

## Screen Capture of a Window

- [Alyce](#)

For an API method, see:

<http://alycesrestaurant.com/windowcapture.htm>

<http://alycesrestaurant.com/desktopcapture.htm>

[Screen Capture of a Window](#) | [GetBmp](#) | [BmpSave](#) | [Coordinates](#) | [Window v Screen Coords](#) | [Window Capture](#) | [DEMO](#) | [End Note](#)

---

## GetBmp

*Using GetBmp to capture a portion of the screen.*

The GetBmp graphics command captures a portion of a graphics window or graphicbox and loads it into memory with the name you specify. The syntax is:

```
#handle "getbmp bmpname x y width height"
```

The actual code looks like this:

```
#graph "getbmp mybmp 5 8 200 143"
```

The x, y, width and height parameters can be anything you want, and you can give the bitmap in memory any name you'd like. The x and y parameters are relative to the upper left corner of the graphics area -- either the graphicbox or the client area of the graphics window. If you use negative x,y coordinates, the screen above and to the left of the graphics area is included in the memory bitmap. **UPDATE: In Windows Vista/7 and above, Liberty BASIC does not capture areas outside of the graphicbox.** To see how this works, run this small program:

```
nomainwin
UpperLeftX=200:UpperLeftY=200
WindowWidth=100:WindowHeight=60
open "test" for graphics as #1
#1 "getbmp screen -206 -230 150 150"
bmpsave "screen", "mycap.bmp"
close #1
```

```
run "mspaint mycap.bmp"  
end
```

The program window is placed at 200,200 on the desktop. The GetBmp command captures a portion of the screen that begins 206 pixels to the left of the graphics area, and 230 pixels above the graphics area. Since it takes a rectangle that is 150x150 pixels, that rectangle doesn't include any part of the program's window!

## BmpSave

*Saving to disk with Bmpsave.*

The BmpSave command saves a named Liberty BASIC bitmap to disk, using the filename specified. The named bitmap can be loaded with LoadBmp or with GetBmp as we are doing here. The syntax for BmpSave is:

```
bmpsave "bmpname", "filename.bmp"
```

The demo saves the memory bitmap to disk, closes the window, then runs MS Paint with the bitmap loaded, so you can view it. You should see the upper left corner of your desktop when you view the demo's bitmap.

## Coordinates

*Coordinates of the program window.*

If we want to do a screen capture of just our program window, we need to know the offset of the upper left corner of our window from the graphics area. We also need to know the width and height of our window. We can't know how many pixels there are between the upper left corner and the upper left corner of the graphics area, because different desktop themes have different sizes for frames, titlebars, etc. If our window includes a sizing frame, we can't know its width and height, either, because the user may have resized it.

If we use a graphics window, the graphics area begins at the upper left corner of the client area. If we are using a graphicbox in another type of window to do the screen capture, it must be located at the 0,0 point in the window, as we've done here. Notice that the graphicbox can have a width and height of 0, so that it isn't even visible, and this method still works.

```
graphicbox #1.g, 0,0,0,0
```

We can use a simple API call to get the coordinates of the upper left corner and the lower right corner of the window at the time we want to do the window capture. We get the window handle with `HWND()`. We also need to create a struct to hold the coordinates. Here, it is named "Rect". We then call the API function `GetWindowRect`.

```
struct Rect,ulx as long, uly as long, lrx as long, lry as long

CallDLL #user32, "GetWindowRect",_
hWin As uLong,_      'window handle
Rect As struct,_     'struct name
result As long
```

After the `GetWindowRect` function returns, the coordinates are contained in the struct, `Rect`. The upper left x coordinate is now in `Rect.ulx.struct` and the upper left y coordinate is in `Rect.uly.struct`. We can assign the values in these struct members to variables, use them in expressions, or we can even use them to fill another struct.

```
LeftX = Rect.ulx.struct
UpperY = Rect.uly.struct
```

## Window v Screen Coords

### *Window Coordinates from Screen Coordinates*

The coordinates returned are relative to the computer screen, not to our program window. There is an API call that converts Screen coords to window coords, called `ScreenToClient`. It works on one point at a time, and it uses a struct, too. We'll create a struct called "Point".

```
struct Point,x As Long, y As Long
```

To get the x,y coordinates of the upper left corner of the window, relative to the upper left corner of the graphics area, we first fill the `Point` struct with the screen coordinates.

```
'fill Point struct with upper left window coords
Point.x.struct=Rect.ulx.struct
```

```
Point.y.struct=Rect.uly.struct
```

When the ScreenToClient function returns, the Point struct contains the coordinates of the upper left corner of the window relative to the graphics area. We'll assign their values to startX and startY, which we'll use when we issue the GetBmp statement.

```
CallDLL #user32, "ScreenToClient",_
hWin As uLong,_      'window handle
Point As struct,_    'struct name
r As Long

startX = Point.x.struct-1
startY = Point.y.struct-1
```

All we need now is the width and height of the window. We can retrieve the width by getting the difference between the rightX and leftX values in the Rect struct. The height can be determined by getting the difference between the lowerY and upperY values in the Rect struct.

```
widthX = Rect.lrx.struct-Rect.ulx.struct
heightY = Rect.lry.struct-Rect.uly.struct
```

## Window Capture

### *Doing the Window Capture*

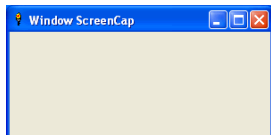
Now that we have the starting and ending coordinates, as well as the width and height, we can issue a GetBmp statement to get the image into memory, then a BmpSave statement to save the screen capture to disk.

```
'getbmp into memory with name = cap
#l.g "getbmp cap ";startX;" ";startY;" ";widthX;" ";heightY

'save bmp to disk
bmpsave "cap", "mycap.bmp"
```

## DEMO

Here is the complete demo of the screen capture of a window. The bitmap looks like this on WindowsXP:



```
nomainwin
UpperLeftX = 20:UpperLeftY=20
WindowWidth=300:WindowHeight=150
'hidden graphicbox at upper left corner of window client area
graphicbox #1.g, 0,0,0,0
open "Window ScreenCap" for window as #1
    #1 "trapclose [quit]"

    'get window handle
    hWin=hwnd(#1)

    'create structs to hold coords
    struct Point,x As Long, y As Long
    struct Rect,ulx as long, uly as long, lrx as long, lry as long

    'get upper left x,y and lower right x,y coords into Rect struct
    CallDLL #user32, "GetWindowRect",_
    hWin As uLong,_      'window handle
    Rect As struct,_     'struct name
    result As long

    'fill Point struct with upper left window coords
    Point.x.struct=Rect.ulx.struct
    Point.y.struct=Rect.uly.struct

    'change screen coords to window client coords
    'the function takes in screen coords in Point struct
    'and replaces them with coords relative to
    'upper left corner of client (graphics) area
    CallDLL #user32, "ScreenToClient",_
    hWin As uLong,_      'window handle
    Point As struct,_    'struct name
    r As Long

    'retrieve coords of upper left corner of
    'window relative to graphics area
    startX = Point.x.struct-1
    startY = Point.y.struct-1
```

```
'get width and height of window
'from coords in Rect struct
'width is rightX - leftX
'height is lowerY - upperY
widthX = Rect.lrx.struct-Rect.ulx.struct
heightY = Rect.lry.struct-Rect.uly.struct

'now we have the coords to use for capturing
'the window with getbmp
'getbmp into memory with name = cap
#1.g "getbmp cap ";startX;" ";startY;" ";widthX;" ";heightY

'save bmp to disk
bmpsave "cap", "mycap.bmp"

'run mspaint and look at our bmp
run "mspaint mycap.bmp"

'for this demo, close program window right away
[quit] close #1: end
```

## End Note

When we write code that is meant to demonstrate a method or instruct people, we often add a lot of comments. We also break API calls into single arguments with comments, by using a line continuation character. API calls must be all on one line. Here is the code for the demo. It only takes 21 lines to show this code when the comments are removed. It's really not long or complicated!

```
nomainwin
UpperLeftX = 20:UpperLeftY=20
WindowWidth=300:WindowHeight=150
graphicbox #1.g, 0,0,0,0
open "Window ScreenCap" for window as #1
#1 "trapclose [quit]"
hWin=hwnd(#1)
struct Point,x As Long, y As Long
struct Rect,ulx as long, uly as long, lrx as long, lry as long
CallDLL #user32, "GetWindowRect",hWin As uLong,Rect As struct,
result As long
Point.x.struct=Rect.ulx.struct
Point.y.struct=Rect.uly.struct
CallDLL #user32, "ScreenToClient",hWin As uLong,Point As struct,r As
Long
```

```
startX = Point.x.struct-1
startY = Point.y.struct-1
widthX = Rect.lrx.struct-Rect.ulx.struct
heightY = Rect.lry.struct-Rect.uly.struct
#1.g "getbmp cap ";startX;" ";startY;" ";widthX;" ";heightY
bmpsave "cap", "mycap.bmp"
run "mspaint mycap.bmp"
[quit] close #1: end
```