Reviewing the Stylebits Parameters

The four parameters of stylebits are AddBit, RemoveBit, AddExtendedBit, RemoveExtendedBit. For a review of these four parameters, and an introduction to Stylebits in general, please view Stylebits - Windows.

Why a Dialog Window?

There are several reasons you might want to use a dialog window in your program. A common reason is the default button. Without stylebits, only dialog windows are capable of default buttons. From the Liberty BASIC help file:

A window of type DIALOG can contain a button with the extension ".default". If the user presses the ENTER key while the dialog window has focus, it is the same as if the button whose extension is "default" is pressed and program execution will continue at the event handler [branchLabel] for that button.

The following code is a simple program illustrating the default button.

```
    Statictext #dlg.txt,
"Press Okay to close this window.", 20, 50, 100, 20
    Button #dlg.default, " Okay ", [OkayButton], UL, 140, 100
    Open "Default Button" for Dialog as #dlg
    Wait

[OkayButton]
    Close #dlg
    End
```

The Choice: A Default Button or a Menu

Unfortunately, dialog windows do not support menus. For a long time, Liberty BASIC programmers have had to choose between a default button and a menu. Recently, Brent Thorn posted code using Stylebits that will allow a default button in a regular window. So, while you still can't program a menu in a dialog window, you can code a default button in a regular window. This means a default button AND a menu in the same window. Thanks, Brent! One peculiarity I did discover was that one of the window controls needs to be given focus prior to the activation of the default button. A simple Setfocus to a textbox (any textbox) will accomplish that. Here is a modification of Brent's code.

```
'Using a Dialog Default Button in a Regular Window
'Thanks to Brent Thorn
' http://libertybasic.conforums.com/index.cgi?board=novice&action=disp
```

```
lay&num=1134254812
'Modifications by Janet = Menu and Setfocus to #main.tbx1

    WindowWidth = 300
    WindowHeight = 350
    UpperLeftX = Int((DisplayWidth - WindowWidth)/2)
    UpperLeftY = Int((DisplayHeight - WindowHeight)/2)

    Menu #main, "&Options", "E&xit", EndDemoMenu
    Textbox #main.tbx1, 150, 50, 120, 30
    Textbox #main.tbx2, 150, 100, 120, 30
    Textbox #main.tbx3, 150, 150, 120, 30
    Textbox #main.tbx4, 150, 200, 120, 30
    Stylebits #main.default, _BS_DEFPUSHBUTTON, 0, 0, 0
    Button #main.default, "", DefaultButton, UL, -100, -100

    Open "Default Button in a Regular Window" for Window as #main
    #main "Trapclose EndDemo"
    #main.tbx1 "!Setfocus"

    Wait

    Sub DefaultButton handle$
        #main.tbx1 "!Contents? text1$"
        #main.tbx2 "!Contents? text2$"
        #main.tbx3 "!Contents? text3$"
        #main.tbx4 "!Contents? text4$"
        Notice "The textboxes read";Chr$(13);Chr$(13) + _
            text1$;Chr$(13);text2$;Chr$(13);text3$;Chr$(13);text4$
    End Sub

    Sub EndDemoMenu
        Call EndDemo "#main"
    End Sub

    Sub EndDemo handle$
        Close #main
        End
    End Sub
```

An Improved Stylebits Textbox

Since the release of Liberty BASIC 4.0 and stylebits, multilines and wordwrapping have been

accomplished in textboxes using the sequence _WS_VSCROLL OR _ES_MULTILINE, _ES_AUTOHSCROLL, 0, 0. Stylebits - Textboxes discusses such a textbox. While the text does, indeed, wrap, the user must resort to CTRL - Enter to force a line feed. Using a hidden button with the stylebit _BS_DEFPUSHBUTTON allows the program to retrieve the contents of the textbox, append Chr$(13);Chr$(10) to the text, then rewrite that text to the textbox. Now the textbox behaves just like any other word processing tool. As is common with so many of the Stylebits tricks, an API call is required to add the final polish. With native Liberty BASIC code, the cursor is placed in front of any text that's been printed in a textbox. The API call "SendMessageA" can be used to position the cursor at the end of the text instead. This cursor position makes keyboarding a more natural and fluent activity.

```
    Nomainwin

    WindowWidth = 400
    WindowHeight = 300

    UpperLeftX = Int((DisplayWidth - WindowWidth)/2)
    UpperLeftY = Int((DisplayHeight - WindowHeight)/2)

' Add a Textbox with Word Wrapping Capability
    Stylebits #main.tbx, _WS_VSCR
OLL OR _ES_MULTILINE, _ES_AUTOHSCROLL, 0, 0
    Textbox #main.tbx, 20, 20, 150, 200

' Add a Hidden Default Button - Use Stylebits Addbits _BS_DEFPUSHBUTTO
N
    Stylebits #main.btn, _BS_DEFPUSHBUTTON, 0, 0, 0
    Button #main.btn, "", DefaultButton, UL, -10, -10

    Open "Main Window" for Window as #main
    #main "Trapclose EndDemo"
    #main "Font Verdana 10 Bold"
    #main.tbx "!Setfocus"

    Wait

    Sub EndDemo handle$
        Close #main
        End
    End Sub

    Sub DefaultButton handle$
        hTextbox = hWnd(#main.tbx)
        #main.tbx "!Contents? text$"
        text$ = text$;Chr$(13);Chr$(10)
        #main.tbx text$
```

```
        pos = Len(text$)
        CallDLL #user32, "SendMessageA", _
            hTextbox as Ulong, _
            _EM_SETSEL as Long, _
            pos as Long, _
            pos as Long, _
            result as Long
    End Sub
```

If you desire or need a truly editable textbox that mimics a text editor, use Alyce Watson's free API Text Editor available at Alyce's Restaurant.

Resizing a Dialog Window

The borders of a native dialog Window (default style = _WS_DLGFRAME) cannot be resized by the user. Neither can the dialog window be maximized or minimized.

```
    Nomainwin
    WindowWidth = 300
    WindowHeight = 350
    Open "Testing" for Dialog as #dlg
    #dlg "Trapclose CloseDlg"

    Wait

    Sub CloseDlg handle$
        Close #dlg
        End
    End Sub
```

In a regular window, you can use _WS_MAXIMIZEBOX or _WS_MINIMIZEBOX in RemoveBits to remove the Maximize and Minimize Controls. In a dialog window, place one or both of these stylebits in the AddBits. To allow border resizing, add the stylebit _WS_THICKFRAME

```
'Demo illustrating a Dialog Window with Resizable Edges

    Nomainwin

    WindowWidth = 800
    WindowHeight = 600

    UpperLeftX = Int((DisplayWidth - WindowWidth)/2)
```

```
    UpperLeftY = Int((DisplayHeight - WindowHeight)/2)

    Button #main.b1,
"  Normal Dialog Window  ", DialogWin1, UL, 320, 250
    Button #main.b2,
" Stylebits Dialog Window ", DialogWin2, UL, 320, 300

    Open "Main Window" for Window as #main
    #main "Trapclose EndDemo"
    Wait

    Sub EndDemo handle$
        Close #main
        End
        End Sub

    Sub DialogWin1 handle$
        WindowWidth = 400
        WindowHeight = 300
        Stylebits #dlg, 0, 0, 0, 0
        Button #dlg.default, " Close ", CloseDlg, UL, 180, 200
        Open "Dialog Window" for Dialog_Modal as #dlg
        #dlg "Trapclose CloseDlg"
        End Sub

    Sub DialogWin2 handle$
        WindowWidth = 400
        WindowHeight = 300
        Stylebits #dlg, _WS_THI
CKFRAME or _DS_CENTER or _WS_MAXIMIZEBOX or _WS_MINIMIZEBOX, 0, 0, 0
        Button #dlg.default, " Close ", CloseDlg, UL, 180, 200
        Open "Dialog Window" for Dialog_Modal as #dlg
        #dlg "Trapclose CloseDlg"
        End Sub

    Sub CloseDlg handle$
        Close #dlg
        End Sub
```

Don't expect to be able to issue a resizehandler control to this modified dialog window, though. The resizehandler command wasn't designed to work with a dialog window, not even a 'stylebitized' dialog window.

Positioning a Dialog Window

A dialog window can easily be centered using the Stylebit _DS_CENTER. See Tip Corner - Center a Dialog with Stylebits by Alyce Watson in the Liberty BASIC Newsletter Issue #128. This centering is in relationship to the screen (monitor) display. By default, dialog windows position themselves according to the client (calling window) area. In the following demo, move the main window (#main) around the screen prior to opening the dialog window. The dialog window will always position itself according to the position of the main window.

```
    Nomainwin
    WindowWidth = 600
    WindowHeight = 400

    UpperLeftX = Int((DisplayWidth - WindowWidth)/2)
    UpperLeftY = Int((DisplayHeight - WindowHeight)/2)

    Button #main.dlg, "Native Dialog", dlg, UL, 220, 200, 140, 30

    Open "Stylebits and Dialogs" for Window as #main
    Wait

    Sub CloseMain handle$
        Close #main
        End
    End Sub

    Sub CloseDlg handle$
        Close #handle$
    End Sub

    Sub dlg handle$
        WindowWidth = 300
        WindowHeight = 250

        UpperLeftX = 20
'UpperLeftX 20 pixels to right of #main upper left corner
        UpperLeftY = 20
'UpperLeftY 20 pixels down from #main upper left corner

        text$ =
"UpperLeftX and UpperLeftY are, by default, based upon the client "
        text$ = text$;
"coordinates of the calling window and not the screen coordinates."
        text$ = text$;Chr$(13)
;Chr$(13);"UpperLeftX = 20";Chr$(13);"UpperLeftY = 20"
        Statictext #dlg1, text$, 50, 50, 200, 200
        Open "Dialog - No Stylebits" for Dialog_Modal as #dlg
```

```
        #dlg "Trapclose CloseDlg"
    End Sub
```

If your program requires a dialog window to be placed according to screen coordinates, use the stylebit _DS_ABSALIGN.

```
    Nomainwin
    WindowWidth = 600
    WindowHeight = 400

    UpperLeftX = Int((DisplayWidth - WindowWidth)/2)
    UpperLeftY = Int((DisplayHeight - WindowHeight)/2)

    Button #main.dlg, "_DS_ABSALIGN", dlg, UL, 220, 200, 140, 30

    Open "Stylebits and Dialogs" for Window as #main
    Wait

    Sub CloseMain handle$
        Close #main
        End
    End Sub

    Sub CloseDlg handle$
        Close #handle$
    End Sub

    Sub dlg handle$
        WindowWidth = 300
        WindowHeight = 250

        UpperLeftX = 20
'UpperLeftX 20 pixels to right of upper left screen corner
        UpperLeftY = 20
'UpperLeftY 20 pixels down from upper left screen corner

        text$ =
"_DS_ABSALIGN causes the UpperLeftX and UpperLeftY to be "
        text$ = text$;
"determined not as client coordinates but as screen coordinates."
        text$ = text$;Chr$(13)
;Chr$(13);"UpperLeftX = 20";Chr$(13);"UpperLeftY = 20"
        Statictext #dlg, text$, 50, 50, 200, 200
        Stylebits #dlg, _DS_ABSALIGN, 0, 0, 0
        Open "Dialog - _DS_ABSALIGN" for Dialog_Modal as #dlg
```

```
        #dlg "Trapclose CloseDlg"
    End Sub
```

One other coordinate positioning trick can be accomplished with stylebits. Assigning the stylebit _DS_CENTERMOUSE will cause the dialog window to position itself so that the cursor lies in the center of the dialog window.

```
    Nomainwin
    WindowWidth = 600
    WindowHeight = 400

    UpperLeftX = Int((DisplayWidth - WindowWidth)/2)
    UpperLeftY = Int((DisplayHeight - WindowHeight)/2)

'All 4 Buttons Call Same Sub
    Button #main.dlg1, "Mouse Centered Dialog"
, dlg, UL, 20, 80, 140, 30
    Button #main.dlg2, "Mouse Centered Dialog"
, dlg, UL, 160, 120, 140, 30
    Button #main.dlg3, "Mouse Centered Dialog"
, dlg, UL, 300, 220, 140, 30
    Button #main.dlg4, "Mouse Centered Dialog"
, dlg, UL, 400, 300, 140, 30

    Open "Stylebits and Dialogs" for Window as #main

    #main "Trapclose CloseMain"

    Wait

    Sub CloseMain handle$
        Close #main
        End
    End Sub

    Sub CloseDlg handle$
        Close #handle$
    End Sub

    Sub dlg handle$
        WindowWidth = 300
        WindowHeight = 250

        UpperLeftX = 20 'UpperLeftX will be ignored
        UpperLeftY = 20 'UpperLeftY will be ignored
```

```
        Stylebits #dlg, _DS_CENTERMOUSE, 0, 0, 0
        Open "Dialog - _DS_CENTERMOUSE" for Dialog_Modal as #dlg
        #dlg "Trapclose CloseDlg"
    End Sub
```

A dialog window can also be positioned in the forefront of all other windows, even when that dialog window isn't the window with focus. The _DS_SYSMODAL stylebit acts the same as the _WS_EX_TOPMOST stylebit for regular windows.

```
    Nomainwin
    WindowWidth = 600
    WindowHeight = 400

    UpperLeftX = Int((DisplayWidth - WindowWidth)/2)
    UpperLeftY = Int((DisplayHeight - WindowHeight)/2)

'All 4 Buttons Call Same Sub
    Button #main.dlg, "_DS_SYSMODAL", dlg, UL, 220, 200, 140, 30

    Open "Stylebits and Dialogs" for Window as #main

    #main "Trapclose CloseMain"

    Wait

    Sub CloseMain handle$
        Close #main
        End
    End Sub

    Sub CloseDlg handle$
        Close #handle$
    End Sub

    Sub dlg handle$
        WindowWidth = 300
        WindowHeight = 250

        UpperLeftX = 20
        UpperLeftY = 20

        Stylebits #dlg, _DS_SYSMODAL, 0, 0, 0
        Open "Dialog - _DS_SYSMODAL" for Dialog as #dlg
        #dlg "Trapclose CloseDlg"
    End Sub
```

A List of Stylebits

You can get a list of all dwStyles and dwExStyles available with the Stylebits command at the MSDN Library - Dialog Styles.
Be sure to precede these constants with an underscore (if constant is DS_CENTERMOUSE, then LB Stylebits is _DS_CENTERMOUSE) when using Windows constants in your Liberty BASIC programs.