Stringheight for Liberty BASIC

`Stringwidth?` is the Liberty BASIC command for obtaining the width in pixels of any given character or chain of characters. `Stringwidth?` calculates this measurement based upon the current font in use.

```
    Open "The Stringwidth? Command" for Graphics as #g
    #g "Trapclose XbyTrap"
    char$ = "X"
    #g "Down; Place 20 50"
    #g "Font Times_New_Roman 14"
    #g "Stringwidth? char$ PixelWidth"
    #g "\The width of ";char$;" is ";PixelWidth;" pixels."
    #g "Font Courier_New 36"
    #g "Stringwidth? char$ PixelWidth"
    #g "\\The width of ";char$;" is ";PixelWidth;" pixels."
    #g "Flush"
Wait

Sub XbyTrap handle$
    Close #g
End
End Sub
```

Calculating Stringheight with Posxy

Unfortunately, there is no native Stringheight? function. A simple way to determine the height of a font is to use the `posxy xVar yVar` command. The `posxy` function returns the position of the graphic pen. Define the font, position the pen, draw text, get the new y position. The difference between the two y positions is the height of the font.

```
    Open "Calculating Stringheight with Posxy" for Graphics as #g
    #g "Trapclose XbyTrap"
    Char$ = "X"
    #g "Down; Place 20 50"

' Select a font
    #g "Font Times_New_Roman 14"

' Get the pen position before drawing the text
    #g "Posxy x1Var y1Var"

' Draw the text
    #g "\";Char$
```

```
' Get the pen position after drawing the text
    #g "Posxy x2Var y2Var"

' Subtract to determine Stringheight
    Stringheight = y2Var - y1Var

' Show the value of Stringheight
    #g "\The height of ";Char$;" is ";Stringheight;" pixels."

' Select another font
    #g "Font Courier_New 36"

' Create a linefeed
    #g "\"

' Get the pen position before drawing the text
    #g "Posxy x1Var y1Var"

' Draw the second text
    #g "\";Char$

' Get the pen position after drawing the text
    #g "Posxy x2Var y2Var"

' Subtract to determine Stringheight
    Stringheight = y2Var - y1Var

    #g "\The height of ";Char$;" is ";Stringheight;" pixels."

    #g "Flush"
Wait

Sub XbyTrap handle$
    Close #g
End
End Sub
```

This Stringheight method does not require the text to be drawn, neither does the text have to be visible. Here is the same method using a 1 x 1 graphicbox that is set offscreen, thus invisible to the user.

```
    Graphicbox #main.g, -10, -10, 1, 1
    Statictext #main.t, "Font Dimensions", 20, 50, 250, 100
    Open "Calculating Stringheight with Posxy" for Window as #main
    #main "Trapclose XbyTrap"
    #main "Font Verdana 12 Bold"
```

```
    Char$ = "X"
    #main.g "Down; Place 20 50"

' Select a font
    #main.g "Font Times_New_Roman 14"

' Calculate Stringwidth using the native Stringwidth? function
    #main.g "Stringwidth? Char$ Stringwidth"

' Get the pen position before drawing the text
    #main.g "Posxy x1Var y1Var"

' Draw the text
    #main.g, "\";Char$

' Get the pen position after drawing the text
    #main.g "Posxy x2Var y2Var"

' Subtract to determine Stringheight
    Stringheight = y2Var - y1Var

' Show the value of Stringwidth and Stringheight
    FontDimensions$ = "The dimensions of " ;Char$;" are " + _
        Str$(Stringwidth);" pixels wide and ";Str$(Stringheight) + _
        " pixels high."
    #main.t FontDimensions$

Wait

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

Disadvantages of Using Posxy to Calculate Stringheight

`Posxy` will not return the exact position of the drawn text. It only returns the number of vertical (y) pixels advanced to for beginning the next line. The font space includes not just the font, but padding above, below, to the left, and to the right, of the character itself. The y1Var is the lower position of the drawn font, above the lower padding of the font space.

```
    Open "Calculating Stringheight with Posxy" for Graphics as #g
    #g "Trapclose XbyTrap"
    Char$ = "X"
    #g "Down; Place 20 50"
```

```
    #g "Color Black; Backcolor Cyan"
    #g "Font Times_New_Roman 14"
    #g "Posxy x1Var y1Var"
    #g "\";Char$
    #g "Posxy x2Var y2Var"
    Stringheight = y2Var - y1Var

    #g "Color Red"
' Draw a  red line at y1Var
    #g "Line 0 ";y1Var;" 100 ";y1Var

' Draw a red line at y2Var
    #g "Line 0 ";y2Var;" 100 ";y2Var
    #g "Flush"
Wait

Sub XbyTrap handle$
    Close #g
End
End Sub
```

Stringheight Wrapped in a Function

Still, if all that is needed is the vertical height of the font space, a simple custom function using an offscreen graphicbox is quick and easy.

```
    Nomainwin
    WindowWidth = 800
    WindowHeight = 600
    UpperLeftX = Int((DisplayWidth - WindowWidth) /2)
    UpperLeftY = Int((DisplayHeight - WindowHeight) /2)
    Graphicbox #main.h, -10, -10, 1, 1 ' Hidden graphicbox
    Graphicbox #main.g, 0, 0, 800, 600 ' Visible graphicbox
    Open "Calculating Stringheight with Posxy" for Window as #main
    #main "Trapclose XbyTrap"
    CurrentFont$ = "Verdana 32 Bold"
    #main.g "Font ";CurrentFont$
    Stringheight = Stringheight(CurrentFont$)
    #main.g "Down; Place 20 50"
    For i = 1 to 10
        #main.g "\Line ";i
    Next i
    #main.g "Color Red"
    yPos = 55
    For i = 1 to 10
```

```
        For x = 20 to 250 Step 5
            #main.g "Place ";x;" ";yPos
            #main.g "Circle 4"
        Next x
        yPos = yPos + Stringheight
    Next i
    #main.g, "Flush"

Wait

Sub XbyTrap handle$
    Close #main
End
End Sub

Function Stringheight(CurrentFont$)
    #main.h "Cls"
    #main.h "Font ";CurrentFont$
    #main.h "Place 20 500"
    #main.h "Posxy x1Var y1Var"
    #main.h "\X"
    #main.h "Posxy x2Var y2Var"
    Stringheight = y2Var - y1Var
End Function
```

The Font Dimensions

It is possible to calculate all the font dimensions of the font space as well as the dimensions of the character itself by using a simple gdi32 call. The `"GetPixel"` API call returns the pixel color at a given x,y location. Set the `backcolor` to black, then search for 0, the numerical value of black, to find the boundaries of the drawn character.

```
    WindowWidth = 800
    WindowHeight = 600
    Graphicbox #main.g, 0, 0, 800, 600
    Open "StringHeight for Liberty BASIC" for Window as #main
    #main "Trapclose XbyTrap"
    #main.g "Down; Color Blue; Backcolor Black"

' Assign a font
    #main.g "Font Times_New_Roman 24"

' Draw a character
    #main.g "Place 100 100"
    #main.g "\X"
```

```
    #main.g "Flush"
Wait

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

The GetPixel Call

Include a custom function to return the pixel value in your code. Since the `"GetPixel"` call requires the handle of the device context, the `"GetDC"` custom function should also be included in your code. Before ending your program, release the retrieved device contexts with the `"ReleaseDC"` sub.

```
' Get the device context
    hDC = hDC(hWnd(#main.g))

Function PixelLong(hDC, xVar, yVar)
    Open "gdi32"for DLL as #gdi
    CallDLL #gdi, "GetPixel",_
        hDC as Ulong,_
        xVar as Long,_
        yVar as Long,_
        PixelLong as Long
   Close #gdi
End Function

Function hDC(handle)
    CallDLL #user32, "GetDC",_
        handle as Ulong,_
        hDC as Ulong
End Function

Sub ReleaseDC hW, hDC
    CallDLL#user32,"ReleaseDC", _
        hW as Ulong, _
        hDC as Ulong, _
        result as Long
End Sub
```

Find the Boundaries of the Font Space

The font space contains more than just the font. There is usually a padding above and below the font, as well as to the left and right of the font. To find the font space boundaries, start at a location outside of the font and work toward the font, stopping when a black pixel is found.

- Finding UpperY of the Font Space

Beginning at 0, work down until a black pixel is found. The x value of 101 is used because the font was drawn at 100, 100.

```
For y = 1 to 600
    If PixelLong(hDC, 101, y) = 0 Then
        UpperY = y
        Exit For
    End If
Next y
```

- Finding LowerY of the Font Space

Start at the lowest possible and work up until a black pixel is found.

```
For y = 600 to 1 Step -1
    If PixelLong(hDC, 101, y) = 0 then
        LowerY = y
        Exit For
    End If
Next y
```

- Finding the LeftX of the Font Space

LeftX is the first black pixel encountered beginning at 0 and working to the right (x increases).

```
For x = 0 to 800
    If PixelLong(hDC, x, 100) = 0 Then
        LeftX = x
        Exit For
    End If
Next x
```

- Finding the RightX of the Font Space

RightX is the first black pixel encountered moving from the farthest right of the window to the left (x

decreases).

```
For x = 800 to 0 Step -1
    If PixelLong(hDC, x, 100) = 0 Then
        RightX = x
        Exit For
    End If
Next x
```

- Boxing the Font Space

To show that the variables are accurate, draw a box around the font space beginning with LeftX, UpperY and extending to RightX, LowerY.

```
#main.g "Color Red; Backcolor White"
#main.g "Place ";LeftX;" ";UpperY
#main.g "Box ";RightX + 1;" ";LowerY + 1
```

- Calculating Stringheight

Stringheight is RightX - LeftX.

```
Stringheight = LowerY - UpperY
#main.g "Place 100 "; 100 + Stringheight * 2
#main.g "\Stringheight = ";Stringheight
```

- Release the Device Context

Finally, release the device context from memory.

```
Call ReleaseDC hWnd(#main.g), hDC
```

- Putting It All Together

Here is the code in its entirety.

```
WindowWidth = 800
WindowHeight = 600
Graphicbox #main.g, 0, 0, 800, 600
Open "StringHeight for Liberty BASIC" for Window as #main
#main "Trapclose XbyTrap"
```

```
    #main.g "Down; Color Blue; Backcolor Black"

' Assign a font
    #main.g "Font Times_New_Roman 24"

' Draw a character
    #main.g "Place 100 100"
    #main.g "\X"
    #main.g "Flush"

' Get the device context
    hDC = hDC(hWnd(#main.g))

' Find the UpperY boundary of the font space
    For y = 0 to 600
        If PixelLong(hDC, 101, y) = 0 Then
            UpperY = y
            Exit For
        End If
    Next y

' Find the LowerY boundary of the font space
    For y = 600 to 0 Step -1
        If PixelLong(hDC, 101, y) = 0 then
            LowerY = y
            Exit For
        End If
    Next y

' Find the LeftX boundary of the font space
    For x = 0 to 800
        If PixelLong(hDC, x, 100) = 0 Then
            LeftX = x
            Exit For
        End If
    Next x

' Find the RightX boundary of the font space
    For x = 800 to 0 Step -1
        If PixelLong(hDC, x, 100) = 0 Then
            RightX = x
            Exit For
        End If
    Next x

' Outline the font
```

```
    #main.g "Color Red; Backcolor White"
    #main.g "Place ";LeftX;" ";UpperY
    #main.g "Box ";RightX + 1;" ";LowerY + 1

' Calculate Stringheight
    Stringheight = LowerY - UpperY
    #main.g "Place 100 "; 100 + Stringheight * 2
    #main.g "\Stringheight = ";Stringheight

' Release the device context
    Call ReleaseDC hWnd(#main.g), hDC
Wait

Sub XbyTrap handle$
    Close #main
End
End Sub

Function PixelLong(hDC, xVar, yVar)
    Open "gdi32"for DLL as #gdi
    CallDLL #gdi, "GetPixel",_
        hDC as Ulong,_
        xVar as Long,_
        yVar as Long,_
        PixelLong as Long
   Close #gdi
End Function

Function hDC(handle)
    CallDLL #user32, "GetDC",_
        handle as Ulong,_
        hDC as Ulong
End Function

Sub ReleaseDC hW, hDC
    CallDLL#user32,"ReleaseDC", _
        hW as Ulong, _
        hDC as Ulong, _
        result as Long
End Sub
```

Actual Font Height

If the actual font height, not including the over and under padding, is required, additional searching of black pixels is required. In this instance, the area inside the font space is searched looking for black and non-black pixels. The following demo shows how the width and height of any character of any font can be

determined using the "GetPixel" API call.

```
    Global CurrentFontSpec$, Char$
    Global x1Left, x1Right, y1Upper, y1Lower
    Global x2Left, x2Right, y2Upper, y2Lower
    Global FontPixelWidth1, FontPixelHeight1
    Global FontPixelWidth2, FontPixelHeight2

    Nomainwin
    WindowWidth = 800
    WindowHeight = 600
    UpperLeftX = Int((DisplayWidth - WindowWidth) /2)
    UpperLeftY = Int((DisplayHeight - WindowHeight) /2)
    Graphicbox #main.g, 100, 0, 700, 570
    Stylebits #main.f1, _BS_MULTILINE, 0, 0, 0
    Button #main.f1, "Select Font", FontSelect, UL, 5, 50, 90, 50
    Stylebits #main.f2, _BS_MULTILINE, 0, 0, 0
    Button #main.f2 "Font Character"
, FontCharacter, UL, 5, 120, 90, 50
    Stylebits #main.f3, _BS_MULTILINE, 0, 0, 0
    Button #main.f3 "Font Dimensions"
, FontDimensions, UL, 5, 190, 90, 50
    Open "Calculating Graphic Text Dimensions" for Window as #main
    Print #main, "Trapclose XbyTrap"
    CurrentFontSpec$ = "Times_New_Roman 12 Bold"
    Char$ = "X"
    Print #main, " Font ";CurrentFontSpec$
    Print #main.g, "Font ";CurrentFontSpec$
    Print #main.g, "Down; Fill White"
    Print #main.g, "Color Black; Flush"
Wait

Sub XbyTrap handle$
    Close #main
End
End Sub

Sub FontSelect handle$
    FontDialog CurrentFontSpec$, NewFontSpec$
    If NewFontSpec$ <> "" Then
        CurrentFontSpec$ = NewFontSpec$
    End If
End Sub
```

```
Sub FontCharacter handle$
    p$ = "Character to be measured";Chr$(13)
    p$ = p$;"(Current = ";Char$;")"
    Prompt p$;c$
    If c$ <> "" Then
        Char$ = c$
    End If
End Sub

Sub FontDimensions handle$
    #main.g "Fill White; Cls; Fill White"
    #main.g "Backcolor Black; Color Blue"
    #main.g "Font ";CurrentFontSpec$
    #main.g "Stringwidth? Char$ FontStringWidth"
    h1 = Val(Word$(CurrentFontSpec$, 2))
    h2 = Val(Word$(CurrentFontSpec$, 3))
    yPos = Max(h1, h2) * 2
    #main.g "Place 20 ";yPos
    #main.g "\";Char$
    hDC = hDC(hWnd(#main.g))
    For y = 0 to yPos
        If PixelLong(hDC, 20, y) = 0 Then
            y1Upper = y
            Exit For
        End If
    Next y
    For y = yPos * 2 to 0 Step -1
        If PixelLong(hDC, 20, y) = 0 Then
            y1Lower = y
            Exit For
        End If
    Next y
    FontPixelHeight1 = y1Lower - y1Upper
    For x = 10 to 800
        If PixelLong(hDC, x, y1Upper) = 0 Then
            x1Left = x
            Exit For
        End If
    Next x
    For x = yPos * 2 to 10 Step -1
        If PixelLong(hDC, x, y1Upper) = 0 Then
            x1Right = x
            Exit For
        End If
    Next x
    FontPixelWidth1 = x1Right - x1Left
```

```
    For y = y1Upper to y1Lower
        pixel = 0
        For x = x1Left + 1 to x1Right
            If PixelLong(hDC, x, y) <> 0 Then
                pixel = 1
                Exit For
            End If
        Next x
        If pixel <> 0 Then
            y2Upper = y
            Exit For
        End If
    Next y
    For y = y1Lower to y1Upper Step -1
        pixel = 0
        For x = x1Left to x1Right
            If PixelLong(hDC, x, y) <> 0 Then
                pixel = 1
                Exit For
            End If
        Next x
        If pixel <> 0 Then
            y2Lower = y
            Exit For
        End If
    Next y
    FontPixelHeight2 = y2Lower - y2Upper
    For x = x1Left to x1Right
        pixel = 0
        For y = y2Upper to y2Lower
            If PixelLong(hDC, x, y) <> 0 Then
                pixel = 1
                Exit For
            End If
        Next y
        If pixel <> 0 Then
            x2Left = x
            Exit For
        End If
    Next x
    For x = x1Right to x2Left Step -1
        pixel = 0
        For y = y2Upper to y2Lower
            If PixelLong(hDC, x, y) <> 0 Then
                pixel = 1
                Exit For
```

```
            End If
        Next y
        If pixel <> 0 Then
            x2Right = x
            Exit For
        End If
    Next x
    FontPixelWidth2 = x2Right - x2Left
    #main.g "Backcolor White"
    #main.g "Color Yellow"
    #main.g "Line ";x1Left;" ";yPos;" ";x1Right;" ";yPos
    #main.g "Color Red"
    #main.g "Place ";x1Left - 1;" ";y1Upper - 1
    #main.g "Box ";x1Right + 1;" ";y1Lower + 1
    #main.g "Place ";x1Left;" ";y1Lower + 20
    #main.g "Font Times_New_Roman 12 Bold"
    #main.g "\Total Width = ";FontPixelWidth1;" pixels"
    #main.g "\Total Height = ";FontPixelHeight1;" pixels"
    #main.g "Color Cyan"
    #main.g "Place ";x2Left;" ";y2Upper
    #main.g "Box ";x2Right + 1;" ";y2Lower + 1
    #main.g "Place ";x1Left;" ";y1Lower + 70
    #main.g "Font Times_New_Roman 12 Bold"
    #main.g "\Font Width = ";FontPixelWidth2;" pixels"
    #main.g "\Font Height = ";FontPixelHeight2;" pixels"
    #main.g "Color Black"
    #main.g "\\Stringwidth? = ";FontStringWidth;" pixels"
    #main.g "Flush"
    Call ReleaseDC hWnd(#main.g), hDC
End Sub

Function PixelLong(hDC, xVar, yVar)
    Open "gdi32"for DLL as #gdi
    CallDLL #gdi, "GetPixel",_
        hDC as Ulong,_
        xVar as Long,_
        yVar as Long,_
        PixelLong as Long
   Close #gdi
End Function

Function hDC(handle)
    CallDLL #user32, "GetDC",_
        handle as Ulong,_
        hDC as Ulong
End Function
```

```
Sub ReleaseDC hW, hDC
    CallDLL#user32,"ReleaseDC", _
        hW as Ulong, _
        hDC as Ulong, _
        result as Long
End Sub
```

Disadvantages of Using GetPixel to Determine String Dimensions

There are at least two disadvantages to this method. The first is that the character has to actually be drawn using graphic text. And, the entire font space must stay visible throughout the calculations. The second disadvantage is the speed, or lack thereof, in which each pixel is identified for color. The slowness may prevent the routine from being used 'on the fly' throughout the program. Still, the routine may be of some benefit for those programmers in need of more specific character dimensions.

Posxy or GetPixel: Which is Right for You?

Choose `posxy` when the height of the font space is all that's required. If you absolutely need to know the specific dimensions of the character, excluding the padding, then `GetPixel` is your best option.