MDI Client to Scroll Controls

For an eBook or printed book on using the API with Liberty BASIC, see:
APIs for Liberty BASIC

Why Scroll?

Sometimes we simply don't have enough room on a window for all of the controls needed by a program. We can get around this in several ways. One way is to move controls on and off the window as needed with the LOCATE statement. Another way is to make the controls do double duty. When one set of conditions is in effect, button one might be the "Okay" button, but when another set of conditions is in effect, it might be the "Save" button. This can get a little complicated! We can also break the program into logical parts and use separate windows to manage different tasks.

Here is a new idea. We can scroll the controls on the window. If we allow scrolling, then more controls, or larger controls may fit on a single window.

MDI is the Key

We could go about the scrolling in several ways, but using an MDI client is probably the easiest way of all. An MDI application has an MDI client that acts as parent to any child windows it contains. When one of the child windows is larger than the workspace, the MDI window automatically adds scrollbars to allow access to the entire child window.

Creating an MDI Client

As with other API-created controls, we create the MDI client with CreateWindowExA. It will itself be a child of our main program window, and it won't really be visible until it displays scrollbars. We need the handle of our program window, and the instance handle, which we retrieve with GetWindowLongA. We also fill arguments for the location and dimensions. See the sample program below, where all arguments for CreateWindowExA are documented.

Creating a Child Window

We'll need a child window to hold all of our controls. We don't want the user to move it around, so we'll create it with no titlebar using the window_popup style. We can make this window as large as we need to hold the controls, and of course we need to include commands to create these controls before the command to open the popup window. We'll locate the popup window so that it fills the client area of our main program window. Once this popup window is created, we'll make it a child of the MDI client with the SetParent function.

Making it Work

We need to check for the resize event of the window and resize the MDI client so that it fills the new workspace whenever the window changes size. If the user has scrolled the MDI window and then resizes the program window, the child popup will not be in the correct locatoin, so we'll also need to relocate our popup child window. We need to force a resize event when the window first opens so that the MDI scrollbars will display. We do all of these sizing and locating chores with MoveWindow.

The demo below only has a single button on the far right side of the popup child window. Your own program would probably have many controls on this window.

Scrolling Demo

```
'MDI window to allow
'scrolling of large window
'area
'based on work
'By Mitchell Kotler

nomainwin

    WindowWidth=350 : WindowHeight=350
    childWide=1400 : childHigh=1000 'child window dims

menu #main, "File", "E&xit",[quit]
open "Scrolling Big Window" for window as #main
print #main, "trapclose [quit]"
print #main, "resizehandler [resize]"
```

```
    hMain=hwnd(#main) 'main window handle

    calldll #user32, "GetWindowLongA",_
        hMain as uLong,_          'handle of window
        _GWl_HINSTANCE as long,_ 'flag for instance handle
        hInstance as uLONG       'returns instance handle of window

    dwStyle=_WS_CLIPCHILDREN OR _WS_CHILD OR _WS_VISIBLE OR _
        _WS_BORDER or _WS_VSCROLL OR _WS_HSCROLL

    'create an MDI Client Control
    calldll #user32, "CreateWindowExA",_
        0 as long,_              'extended class style
        "MDICLIENT" as ptr,_     'class name
        "" as ptr,_              'title or string
        dwStyle as long,_        'window style
        2 as long,_              'x org
        2 as long,_              'y org
        339 as long,_            'width
        302 as long,_            'height
        hMain as ulong,_         'parent window
        0 as ulong,_             'handle to menu = 0 for class menu
        hInstance as ulong,_     'instance handle of parent window
        "" as ptr,_              'always NULL
        hMDI as ulong            'returns handle of MDI Client

WindowWidth=childWide:WindowHeight=childHigh
statictext #child, "Scroll Me A LOT!",10,10,200,24
button #child.b, "Click Me!",[doClick],UL,1200,100,100,24
open "" for window_popup as #child
print #child, "trapclose [quit]"

    hChild=hwnd(#child) 'handle of popup window

    calldll #user32, "SetParent",_
        hChild as ulong,_    'make popup the child
        hMDI as ulong,_      'make MDI the parent
        result as long

    'use MoveWindow to force window resize
    'so scrollbars will show
    CallDLL #user32, "MoveWindow",hMain As uLong,_
         11 As Long, 11 As Long,_
         333 As Long, 333 As Long,_
         1 As long, r As long
```

```
wait

[resize]
    newWide=WindowWidth-4
    newHigh=WindowHeight-4
    ret=MoveWindow(hMDI,2,2,newWide,newHigh)
    ret=MoveWindow(hChild,0,0,childWide,childHigh)
    wait

[doClick]
    notice "Thanks for clicking me!"
    wait

[quit] close #child : close #main : end

Function MoveWindow(hWnd,x,y,wide,high)
    CallDLL #user32, "MoveWindow",_
        hWnd As uLong,_          'handle
        x As Long, y As Long,_   'x,y pos
        wide As Long,_           'width
        high As Long,_           'height
        1 As long,_              'repaint flag
        MoveWindow As long
    end function
```

Why Scroll? | MDI is the Key | Creating an MDI Client | Creating a Child Window | Making it Work | Scrolling Demo