The PUSHLIKE Stylebits

Stylebits allows checkboxes and radiobuttons to appear to be pushed rather than selected. Stefan Pendl, always generous with explanations and code, offered button information in response to a query posed at the Liberty BASIC yahoo! message group. (See the original message #27196.)

For a better understanding, the following controls are all defined as buttons by Windows:

1. button
2. groupbox
3. bmpbutton
4. checkbox
5. radiobutton

You can use the style _BS_PUSHLIKE to change the radiobutton and checkbox into a pushbutton, that will keep its state.

These buttons are called toggle buttons, they are raised when inactive and sunken when active. To find out more about toggle buttons, read the information at the MSDN Library.

Stefan's Demo

Stefan was kind enough to grant permission for his yahoo! code to be republished here. Using the _BS_PUSHLIKE stylebits, a checkbox and two grouped radiobuttons are altered from their usual forms to button forms. The change in style does not alter the set and unset properties.

```
    nomainwin

    groupbox #main.gb, "Toggle Radiobuttons", 10, 10, 120, 100
    stylebits #main.bt, _BS_PUSHLIKE, 0, 0, 0
    radiobutton #main.bt, "Toggle"
, [pushed], [pushed], 20, 30, 100, 30
    stylebits #main.bt1, _BS_PUSHLIKE, 0, 0, 0
    radiobutton #main.bt1, "Toggle1"
, [pushed], [pushed], 20, 70, 100, 30

    stylebits #main.bt2, _BS_PUSHLIKE, 0, 0, 0
    checkbox #main.bt2, "Toggle Checkbox"
, [pushed], [pushed], 20, 120, 100, 30

    open "Toggle demo" for window as #main
    #main "trapclose [quit]"
```

```
    wait

[pushed]
    wait

[quit]
    close #main
    end
```

Stylebits, Graphics and the API call, "SendMessageA"

Creating graphic buttons using _BS_BITMAP and CallDLL #user32, "SendMessageA" has been discussed by Alyce Watson in API Corner - Easy BMP Buttons (LB Newsletter #123) and also in Stylebits - Buttons. In these next two demos, the graphics are drawn and captured within the program. You could easily substitute your own code for loading the bitmaps from file.

```
'Demonstration of using Stylebits to turn checkboxes into
'push buttons
'Based on code provided by Stefan Pendl
'Yahoo! Message ##27196

    Nomainwin

    WindowWidth = 207
    WindowHeight = 297
    UpperLeftX = Int((DisplayWidth - WindowWidth)/2)
    UpperLeftY = Int((DisplayHeight - WindowHeight)/2)

'Groupbox defines Checkbox Area
    Groupbox #main.gp, "Color", 10, 10, 120, 100

'Draw checkboxes that will look like push buttons
'Stylebits _BS_PUSHLIKE causes checkbox to resemble push button
'Stylebits _BS_BITMAP allows a bitmap to be drawn on button
'Be sure to set width and height to bitmap dimensions
    Stylebits #main.bt1, _BS_PUS
HLIKE or _BS_BITMAP, 0, _WS_EX_DLGMODALFRAME, 0
    Checkbox #main.bt1, "", checked, unchecked, 20, 30, 40, 30
    Stylebits #main.bt2, _BS_PUS
HLIKE or _BS_BITMAP, 0, _WS_EX_DLGMODALFRAME, 0
    Checkbox #main.bt2, "", checked, unchecked, 20, 70, 40, 30
    Stylebits #main.bt3, _BS_PUS
HLIKE or _BS_BITMAP, 0, _WS_EX_DLGMODALFRAME, 0
```

```
    Checkbox #main.bt3, "", checked, unchecked, 80, 30, 40, 30
    Stylebits #main.bt4, _BS_PUS
HLIKE or _BS_BITMAP, 0, _WS_EX_DLGMODALFRAME, 0
    Checkbox #main.bt4, "", checked, unchecked, 80, 70, 40, 30

'For this demo, a graphicbox is needed to draw and capture bitmaps
    Graphicbox #main.gb, 0, 150, 200, 120
    Open "Stylebits and Checkboxes" for Window as #main
    #main "Trapclose endDemo"

'Obtain handle of the main window
    hMain = hWnd(#main)
'Obtain handles of the 4 checkboxes
    hButton1 = hWnd(#main.bt1)
    hButton2 = hWnd(#main.bt2)
    hButton3 = hWnd(#main.bt3)
    hButton4 = hWnd(#main.bt4)

'================================================
'To load bitmaps from disk, use
    'Loadbmp "buttonPic1", "buttonPic1.bmp"
    'Loadbmp "buttonPic2", "buttonPic2.bmp"
    'etc.
'================================================
'For this demo, the bitmaps will be drawn and captured
    #main.gb "Down; Fill Black; Size 2"
    #main.gb "Color Lightgray; Backcolor DarkRed"
    #main.gb "Place 2 2; Boxfilled 38 28"
    #main.gb "Getbmp buttonPic1 0 0 40 30"
    #main.gb "Backcolor Darkblue"
    #main.gb "Place 2 2; Boxfilled 38 28"
    #main.gb "Getbmp buttonPic2 0 0 40 30"
    #main.gb "Backcolor Yellow"
    #main.gb "Place 2 2; Boxfilled 38 28"
    #main.gb "Getbmp buttonPic3 0 0 40 30"
    #main.gb "Backcolor Darkgreen"
    #main.gb "Place 2 2; Boxfilled 38 28"
    #main.gb "Getbmp buttonPic4 0 0 40 30"
'================================================

'Get the handle of the bitmaps
    hPic1 = hBmp("buttonPic1")
    hPic2 = hBmp("buttonPic2")
    hPic3 = hBmp("buttonPic3")
    hPic4 = hBmp("buttonPic4")
```

```
'Paint the bitmaps on the buttons
    For i = 1 to 4
        Select Case i
            Case 1
                hBtn = hButton1
                hPic = hPic1
            Case 2
                hBtn = hButton2
                hPic = hPic2
            Case 3
                hBtn = hButton3
                hPic = hPic3
            Case 4
                hBtn = hButton4
                hPic = hPic4
        End Select
        CallDll #user32, "SendMessageA", _
            hBtn as uLong, _
            _BM_SETIMAGE as Long, _
            _IMAGE_BITMAP as Long, _
            hPic as uLong, _
            result as Long
    Next i

'Erase button drawings in graphicbox
    #main.gb "Cls; Fill Black"

'Wait for user input
    Wait

'Find which checkbox checked
'Fill quadrant with designated color
    Sub checked handle$
        clr = Val(Right$(handle$, 1))
        Select Case clr
            Case 1
                #main.gb "Color Darkred; Backcolor Darkred"
                x = 1
                y = 1
            Case 2
                #main.gb "Color Darkblue; Backcolor Darkblue"
                x = 1
                y = 61
            Case 3
                #main.gb "Color Yellow; Backcolor Yellow"
                x = 101
```

```
                y = 1
            Case 4
                #main.gb "Color Darkgreen; Backcolor Darkgreen"
                x = 101
                y = 61
        End Select
        #main.gb "Place ";x;" ";y
        #main.gb "Boxfilled ";x + 99;" ";y + 59
        End Sub

'Find which checkbox unchecked
'Fill appropriate quadrant with black
    Sub unchecked handle$
        clr = Val(Right$(handle$, 1))
        #main.gb "Color Black; Backcolor Black"
        Select Case clr
            Case 1
                x = 1
                y = 1
            Case 2
                x = 1
                y = 61
            Case 3
                x = 101
                y = 1
            Case 4
                x = 101
                y = 61
        End Select
        #main.gb "Place ";x;" ";y
        #main.gb "Boxfilled ";x + 99;" ";y + 59
        End Sub

'End the program
    Sub endDemo handle$
'Unload bitmaps to free memory
        For i = 1 to 4
            Unloadbmp "buttonPic";i
        Next i
        Close #main
        End
        End Sub
```

Button Event Handlers: Branch Label or Sub?

As is the case with most event handlers, the button event handler can be either a branch label or a sub. The advantage of using a sub with grouped checkboxes or radiobuttons is that the control handle is automatically passed into the sub. With careful parsing of the control handle extension and the use of Select Case, code becomes much more efficient than is possible with branch labels. The following demo contains just four radiobuttons within a group, but could easily serve a group of 100 radiobuttons or more.

```
'Demonstration of using Stylebits to turn radiobuttons into
'push buttons
'Based on code provided by Stefan Pendl
'Yahoo! Message ##27196

    Nomainwin

    WindowWidth = 207
    WindowHeight = 297
    UpperLeftX = Int((DisplayWidth - WindowWidth)/2)
    UpperLeftY = Int((DisplayHeight - WindowHeight)/2)

'Groupbox defines Radiobuttons Area
    Groupbox #main.gp, "Color", 10, 10, 120, 100

'Draw radiobuttons that will look like push buttons
'Stylebits _BS_PUSHLIKE causes radiobutton to resemble push button
'Stylebits _BS_BITMAP allows a bitmap to be drawn on button
'Be sure to set width and height to bitmap dimensions
    Stylebits #main.bt1, _BS_PUS
HLIKE or _BS_BITMAP, 0, _WS_EX_DLGMODALFRAME, 0
    Radiobutton #main.bt1, "", selected, unselected, 20, 30, 40, 30
    Stylebits #main.bt2, _BS_PUS
HLIKE or _BS_BITMAP, 0, _WS_EX_DLGMODALFRAME, 0
    Radiobutton #main.bt2, "", selected, unselected, 20, 70, 40, 30
    Stylebits #main.bt3, _BS_PUS
HLIKE or _BS_BITMAP, 0, _WS_EX_DLGMODALFRAME, 0
    Radiobutton #main.bt3, "", selected, unselected, 80, 30, 40, 30
    Stylebits #main.bt4, _BS_PUS
HLIKE or _BS_BITMAP, 0, _WS_EX_DLGMODALFRAME, 0
    Radiobutton #main.bt4, "", selected, unselected, 80, 70, 40, 30

'For this demo, a graphicbox is needed to draw and capture bitmaps
    Graphicbox #main.gb, 0, 150, 200, 120
    Open "Stylebits and Radiobuttons" for Window as #main
    #main "Trapclose endDemo"

'Obtain handle of the main window
    hMain = hWnd(#main)
```

```
'Obtain handles of the 4 radiobuttons
    hButton1 = hWnd(#main.bt1)
    hButton2 = hWnd(#main.bt2)
    hButton3 = hWnd(#main.bt3)
    hButton4 = hWnd(#main.bt4)

'================================================
'To load bitmaps from disk, use
    'Loadbmp "buttonPic1", "buttonPic1.bmp"
    'Loadbmp "buttonPic2", "buttonPic2.bmp"
    'etc.
'================================================
'For this demo, the bitmaps will be drawn and captured
    #main.gb "Down; Fill Black; Size 2"
    #main.gb "Color Lightgray; Backcolor DarkRed"
    #main.gb "Place 2 2; Boxfilled 38 28"
    #main.gb "Getbmp buttonPic1 0 0 40 30"
    #main.gb "Backcolor Darkblue"
    #main.gb "Place 2 2; Boxfilled 38 28"
    #main.gb "Getbmp buttonPic2 0 0 40 30"
    #main.gb "Backcolor Yellow"
    #main.gb "Place 2 2; Boxfilled 38 28"
    #main.gb "Getbmp buttonPic3 0 0 40 30"
    #main.gb "Backcolor Darkgreen"
    #main.gb "Place 2 2; Boxfilled 38 28"
    #main.gb "Getbmp buttonPic4 0 0 40 30"
    #main.gb "Backcolor Black; Fill Black"
'================================================

'Get the handle of the bitmaps
    hPic1 = hBmp("buttonPic1")
    hPic2 = hBmp("buttonPic2")
    hPic3 = hBmp("buttonPic3")
    hPic4 = hBmp("buttonPic4")

'Paint the bitmaps on the radiobuttons
    For i = 1 to 4
        Select Case i
            Case 1
                hBtn = hButton1
                hPic = hPic1
            Case 2
                hBtn = hButton2
                hPic = hPic2
            Case 3
                hBtn = hButton3
```

```
                hPic = hPic3
            Case 4
                hBtn = hButton4
                hPic = hPic4
        End Select
        CallDll #user32, "SendMessageA", _
            hBtn as uLong, _
            _BM_SETIMAGE as Long, _
            _IMAGE_BITMAP as Long, _
            hPic as uLong, _
            result as Long
    Next i

'Erase button drawings in graphicbox
    #main.gb "Cls; Fill Black"

'Wait for user input
    Wait

'Find which radiobutton checked
'Fill quadrant with designated color
    Sub selected handle$
        clr = Val(Right$(handle$, 1))
        #main.gb "Cls; Fill Black"
        Select Case clr
            Case 1
                #main.gb "Color Darkred; Backcolor Darkred"
                x = 1
                y = 1
            Case 2
                #main.gb "Color Darkblue; Backcolor Darkblue"
                x = 1
                y = 61
            Case 3
                #main.gb "Color Yellow; Backcolor Yellow"
                x = 101
                y = 1
            Case 4
                #main.gb "Color Darkgreen; Backcolor Darkgreen"
                x = 101
                y = 61
        End Select
        #main.gb "Place ";x;" ";y
        #main.gb "Boxfilled ";x + 99;" ";y + 59
        End Sub
```

```
'As is the case with any radiobutton, an
'Unset event handler needs to be defined
'but is never actually executed
    Sub unselected handle$
        clr = Val(Right$(handle$, 1))
        #main.gb "Color Black; Backcolor Black"
        #main.gb "Cls; Fill Black"
        End Sub

'End the program
    Sub endDemo handle$
'Unload bitmaps to free memory
        For i = 1 to 4
            Unloadbmp "buttonPic";i
        Next i
        Close #main
        End
        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 - Button Styles.
Be sure to precede these constants with an underscore (if constant is BS_PUSHLIKE, then LB Stylebits is _BS_PUSHLIKE) when using Windows constants in your Liberty BASIC programs.