXFRAME), GetSystemMetrics(SM_CYFRAME), GetSystemMetrics(SM_CXSIZE),
    GetSystemMetrics(SM_CYSIZE), Picture1.hDC, 0, 0, Picture1.ScaleWidth, Picture1.ScaleHeight, SRCCOPY
    ReleaseDC hWnd, dc
End Sub

'Declares and Constants

Declare Function GetSystemMetrics% Lib "User" (ByVal nIndex%)
Declare Function GetWindowDC% Lib "User" (ByVal hwnd%)
Declare Sub StretchBlt Lib "GDI" (ByVal hdc%, ByVal x%, ByVal y%, ByVal nWidth%, ByVal nHeight%, ByVal hSrcDC%, ByVal XSrc%, ByVal YSrc%, ByVal nSrcWidth%, ByVal nSrcHeight%, ByVal dwRop&)
Declare Sub ReleaseDC Lib "User" (ByVal hwnd%, ByVal hdc%)
Const WM_NCPAINT = &H85
Const WM_NCACTIVATE = &H86
Const SM_CXSIZE = 30
Const SM_CYSIZE = 31
Const SM_CXFRAME = 32
Const SM_CYFRAME = 33
Const SRCCOPY = &HCC0020
```


MsgQueue Event

Description

Event fired for each message sent to a subclassed window (excluding WM_USER + &HC00 or greater).

Syntax

Sub **SubClass1_MsgQueue** (*wnd* As Integer, *msg* As Integer, *wp* As Integer, *lp* As Long, *retval* As Long, *doDef* As Integer)

Remarks

The MsgQueue event uses these arguments:

Argument	Description
<i>wnd</i>	hWnd of window receiving message.
<i>msg</i>	message received.
<i>wp</i>	wParam of message.
<i>lp</i>	lParam of message.
<i>retval</i>	optional return value.
<i>doDef</i>	option for event after default processing.

You may modify any of the first four arguments by simply changing their value. Changing the *retval* argument from its initial value (&HFFF) prevents the subclassed window from receiving the current message and returns the value you provide. Changing the *doDef* argument to true fires an AfterDefProcess event after default processing of a message. If you do not modify any arguments the message is passed on as is for default processing. **Important:** Do not use DoEvents, a MsgBox, or an InputBox in a MsgQueue event procedure.

Example

The example subclasses a command button and intercepts the WM_SETCURSOR message, substituting a custom cursor.

```
Sub Form_Load ()
Dim hLib%
    SubClass1.HwndParam = Command1.HWnd
    hLib = LoadLibrary("subezd.vbx") 'use "subez.vbx" for licensed version
    FreeLibrary hLib
    hCur = LoadCursor(hLib, "#500")
End Sub

Sub SubClass1_MsgQueue (wnd As Integer, msg As Integer, wp As Integer, lp As Long, retval As Long, doDef As Integer)
If msg = WM_SETCURSOR Then
    Dim hOldCur%, tCur%
    tCur = GetCursor()
    If tCur <> hCur Then
        hOldCur = SetCursor(hCur)
    End If
    retval = True
End If
End Sub
```

'Declares and Constants

```
Const WM_SETCURSOR = &H20
Declare Function SetCursor% Lib "User" (ByVal hCursor%)
Declare Function GetCursor% Lib "User" ()
Declare Function LoadCursor% Lib "User" (ByVal hInstance%, ByVal lpCursorName As Any)
Declare Function LoadLibrary Lib "Kernel" (ByVal lpLibFileName As String) As Integer
Declare Sub FreeLibrary Lib "Kernel" (ByVal hLibModule As Integer)
Dim hCur
```

Functions

LoWord

Description

Returns the loword from a long integer.

Syntax

LoWord(dWord&)

HiWord

Description

Returns the hiword from a long integer.

Syntax

HiWord(dWord&)

MakeLong

Description

Returns a long integer from two short integers.

Syntax

MakeLong(loWord%,hiWord%)

GetControlHwnd

Description

Returns the hwnd of a control.

Syntax

GetControlHwnd(ControlName)

Declares:

Declare Function **LoWord** Lib "SUBEZD.VBX" (ByVal dWord As Long) As Integer

Declare Function **HiWord** Lib "SUBEZD.VBX" (ByVal dWord As Long) As Integer

Declare Function **MakeLong** Lib "SUBEZD.VBX" (ByVal LoWord As Integer, ByVal HiWord As Integer) As Long

Declare Function **GetControlHwnd** Lib "SUBEZD.VBX" (hctl As Control) As Integer

Note: substitute "SUBEZ.VBX" with licensed version.

Introduction

What is subclassing?

Instance subclassing (as is used by SubClass-EZ) means setting up a message filter for a window. It is a relatively easy thing to do but can't be done in VB without using a dll. With a call to SetWindowLong() the address of the dll's subclass function is substituted for the address of the original window procedure for a window.

After this is done all messages intended for the subclassed window go to the subclass function. There we can modify a message, respond to a message, not pass the message or pass it along to the subclassed window using CallWindowProc().

What is SubClass-EZ?

SubClass-EZ is a generic subclass control. It makes setting up a message filter as easy as setting a property. It fires an event for each message intended for a subclassed form or control. In that (MsgQueue) event procedure you can modify the message, pass or not pass the message along, and respond before and/or after default processing (in AfterDefProcess event) of the message.

What can SubClass-EZ do?

Most importantly, it allows you easy access to messages that are not normally available with VB. Where a standard or custom control or form may be lacking exactly what you need, SubClass-EZ may be able to customize it the way you want it without the need for the CDK (Control Development Kit) or having to write a dll or having to buy an additional control each time.

Examples:

- 1) add a status bar - trap a form's WM_MENUSELECT message.
- 2) use a custom cursor - see MsgQueue event example.
- 3) have a form that stays minimized - trap a form's WM_QUERYOPEN message.
- 4) make a form a drag-n-drop client - trap the WM_DROPFILES message.
- 5) add items to the system menu - see HwndParam property example.

The SubClass-EZ demo control is fully functional (in VB design environment) so you can experiment with it to see what it can do for you.

Is there much overhead with SubClass-EZ?

Subclassing in general adds a little overhead to message processing. SubClass-EZ works efficiently but it does add some overhead. Most likely this will not be noticeable. To keep overhead to a minimum, it is best to keep the code in a MsgQueue event as small as possible, using additional SubClass-EZ controls where appropriate.

Is SubClass-EZ safe?

SubClass-EZ is safe and has been thoroughly tested and is guaranteed to work with VB1 & 2, Win3 & 3.1. There are a couple important things to remember. **1)** You should not use DoEvents in a MsgQueue or AfterDefProcess event. **2)** You should not use a MsgBox or InputBox in a MsgQueue

event. If need be use the AfterDefProcess event instead - see HwndParam property example.

SubClass-EZ automatically handles the necessary un-subclassing of a subclassed form or control when that object is destroyed.

Ordering and support info:

A single user license may be ordered on CompuServe, GO SWREG (reg. ID# 622) for \$6.50. It may also be ordered directly from the author by sending \$8.50 (includes s&h) to:

Jeff Simms,
SimPlex Software
813 Hyde Rd.
Silver Spring, MD 20902

Support is available on CompuServe. Leave questions or suggestions in the VB/Win 3rd Party section (6) of the MSBASIC forum or send e-mail to Jeff Simms, 72200,3173.

Other Properties

(About) - *about this control*

Name (or CtlName) - *control name*

Index - *array index*

Left - Top - Width - Height

not important - control is invisible at runtime

