BitBlt function

Applies to: desktop apps only

The **BitBlt** function performs a bit-block transfer of the color data corresponding to a rectangle of pixels from the specified source device context into a destination device context.

Syntax

```
BOOL BitBlt(
  __in  HDC hdcDest,
  __in  int nXDest,
  __in  int nYDest,
  __in  int nWidth,
  __in  int nHeight,
  __in  HDC hdcSrc,
  __in  int nXSrc,
  __in  int nYSrc,
  __in  DWORD dwRop
);
```

Parameters

hdcDest [in]

A handle to the destination device context.

nXDest [in]

The x-coordinate, in logical units, of the upper-left corner of the destination rectangle.

nYDest [in]

The y-coordinate, in logical units, of the upper-left corner of the destination rectangle.

nWidth [in]

The width, in logical units, of the source and destination rectangles.

nHeight [in]

The height, in logical units, of the source and the destination rectangles.

hdcSrc [in]

A handle to the source device context.

nXSrc [in]

The x-coordinate, in logical units, of the upper-left corner of the source rectangle.

nYSrc [in]

The y-coordinate, in logical units, of the upper-left corner of the source rectangle.

dwRop [in]

A raster-operation code. These codes define how the color data for the source rectangle is to be combined with the color data for the destination rectangle to achieve the final color.

The following list shows some common raster operation codes.

Value	Meaning
BLACKNESS	Fills the destination rectangle using the color associated with index 0 in the physical palette. (This color is black for the default physical palette.)
CAPTUREBLT	Includes any windows that are layered on top of your window in the resulting image. By default, the image only contains your window. Note that this generally cannot be used for printing device contexts.
DSTINVERT	Inverts the destination rectangle.
MERGECOPY	Merges the colors of the source rectangle with the brush currently selected in *hdcDest*, by using the Boolean AND operator.
MERGEPAINT	Merges the colors of the inverted source rectangle with the colors of the destination rectangle by using the Boolean OR operator.
NOMIRRORBITMAP	Prevents the bitmap from being mirrored.
NOTSRCCOPY	Copies the inverted source rectangle to the destination.
NOTSRCERASE	Combines the colors of the source and destination rectangles by using the Boolean OR operator and then inverts the resultant color.
PATCOPY	Copies the brush currently selected in *hdcDest*, into the destination bitmap.
PATINVERT	Combines the colors of the brush currently selected in *hdcDest*, with the colors of the destination rectangle by using the Boolean XOR operator.
PATPAINT	Combines the colors of the brush currently selected in *hdcDest*, with the colors of the inverted source rectangle by using the Boolean OR operator. The result of this operation is combined with the colors of the destination rectangle by using the Boolean OR operator.
SRCAND	Combines the colors of the source and destination rectangles by using the Boolean AND operator.
SRCCOPY	Copies the source rectangle directly to the destination rectangle.
SRCERASE	Combines the inverted colors of the destination rectangle with the colors of the source rectangle by using the Boolean AND operator.
SRCINVERT	Combines the colors of the source and destination rectangles by using the Boolean XOR operator.
SRCPAINT	Combines the colors of the source and destination rectangles by using the Boolean OR operator.

WHITENESS	Fills the destination rectangle using the color associated with index 1 in the physical palette. (This color is white for the default physical palette.)

Return value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call **GetLastError**[1].

Remarks

BitBlt only does clipping on the destination DC.

If a rotation or shear transformation is in effect in the source device context, **BitBlt** returns an error. If other transformations exist in the source device context (and a matching transformation is not in effect in the destination device context), the rectangle in the destination device context is stretched, compressed, or rotated, as necessary.

If the color formats of the source and destination device contexts do not match, the **BitBlt** function converts the source color format to match the destination format.

When an enhanced metafile is being recorded, an error occurs if the source device context identifies an enhanced-metafile device context.

Not all devices support the **BitBlt** function. For more information, see the RC_BITBLT raster capability entry in the **GetDeviceCaps**[2] function as well as the following functions: **MaskBlt**[3], **PlgBlt**[4], and **StretchBlt**[5].

BitBlt returns an error if the source and destination device contexts represent different devices. To transfer data between DCs for different devices, convert the memory bitmap to a DIB by calling **GetDIBits**[6]. To display the DIB to the second device, call **SetDIBits**[7] or **StretchDIBits**[8].

ICM: No color management is performed when blits occur.

Examples

For an example, see Capturing an Image[9].

Requirements

Minimum supported client	Windows 2000 Professional
Minimum supported server	Windows 2000 Server
Header	Wingdi.h (include Windows.h)
Library	Gdi32.lib
DLL	Gdi32.dll

See also

Bitmaps Overview[10]
Bitmap Functions[11]

GetDeviceCaps[2]
GetDIBits[6]
MaskBlt[3]
PlgBlt[4]
SetDIBits[7]
StretchBlt[5]
StretchDIBits[8]

Send comments about this topic to Microsoft[12]

Build date: 3/7/2012

Links Table

[1]*http://msdn.microsoft.com/en-us/library/ms679360(v=vs.85).aspx*

[2]*http://msdn.microsoft.com/en-us/library/dd144877(v=vs.85).aspx*

[3]*http://msdn.microsoft.com/en-us/library/dd145047(v=vs.85).aspx*

[4]*http://msdn.microsoft.com/en-us/library/dd162804(v=vs.85).aspx*

[5]*http://msdn.microsoft.com/en-us/library/dd145120(v=vs.85).aspx*

[6]*http://msdn.microsoft.com/en-us/library/dd144879(v=vs.85).aspx*

[7]*http://msdn.microsoft.com/en-us/library/dd162973(v=vs.85).aspx*

[8]*http://msdn.microsoft.com/en-us/library/dd145121(v=vs.85).aspx*

[9]*http://msdn.microsoft.com/en-us/library/dd183402(v=vs.85).aspx*

[10]*http://msdn.microsoft.com/en-us/library/dd183377(v=vs.85).aspx*

[11]*http://msdn.microsoft.com/en-us/library/dd183385(v=vs.85).aspx*

[12]**Send comments about this topic to Microsoft**;
mailto:wsddocfb@microsoft.com?subject=Documentation%20feedback%20%5Bgdi%5Cgdi%5D:%20BitBlt%20function%20%20RELEASE:%20(3/7/2012)&body=%0A%0APRIVACY%20STATEMENT%0A%0AThe%20SDK%20team%20uses%20the%20feedback%20submitted%20to%20improve%20the%20SDK%20documentation.%20We%20do%20not%20use%20your%20email%20address%20for%20any%20other%20purpose.%20We%20will%20remove%20your%20email%20address%20from%20our%20system%20after%20the%20issue%20you%20are%20reporting%20has%20been%20resolved.%20While%20we%20are%20working%20to%20resolve%20this%20issue,%20we%20may%20send%20you%20an%20email%20message%20to%20request%20more%20information%20about%20your%20feedback.%20After%20the%20issues%20have%20been%20addressed,%20we%20may%20send%20you%20an%20email%20message%20to%20let%20you%20know%20that%20your%20feedback%20has%20been%20addressed.%0A%0AFor%20more%20information%20about%20Microsoft's%20privacy%20policy,%20see%20http://privacy.microsoft.com/en-us/default.aspx.

Community Content

A few notes

Note well that if you have a resulting BitMap image that contains more and more black space (image

is shifted down or right), it may be because you are using nXDest instead of nXSrc for your starting offset. If you want to grab an offset from the screen, you use nXSrc. Just experiment with them until you get the right one.

Also note that there may be more efficient ways to get the screen capture: http://www.codeproject.com/KB/dialog/screencap.aspx (LGPL)

Also note that for me, disabling aero ("Enable desktop composition" in my system properties -> advanced -> Performance Options -> Visual effects) made this method call *much* faster. There may be other tweaks that matter as well, speed-wise. Also using Bitblt against a DC from the Desktop may be slower than say the hwnd of a window, with aero on.

Also note that with aero on, you can capture the bitmap of a window with its HWND, even if it's obscured or off screen. Of course, if it has any subwindows they'll probably just show up as white squares, so PrintWindow may help there.

Also note that "translucent" windows may not be captured with Aero turned on unless you use CAPTUREBLT http://stackoverflow.com/questions/1193335/screen-capture-ignores-some-windows

9/20/2011
rogerdpack2

7/12/2010
rogerdpack2

CAPTUREBLT not needed when Aero is on

$0If Aero is activce BitBlt() will capture transparent windows even if the CAPTUREBLT flag isn't set. This is nice as the CAPTUREBLT flag causes the mouse to disapear during the BitBlt(), but you still get the transparency. This only works with BitBlt'ing from a DC from the "DISPLAY" or A DC from the Desktop's HWND, if you BitBlt from the DC created from the HWND of some arbitrary window, with Aero on, it will just capture that windows with "black" in place of any aero transparency (i.e. most of the title bar). With or without CAPTUREBLT (which is weird). $0

9/20/2011
rogerdpack

4/28/2010
Daniel Reinhard

"Ternary Raster Operations"

http://msdn.microsoft.com/en-us/library/dd145130(VS.85).aspx

"Ternary Raster Operations" - Every other ROP that isn't defined in wingdi.h.

8/14/2011
Elliot_009

dwRop Values

From wingdi.h:

```
#define BLACKNESS 0x42
#define DSTINVERT 0x550009
#define MERGECOPY 0xC000CA
#define MERGEPAINT 0xBB0226
#define NOTSRCCOPY 0x330008
#define NOTSRCERASE 0x1100A6
#define PATCOPY 0xF00021
#define PATINVERT 0x5A0049
#define PATPAINT 0xFB0A09
#define SRCAND 0x8800C6
```

```
#define SRCCOPY 0xCC0020
#define SRCERASE 0x440328
#define SRCINVERT 0x660046
#define SRCPAINT 0xEE0086
#define WHITENESS 0xFF0062
```

8/11/2009
Phil Shirey