BMP file format

From Wikipedia, the free encyclopedia

Windows Bitmap

Filename extension	`.bmp or .dib`
Internet media type	`image/x-ms-bmp` (unofficial) or `image/x-bmp` (unofficial)
Type code	`'BMP '` `'BMPf'` `'BMPp'`
Uniform Type Identifier	com.microsoft.bmp
Type of format	Raster graphics

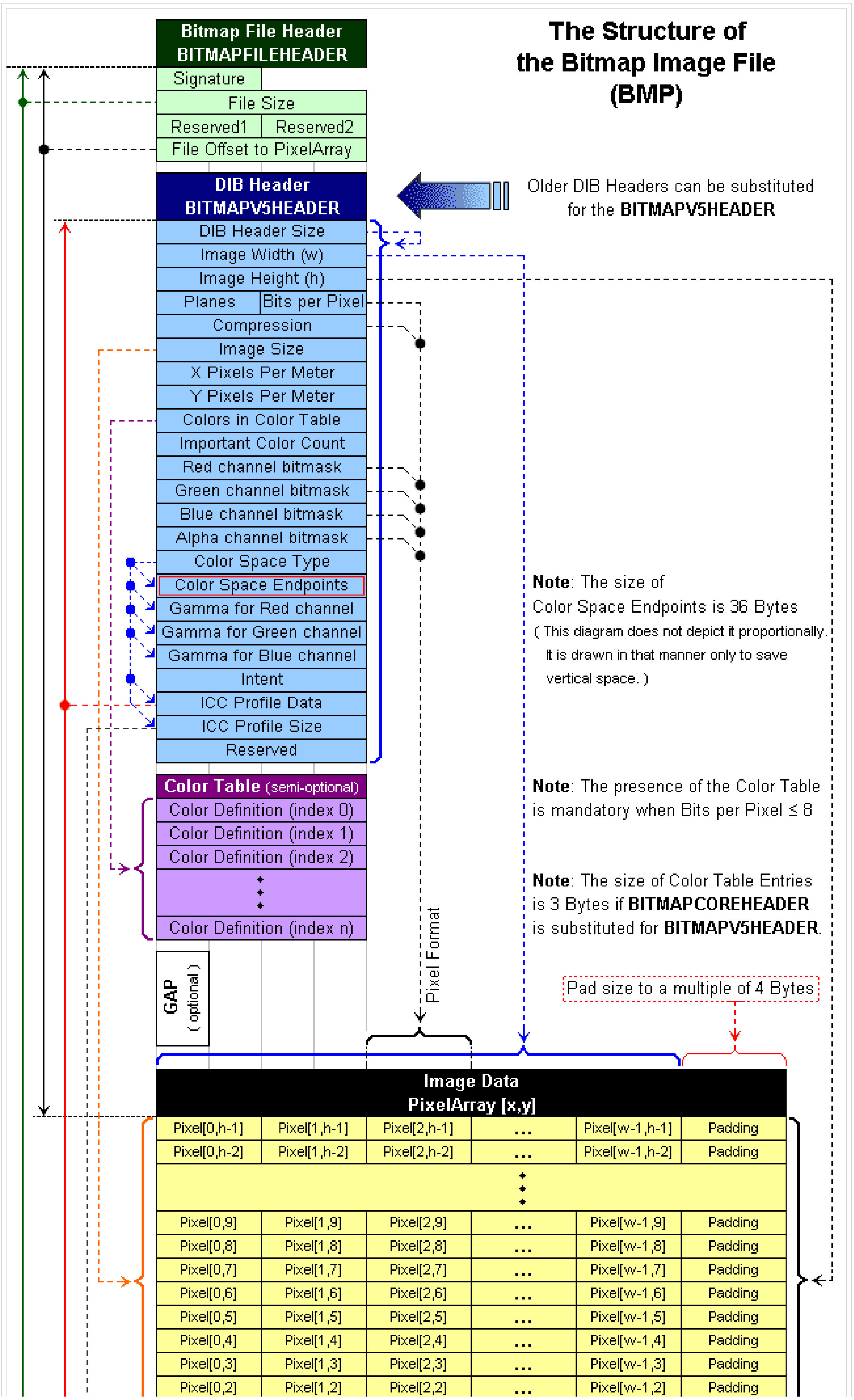
The **BMP File Format**, also known as **Bitmap Image File** or **Device Independent Bitmap (DIB) file format** or simply a **Bitmap**, is an Raster graphics image file format used to store bitmap digital images, independently of the display device (such as a graphics adapter), especially on Microsoft Windows and OS/2 operating systems.

The BMP File Format is capable of storing:

- 2D digital images,
- monochrome and color images,
- Images of arbitrary width and height,
- Both compressed and uncompressed images,
- Images in various color depths,
- Images in various resolutions,
- Images with embedded Alpha channel,
- Images with embedded color profiles.

Contents

File Structure


The Structure of the Bitmap Image File (BMP)
Bitmap File Header
BITMAPFILEHEADER
Signature
File Size
Reserved1
Reserved2
File Offset to PixelArray
DIB Header
BITMAPV5HEADER
Older DIB Headers can be substituted for the BITMAPV5HEADER
DIB Header Size
Image Width (w)
Image Height (h)
Planes
Bits per Pixel
Compression
Image Size
X Pixels Per Meter
Y Pixels Per Meter
Colors in Color Table
Important Color Count
Red channel bitmask
Green channel bitmask
Blue channel bitmask
Alpha channel bitmask
Color Space Type
Color Space Endpoints
Gamma for Red channel
Gamma for Green channel
Gamma for Blue channel
Intent
ICC Profile Data
ICC Profile Size
Reserved
Note: The size of Color Space Endpoints is 36 Bytes
(This diagram does not depict it proportionally. It is drawn in that manner only to save vertical space.)
Color Table (semi-optional)
Color Definition (index 0)
Color Definition (index 1)
Color Definition (index 2)
Color Definition (index n)
Note: The presence of the Color Table is mandatory when Bits per Pixel ≤ 8
Note: The size of Color Table Entries is 3 Bytes if BITMAPCOREHEADER is substituted for BITMAPV5HEADER.
GAP (optional)
Pixel Format
Pad size to a multiple of 4 Bytes
Image Data
PixelArray [x,y]
Pixel[0,h-1]
Pixel[1,h-1]
Pixel[2,h-1]
...
Pixel[w-1,h-1]
Padding
Pixel[0,h-2]
Pixel[1,h-2]
Pixel[2,h-2]
Pixel[w-1,h-2]
Pixel[0,9]
Pixel[1,9]
Pixel[2,9]
Pixel[w-1,9]
Pixel[0,8]
Pixel[1,8]
Pixel[2,8]
Pixel[w-1,8]
Pixel[0,7]
Pixel[1,7]
Pixel[2,7]
Pixel[w-1,7]
Pixel[0,6]
Pixel[1,6]
Pixel[2,6]
Pixel[w-1,6]
Pixel[0,5]
Pixel[1,5]
Pixel[2,5]
Pixel[w-1,5]
Pixel[0,4]
Pixel[1,4]
Pixel[2,4]
Pixel[w-1,4]
Pixel[0,3]
Pixel[1,3]
Pixel[2,3]
Pixel[w-1,3]
Pixel[0,2]
Pixel[1,2]
Pixel[2,2]
Pixel[w-1,2]

The **Bitmap Image File** consists of fixed-size structures (headers) as well as variable-size structures appearing in a predetermined sequence. Many different versions of some of these structures can appear in the file, due to the long evolution of this file format.

Referring to the Diagram 1, the **Bitmap Image File** is composed of structures in the following order:

Structure Name	Optional	Size	Purpose	Comments
Bitmap File Header	No	14 Bytes	To store general information about the Bitmap Image File	Not needed after the file is loaded in memory
DIB Header	No	Fixed-size (however 7 different versions exist)	To store detailed information about the bitmap image	Immediately follows the Bitmap File Header
Extra bit masks	Yes	3 or 4 DWORDs[1] (12 or 16 Bytes)	To define the pixel format	Present only in case the DIB Header is the BITMAPINFOHEADER
Color Table	Semi-optional	Variable-size	To define colors used by the bitmap image	Mandatory for color depths <= 8
Gap	Yes	Variable-size	Structure alignment	An artifact of the File Offset to PixelArray in the Bitmap File Header
Pixel Array	No	Variable-size	To define the actual values of the pixels	The pixel format is defined by the DIB Header. Each row in the Pixel Array is padded to a multiple of 4 bytes in size
ICC Color Profile	Yes	Variable-size	To define the color profile for Color management	Can also contain a path to an external file containing the Color Profile

Device-independent bitmaps and BMP file format

Microsoft has defined a particular representation of color bitmaps of different color depths, as an aid to exchanging bitmaps between devices and applications with a variety of internal representations. They called these device-independent bitmaps or DIBs, and the file format for them is called DIB file format or BMP file format. According to Microsoft support:[2]

> A device-independent bitmap (DIB) is a format used to define device-independent bitmaps in various color resolutions. The main purpose of DIBs is to allow bitmaps to be moved from one device to another (hence, the device-independent part of the name). A DIB is an external format, in contrast to a device-dependent bitmap, which appears in the system as a bitmap object (created by an application...). A DIB is normally transported in metafiles (usually using the StretchDIBits() function), BMP files, and the Clipboard (CF_DIB data format).

The following sections discuss the data stored in the BMP file or DIB in details. This is the standard BMP file format.[2] Some bitmap images may be stored using a slightly different format, depending on the application that creates it. Also, not all fields are used; a value of 0 will be found in these unused fields.

DIBs in memory

A Bitmap Image File loaded into memory becomes a DIB data structure - an important component of the

Windows GDI API. The DIB data structure is almost the same as the BMP file format, but without the 14-byte Bitmap File Header. The Pixel Array must begin at a memory address that is a multiple of 4. Also, the optional Color Profile should be located after the Color Table and before the Gap and Pixel Array, when the image is loaded in memory.

Bitmap File Header

This block of bytes is at the start of the file and is used to identify the file. A typical application reads this block first to ensure that the file is actually a BMP file and that it is not damaged. The first two bytes of the BMP file format are the character 'B' then the character 'M' in 1-byte ASCII encoding. All of the integer values are stored in little-endian format (i.e. least-significant byte first).

Offset#	Size	Purpose
0000_h	2 bytes	the magic number used to identify the BMP file, usually 0x42 0x4D in hex, same as **BM** in ASCII. The following entries are possible: ▪ **BM** – Windows 3.1x, 95, NT, ... etc. ▪ **BA** – OS/2 Bitmap Array ▪ **CI** – OS/2 Color Icon ▪ **CP** – OS/2 Color Pointer ▪ **IC** – OS/2 Icon ▪ **PT** – OS/2 Pointer
0002_h	4 bytes	the size of the BMP file in bytes
0006_h	2 bytes	reserved; actual value depends on the application that creates the image
0008_h	2 bytes	reserved; actual value depends on the application that creates the image
$000A_h$	4 bytes	the offset, i.e. starting address, of the byte where the bitmap data can be found.

Equivalent C++ -Language Header

```
#include <stdint.h>

/* Note: the magic number has been removed from the bmpfile_header structu
   since it causes alignment problems
     struct bmpfile_magic should be written/read first
   followed by the
     struct bmpfile_header
   [this avoids compiler-specific alignment pragmas etc.]
*/

struct bmpfile_magic {
  unsigned char magic[2];
};

struct bmpfile_header {
  uint32_t filesz;
  uint16_t creator1;
  uint16_t creator2;
  uint32_t bmp_offset;
};
```

DIB Header (Bitmap Information Header)

This block of bytes tells the application detailed information about the image, which will be used to display the image on the screen. The block also matches the header used internally by Windows and OS/2 and has several different variants. All of them contain a dword (32 bit) field, specifying their size, so that an application can easily determine which header is used in the image. The reason that there are different headers is that Microsoft extended the DIB format several times. The new extended headers can be used with some GDI functions instead of the older ones, providing more functionality. Since the GDI supports a function for loading bitmap files, typical Windows applications use that functionality. One consequence of this is that for such applications, the BMP formats that they support match the formats supported by the Windows version being run. See the table below for more information.

Size	Header Name	OS support	Features Added	in Bitmap Files Written by
12	BITMAPCOREHEADER OS21XBITMAPHEADER	OS/2 and also all Windows versions since Windows 3.0		
64	BITMAPCOREHEADER2 OS22XBITMAPHEADER	OS/2 and also all Windows versions since Windows 3.0	Adds Halftoning. Adds RLE and Huffman 1D compression.	
40	BITMAPINFOHEADER	all Windows versions since Windows 3.0	Adds RLE Bitmap Compression. Adds 16bpp and 32bpp pixel formats. Adds optional RGB bitmasks.	Adobe Photoshop
52	BITMAPV2INFOHEADER	Undocumented.	Adds mandatory RGB bitmasks, a.k.a. BITFIELDS.	
56	BITMAPV3INFOHEADER	Undocumented.	Adds mandatory RGBA bitmasks, a.k.a. BITFIELDS.	Adobe Photoshop

108	BITMAPV4HEADER	all Windows versions since Windows 95/NT4	Adds Color space and Gamma	
124	BITMAPV5HEADER	Windows 98/2000 and newer	Adds Color Profiles	

Versions after BITMAPCOREHEADER only add fields to the end of the header of the previous version. For example: BITMAPV2INFOHEADER adds fields to BITMAPINFOHEADER and BITMAPV3INFOHEADER adds fields to BITMAPV2INFOHEADER (or, BITMAPINFOHEADER is a stripped version of BITMAPV2INFOHEADER)

For compatibility reasons, most applications use the older DIB headers for saving files. With OS/2 being obsolete, for now the common format is the BITMAPINFOHEADER header. See next table for its description. All values are stored as unsigned integers, unless explicitly noted.

Offset #	Size	Purpose
0Eh	4	the size of this header (40 bytes)
12h	4	the bitmap width in pixels (signed integer).
16h	4	the bitmap height in pixels (signed integer).
1Ah	2	the number of color planes being used. Must be set to 1.
1Ch	2	the number of bits per pixel, which is the color depth of the image. Typical values are 1, 4, 8, 16, 24 and 32.
1Eh	4	the compression method being used. See the next table for a list of possible values.
22h	4	the image size. This is the size of the raw bitmap data (see below), and should not be confused with the file size.
26h	4	the horizontal resolution of the image. (pixel per meter, signed integer)
2Ah	4	the vertical resolution of the image. (pixel per meter, signed integer)
2Eh	4	the number of colors in the color palette, or 0 to default to 2^n.
32h	4	the number of important colors used, or 0 when every color is important; generally ignored.

Note: The image size field can be 0 for BI_RGB bitmaps.

Equivalent C-Language Header

```
typedef struct {
  uint32_t header_sz;
  int32_t width;
  int32_t height;
  uint16_t nplanes;
  uint16_t bitspp;
  uint32_t compress_type;
  uint32_t bmp_bytesz;
  int32_t hres;
  int32_t vres;
  uint32_t ncolors;
  uint32_t nimpcolors;
} BITMAPINFOHEADER;
```

The compression method field (bytes #30-33) can have the following values:

Value	Identified by	Compression method	Comments
0	BI_RGB	none	Most common
1	BI_RLE8	RLE 8-bit/pixel	Can be used only with 8-bit/pixel bitmaps
2	BI_RLE4	RLE 4-bit/pixel	Can be used only with 4-bit/pixel bitmaps
3	BI_BITFIELDS	Bit field or Huffman 1D compression for BITMAPCOREHEADER2	Pixel format defined by bit masks or Huffman 1D compressed bitmap for BITMAPCOREHEADER2
4	BI_JPEG	JPEG or RLE-24 compression for BITMAPCOREHEADER2	The bitmap contains a JPEG image or RLE-24 compressed bitmap for BITMAPCOREHEADER2
5	BI_PNG	PNG	The bitmap contains a PNG image

Note: BI_JPEG and BI_PNG are for printer drivers and are not supported when rendering to the screen.[3]

Equivalent C-Language Header (Compression Methods)

```
typedef enum {
  BI_RGB = 0,
  BI_RLE8,
  BI_RLE4,
  BI_BITFIELDS, //Also Huffman 1D compression for BITMAPCOREHEADER2
  BI_JPEG,      //Also RLE-24 compression for BITMAPCOREHEADER2
  BI_PNG,
} bmp_compression_method_t;
```

The OS/2 BITMAPCOREHEADER header is also popular:

Offset	Size	Purpose
Eh	4	the size of this header (12 bytes)
12h	2	the bitmap width in pixels.

14h	2	the bitmap height in pixels.
16h	2	the number of color planes; 1 is the only legal value
18h	2	the number of bits per pixel. Typical values are 1, 4, 8 and 24.

Note: OS/2 BITMAPCOREHEADER bitmaps cannot be compressed and cannot be 16 or 32 bits/pixel. All values in the OS/2 BITMAPCOREHEADER header are unsigned integers.

A 16-bit and 32-bit version of DIB with an integrated alpha channel has been introduced with the undocumented BITMAPV3INFOHEADER and with the documented BITMAPV4HEADER (since Windows 95) and is used within Windows XP logon and theme system as well as Microsoft Office (since v2000); it is supported by some image editing software, such as Adobe Photoshop since version 7 and Adobe Flash since version MX 2004 (then known as Macromedia Flash). It is also supported by GIMP, Google Chrome, MS-PowerPoint and MS-Word.

Color Table

The Color Table (a.k.a. Palette) occurs in the BMP image file directly after the BMP file header, the DIB header (and after optional three Red, Green and Blue bitmasks if the BITMAPINFOHEADER Header with BI_BITFIELDS option is used). Therefore, its offset is the size of the BITMAPFILEHEADER plus the size of the DIB header (plus optional 12 bytes for the three bit masks).
*Note: On Windows CE the BITMAPINFOHEADER Header can be used with the BI_ALPHABITFIELDS[1]option in the biCompression member. In such case, **four** optional bitmasks follow the BITMAPINFOHEADER Header instead of the three bitmask mentioned above, thus the Color Table's offset is the size of the BITMAPFILEHEADER plus the size of the BITMAPINFOHEADER Header plus the 16 bytes of the four bitmasks (Red, Green, Blue and Alpha).*

The number of entries in the palette is either 2^n or a smaller number specified in the header (in the OS/2 BITMAPCOREHEADER header format, only the full-size palette is supported).[2][4].

The Color Table is a block of bytes (a table) listing the colors used by the image. Each pixel in an indexed color image is described by a number of bits (1, 4, or 8) which is an index of a single color described by this table. The purpose of the color palette in indexed color bitmaps is to inform the application about the actual color that each of these index values corresponds to. The purpose of the Color Table in non-indexed (non-palettized) bitmaps is to list the colors used by the bitmap for the purposes of optimization on devices with limited color display capability and to facilitate future conversion to different pixel formats and paletization.

The colors in the Color Table are specified in the 4-byte per entry 8.8.8.0.8 format (in RGBAX notation) with an exception of the OS/2 BITMAPCOREHEADER's Color Table which uses the 3-byte per entry 8.8.8.0.0 format[2][4].
Microsoft **does not disallow** the presence of a valid Alpha channel bit mask[5] in BITMAPV4HEADER and BITMAPV5HEADER for 1bpp, 4bpp and 8bpp indexed color images, which indicates that the Color Table entries can also specify an Alpha component using the 8.8.8.[0-8].[0-8] format via the RGBQUAD.rgbReserved[6] member. However, some versions of Microsoft's documentation disallow this feature by stating that the RGBQUAD.rgbReserved member "must be zero".

As mentioned above, the Color Table is normally not used when the pixels are in the 16-bit per pixel (16bpp) format (and higher); there are normally no Color Table entries in those bitmap image files. However, the Microsoft documentation (on the MSDN web site as of Nov. 16, 2010) specifies that for 16bpp (and higher), the Color Table can be present to store a list of colors intended for optimization on devices with limited color display capability, while it also specifies, that in such cases, no indexed palette entries are present in this Color Table. This may seem like a contradiction if no distinction is made between the mandatory palette entries and the optional color list.

Pixel Storage

The bits representing the bitmap pixels are packed in rows. The size of each row is rounded up to a multiple of 4 bytes (a 32-bit DWORD) by padding.
For images with Height > 1, multiple padded rows are stored consecutively, forming a Pixel Array.

The total number of bytes necessary to store one row of pixels can be calculated as: **CEILING(ImageWidth * BitsPerPixel / 32) * 4**

$$\text{RowSize} = \left\lceil \frac{\text{BitsPerPixel} \cdot \text{ImageWidth}}{32} \right\rceil \cdot 4,$$

ImageWidth is expressed in pixels.

The total amount of bytes necessary to store an array of pixels in an ***n*** Bits per Pixel (bpp) image, with 2^n colors, can be calculated by accounting for the effect of rounding up the size of each row to a multiple of a 4 bytes, as follows:

$$\text{PixelArraySize} = \text{RowSize} \cdot |\text{ImageHeight}|$$

ImageHeight is expressed in pixels. The absolute value is necessary because ImageHeight can be negative

The total Bitmap Image File size can be approximated as:

$$\text{FileSize} \approx 54 + 4 \cdot 2^{\text{bpp}} + \text{PixelArraySize},$$

for $BPP \leq 8$ (because for pixels larger than 8 bits, the palette is not mandatory)

Only images with 8 or fewer Bits per Pixel must account for the palette. 16bpp images (or higher), may omit the palette part from the size calculation, as follows:

$$\text{FileSize} \approx 54 + \text{PixelArraySize},$$

for *Bits per Pixel* > 8.

In the formulas above, the number **54** is the combined size of the 14 byte Bitmap File Header and the 40 byte popular Windows DIB Header - the BITMAPINFOHEADER (some other DIB Header versions will be larger or smaller than that as described by the table above) and the expression $4 \cdot 2^n$ is the size of the color palette in bytes.

This total file size formula is only an approximation, since the size of the color palette will be $3 \cdot 2^n$ bytes for the OS/2 DIB Header version BITMAPCOREHEADER, and some files may define only the number of colors needed by the image, potentially fewer than 2^n.[2].
An additional size uncertainty is introduced by the optional presence of the 12 or 16 bytes needed for the extra bit masks stored immediately after the BITMAPINFOHEADER DIB Header and the variable-size GAP depiced in Diag.1

Pixel Array (bitmap data)

The Pixel Array is a block of 32-bit DWORDs, that describes the image pixel by pixel. Normally pixels are stored "upside-down" with respect to normal image raster scan order, starting in the lower left corner, going from left to right, and then row by row from the bottom to the top of the image.[2]
Uncompressed Windows bitmaps also can be stored from the top to bottom, when the Image Height value is

negative.

In the original OS/2 DIB, the only four legal values of color depth were 1, 4, 8, and 24 bits per pixel (**bpp**).[2] Contemporary DIB Headers allow pixel formats with 1, 4, 8, 16, 24 and 32 bits per pixel (bpp).[7]

Padding bytes (not necessarily 0) must be appended to the end of the rows in order to bring up the length of the rows to a multiple of four bytes. When the Pixel Array is loaded into memory, each row must begin at a memory address that is a multiple of 4. This address/offset restriction is mandatory **only** for Pixel Array's loaded in memory. For file storage purposes, **only the size** of each row must be a multiple of 4 bytes while the file offset can be arbitrary.[2]
A 24-bit bitmap with Width=1, would have 3 bytes of data per row (blue, green, red) and 1 byte of padding, while Width=2 would have 2 bytes of padding, Width=3 would have 3 bytes of padding, and Width=4 would not have any padding at all.

Bitmap Image Files are typically much larger than image file formats compressed with other algorithms, for the same image. For example, the 1058×1058 Wikipedia logo, which occupies about 271 KB in the lossless PNG format, takes about 3358 KB as a 24bpp BMP Image File. Uncompressed formats are generally unsuitable for transferring images on the Internet or other slow or capacity-limited media.

Compression

Indexed color images may be compressed with 4-bit or 8-bit RLE or Huffman 1D algorithm.
OS/2 BITMAPCOREHEADER2 24bpp images may be compressed with the 24-bit RLE algorithm.
The 16bpp and 32bpp images are *always* stored uncompressed.
Note that images in all color depths can be stored without compression if so desired.

Pixel Format

In bitmap image file on a disk or bitmap image in memory, the pixels can be defined by varying number of bits.

- The 1-bit per pixel (1bpp) format supports 2 distinct colors, (for example: black and white, ...or yellow and pink). The pixel values are stored in each bit, with the first (left-most) pixel in the most-significant bit of the first byte.[2] Each bit is an index into a table of 2 colors. This Color Table is in 32bpp 8.8.8.0.8 RGBAX format. An unset bit will refer to the first color table entry, and a set bit will refer to the last (second) color table entry.

- The 4-bit per pixel (4bpp) format supports 16 distinct colors and stores 2 pixels per 1 byte, the left-most pixel being in the more significant nibble.[2] Each pixel value is a 4-bit index into a table of up to 16 colors. This Color Table is in 32bpp 8.8.8.0.8 RGBAX format.

- The 8-bit per pixel (8bpp) format supports 256 distinct colors and stores 1 pixel per 1 byte. Each byte is an index into a table of up to 256 colors. This Color Table is in 32bpp 8.8.8.0.8 RGBAX format.

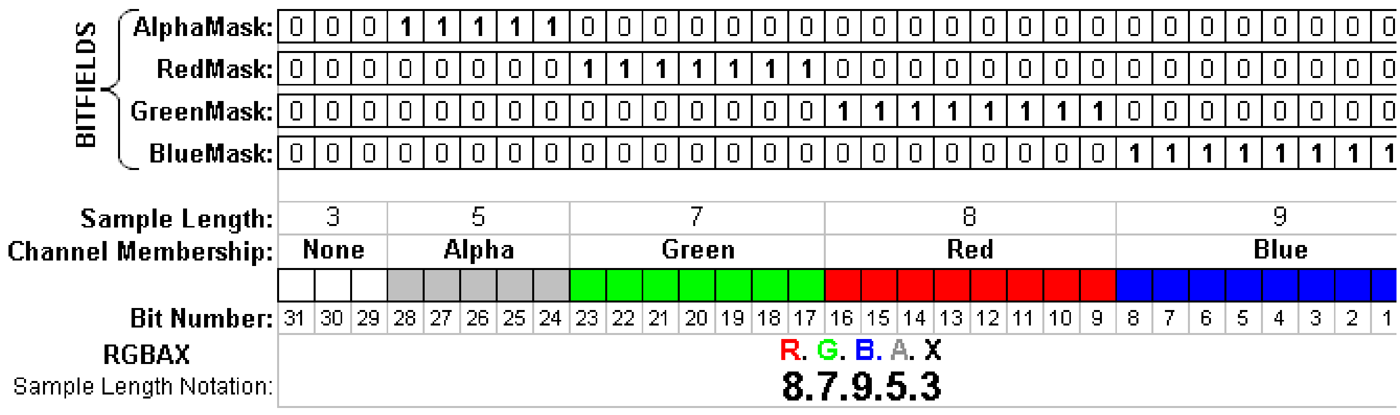
- The 16-bit per pixel (16bpp) format supports 65536 distinct colors and stores 1 pixel per 2 byte WORD. Each WORD can define the Alpha, Red, Green and Blue samples of the pixel.

- The 24-bit pixel (24bpp) format supports 16,777,216 distinct colors and stores 1 pixel value per 3 bytes. Each pixel value defines the Red, Green and Blue samples of the pixel (8.8.8.0.0 in RGBAX notation). Specifically in the order (Blue, Green and Red, 8-bits per each sample).[2]

- The 32-bit per pixel (32bpp) format supports 4,294,967,296 distinct colors and stores 1 pixel per 4 byte DWORD. Each DWORD can define the Alpha, Red, Green and Blue samples of the pixel.

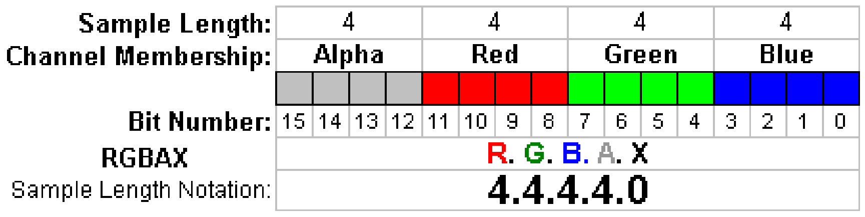
In order to resolve the ambiguity of **which bits define which samples**, the DIB Headers provide certain defaults

as well as specific BITFIELDS which are bit masks that define the membership of particular group of bits in a pixel to a particular channel.
The following diagram defines this mechanism:


BITFIELDS
AlphaMask: 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
RedMask: 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
GreenMask: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
BlueMask: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1
Sample Length: 3 5 7 8 9
Channel Membership: None Alpha Green Red Blue
Bit Number: 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
RGBAX
Sample Length Notation:
R. G. B. A. X
8.7.9.5.3

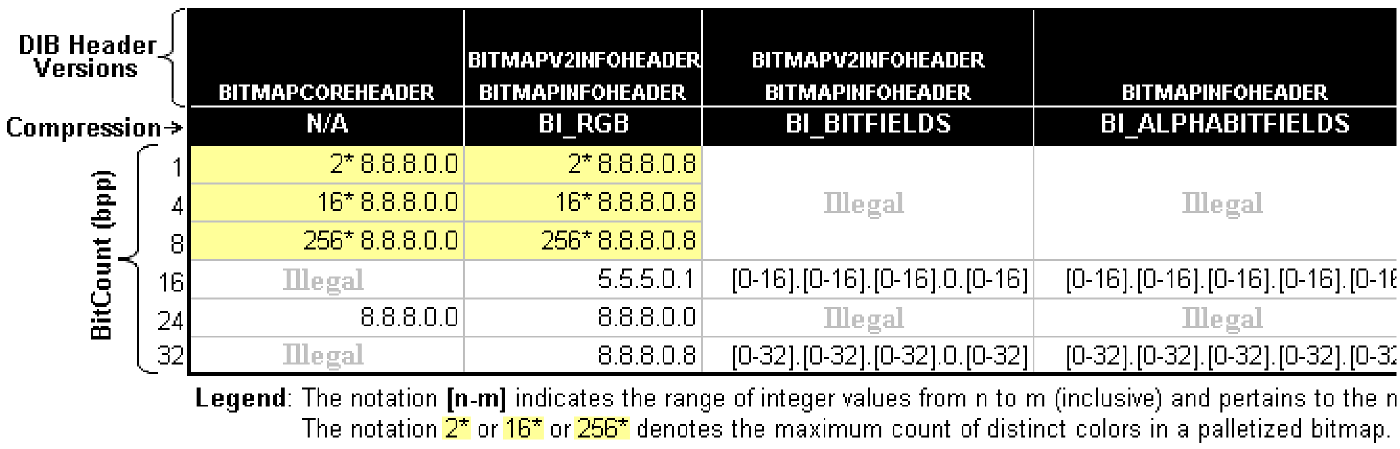
Diag.2 - The BITFIELDS mechanism for a 32 bit pixel depicted in RGBAX Sample Length notation

The sample fields defined by the BITFIELDS bit masks have to be contiguous and non-overlapping.
The Red, Green and Blue bit masks are valid only when the Compression member of the DIB Header is set to BI_BITFIELDS.
The Alpha bit mask is valid whenever it is present in the DIB Header or when the Compression member of the DIB Header is set to BI_ALPHABITFIELDS[1] (Windows CE only).


Sample Length: 4 4 4 4
Channel Membership: Alpha Red Green Blue
Bit Number: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RGBAX
Sample Length Notation:
R. G. B. A. X
4.4.4.4.0

Diag.3 - The pixel format with an Alpha channel in a 16 bit pixel (in RGBAX Sample Length notation) actually generated by Adobe Photohop[8]

The table below lists all of the possible pixel formats of a DIB (in RGBAX notation).



DIB Header Versions		BITMAPCOREHEADER	BITMAPV2INFOHEADER BITMAPINFOHEADER	BITMAPV2INFOHEADER BITMAPINFOHEADER	BITMAPINFOHEADER
Compression→		N/A	BI_RGB	BI_BITFIELDS	BI_ALPHABITFIELDS
BitCount (bpp)	1	2* 8.8.8.0.0	2* 8.8.8.0.8		
	4	16* 8.8.8.0.0	16* 8.8.8.0.8	Illegal	Illegal
	8	256* 8.8.8.0.0	256* 8.8.8.0.8		
	16	Illegal	5.5.5.0.1	[0-16].[0-16].[0-16].0.[0-16]	[0-16].[0-16].[0-16].[0-16].[0-16
	24	8.8.8.0.0	8.8.8.0.0	Illegal	Illegal
	32	Illegal	8.8.8.0.8	[0-32].[0-32].[0-32].0.[0-32]	[0-32].[0-32].[0-32].[0-32].[0-3

Legend: The notation **[n-m]** indicates the range of integer values from n to m (inclusive) and pertains to the n
The notation 2* or 16* or 256* denotes the maximum count of distinct colors in a palletized bitmap.

The BITFIELD mechanism described above allows for the definition of tens of thousands different pixel formats, however only several of them are used in practice[8], such as:

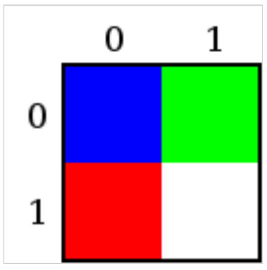
- 8.8.8.0.0
- 8.8.8.0.8
- 8.8.8.8.0
- 5.5.5.0.1
- 5.5.5.1.0
- 5.6.5.0.0
- 4.4.4.0.4
- 4.4.4.4.0
- All palettized formats (marked in yellow in table above)

[The list above is in the RGBAX notation]

Example of a 2×2 Pixel, 24-Bit Bitmap (Windows DIB Header BITMAPINFOHEADER)

Pixel format: 8.8.8.0.0 (in RGBAX notation).

Offset	Size	Hex Value	Value	Description
BMP Header				
0h	2	42 4D	"BM"	Magic Number (unsigned integer 66, 77)
2h	4	46 00 00 00	70 Bytes	Size of the BMP file
6h	2	00 00	Unused	Application Specific
8h	2	00 00	Unused	Application Specific
Ah	4	36 00 00 00	54 bytes	The offset where the Pixel Array (bitmap data) can be found.
DIB Header				
Eh	4	28 00 00 00	40 bytes	The number of bytes in the DIB header (from this point).
12h	4	02 00 00 00	2 pixels	The width of the bitmap in pixels
16h	4	02 00 00 00	2 pixels	The height of the bitmap in pixels
1Ah	2	01 00	1 plane	Number of color planes being used.
1Ch	2	18 00	24 bits	The number of bits per pixel.
1Eh	4	00 00 00 00	0	BI_RGB, no Pixel Array compression used
22h	4	10 00 00 00	16 bytes	The size of the raw data in the Pixel Array (incl. padding)
26h	4	13 0B 00 00	2,835 pixels/meter	The horizontal resolution of the image
2Ah	4	13 0B 00 00	2,835 pixels/meter	The vertical resolution of the image
2Eh	4	00 00 00 00	0 colors	Number of colors in the palette
32h	4	00 00 00 00	0 important colors	Means all colors are important

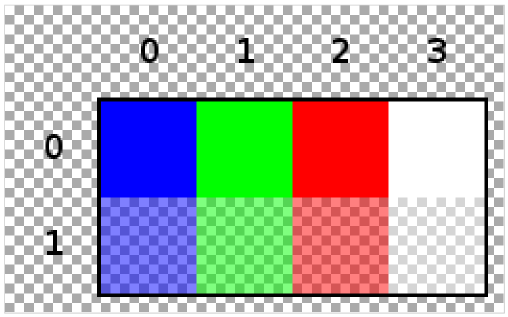


Example of a 2×2 Pixel Bitmap, with 24 bits/pixel encoding

Start of Pixel Array (bitmap data)				
36h	3	`00 00 FF`	0 0 255	Red, Pixel (0,1)
39h	3	`FF FF FF`	255 255 255	White, Pixel (1,1)
3Ch	2	`00 00`	0 0	Padding for 4 byte alignment (Could be a value other than zero)
3Eh	3	`FF 00 00`	255 0 0	Blue, Pixel (0,0)
41h	3	`00 FF 00`	0 255 0	Green, Pixel (1,0)
44h	2	`00 00`	0 0	Padding for 4 byte alignment (Could be a value other than zero)

Example of a 4×2 Pixel, 32bpp Bitmap with opacity values in the Alpha channel (Windows DIB Header BITMAPV4HEADER)

Pixel format: 8.8.8.8.0 (in RGBAX notation).



Example of a 4×2 Pixel Bitmap, with 32 bits/pixel encoding

Offset	Size	Hex Value	Value	Description
BMP Header				
0h	2	`42 4D`	"BM"	Magic Number (unsigned integer 66, 77)
2h	4	`9A 00 00 00`	154 Bytes	Size of the BMP file
6h	2	`00 00`	Unused	Application Specific
8h	2	`00 00`	Unused	Application Specific
Ah	4	`7A 00 00 00`	122 bytes from the start of the file	The offset where the Pixel Array (bitmap data) can be found.
DIB Header				
Eh	4	`6C 00 00 00`	108 bytes	The number of bytes in the DIB header (from this point).
12h	4	`04 00 00 00`	4 pixels	The width of the bitmap in pixels
16h	4	`02 00 00 00`	2 pixels	The height of the bitmap in pixels

1Ah	2	`01 00`	1 plane	Number of color planes being used.
1Ch	2	`20 00`	32 bits	The number of bits per pixel.
1Eh	4	`03 00 00 00`	3	BI_BITFIELDS, no Pixel Array compression used
22h	4	`20 00 00 00`	32 bytes	The size of the raw data in the Pixel Array (incl. padding)
26h	4	`13 0B 00 00`	2,835 pixels/meter	The horizontal physical resolution of the image
2Ah	4	`13 0B 00 00`	2,835 pixels/meter	The vertical physical resolution of the image
2Eh	4	`00 00 00 00`	0 colors	Number of colors in the palette
32h	4	`00 00 00 00`	0 important colors	0 means all colors are important
36h	4	`00 00 FF 00`	00FF0000 in big-endian	Red channel bit mask (valid because BI_BITFIELDS is specified)
3Ah	4	`00 FF 00 00`	0000FF00 in big-endian	Green channel bit mask (valid because BI_BITFIELDS is specified)
3Eh	4	`FF 00 00 00`	000000FF in big-endian	Blue channel bit mask (valid because BI_BITFIELDS is specified)
42h	4	`00 00 00 FF`	FF000000 in big-endian	Alpha channel bit mask
46h	4	`20 6E 69 57`	LCS_WINDOWS_COLOR_SPACE	Type of Color Space
4Ah	24h	`24* 00...00`	CIEXYZTRIPLE Color Space endpoints	Unused when LCS_WINDOWS_COLOR_SPACE is specified
6Eh	4	`00 00 00 00`	0 Red Gamma	Unused when LCS_WINDOWS_COLOR_SPACE is specified
72h	4	`00 00 00 00`	0 Green Gamma	Unused when LCS_WINDOWS_COLOR_SPACE is specified
76h	4	`00 00 00 00`	0 Blue Gamma	Unused when LCS_WINDOWS_COLOR_SPACE is specified
		Start of the Pixel Array (the bitmap Data)		
7Ah	4	`FF 00 00 7F`	255 0 0 127	Blue (Alpha: 127), Pixel (0,1)
7Eh	4	`00 FF 00 7F`	0 255 0 127	Green (Alpha: 127), Pixel (1,1)
82h	4	`00 00 FF 7F`	0 0 255 127	Red (Alpha: 127), Pixel (2,1)
86h	4	`FF FF FF 7F`	255 255 255 127	White (Alpha: 127), Pixel (3,1)
8Ah	4	`FF 00 00 FF`	255 0 0 255	Blue (Alpha: 255), Pixel (0,0)
8Eh	4	`00 FF 00 FF`	0 255 0 255	Green (Alpha: 255), Pixel (1,0)
92h	4	`00 00 FF FF`	0 0 255 255	Red (Alpha: 255), Pixel (2,0)

96h	4	`FF FF FF FF`	255 255 255 255	White (Alpha: 255), Pixel (3,0)

Note that the bitmap data starts with the lower left hand corner of the image.

Usage of BMP format

The simplicity of the BMP file format, and its widespread familiarity in Windows and elsewhere, as well as the fact that this format is relatively well documented and free of patents, makes it a very common format that image processing programs from many operating systems can read and write.

Many older graphical user interfaces used bitmaps in their built-in graphics subsystems;[9] for example, the Microsoft Windows and OS/2 platforms' GDI subsystem, where the specific format used is the *Windows and OS/2 bitmap file format*, usually named with the file extension of `.BMP` or `.DIB`.

While most BMP files have a relatively large file size due to lack of any compression (or generally low-ratio RLE on palletized images), many BMP files can be considerably compressed with lossless data compression algorithms such as ZIP (in extreme cases of non-photographic data, up to 0.1% of original size) because they contain redundant data. Some formats, such as RAR, even include routines specifically targeted at efficient compression of such data.

Related formats

Main article: Image file formats

The X Window System uses a similar XBM format for black-and-white images, and XPM (*pixelmap*) for color images. There are also a variety of "raw" formats, which saves raw data with no other information. The Portable Pixmap (PPM) and Truevision TGA formats also exist, but are less often used – or only for special purposes; for example, TGA can contain transparency information.

Numerous other bitmap file formats are in existence, though most are not widely used.[10]

See also

- Comparison of graphics file formats

References

1. ^ ***a b c*** MSDN - BITMAPINFOHEADER: BI_ALPHABITFIELDS in biCompression member (http://msdn.microsoft.com/en-us/library/aa452885.aspx)
2. ^ ***a b c d e f g h i j k*** "DIBs and Their Uses" (http://support.microsoft.com/kb/q81498/) . *Microsoft Help and Support*. 2005-02-11. http://support.microsoft.com/kb/q81498/.
3. ^ "JPEG and PNG Extensions for Specific Bitmap Functions and Structures" (http://msdn.microsoft.com/en-us/library/dd145023(VS.85).aspx) . http://msdn.microsoft.com/en-us/library/dd145023(VS.85).aspx.
4. ^ ***a b*** "GFF Format Summary: OS/2 Bitmap" (http://netghost.narod.ru/gff/graphics/summary/os2bmp.htm) . http://netghost.narod.ru/gff/graphics/summary/os2bmp.htm.
5. ^ MSDN - BITMAPV4HEADER: The member bV4AlphaMask (http://msdn.microsoft.com/en-us/library/dd183380%28VS.85%29.aspx)
6. ^ MSDN - RGBQUAD: rgbReserved member (http://msdn.microsoft.com/en-us/library/dd162938%28VS.85%29.aspx)
7. ^ MSDN - BITMAPINFOHEADER: The member biBitCount (http://msdn.microsoft.com/en-us/library/dd183376%28VS.85%29.aspx)
8. ^ ***a b*** Adobe Photohop: http://livedocs.adobe.com/en_US/Photoshop

/10.0/WSfd1234e1c4b69f30ea53e41001031ab64-7751.html BMP Format

9. ^ Julian Smart, Stefan Csomor, and Kevin Hock (2006). *Cross-Platform GUI Programming with Wxwidgets* (http://books.google.com/books?id=CyMsvtgnq0QC&pg=PA265&dq=bitmap+pixmap+gui&as_brr=3&ei=4SjwRrTpHYSipgL63NS3BA&sig=4_ev_R-Xs8tXCVONCaiJEnFLtI0) . Prentice Hall. ISBN 0131473816. http://books.google.com/books?id=CyMsvtgnq0QC&pg=PA265&dq=bitmap+pixmap+gui&as_brr=3&ei=4SjwRrTpHYSipgL63NS3BA&sig=4_ev_R-Xs8tXCVONCaiJEnFLtI0.
10. ^ "List of bitmap file types" (http://www.file-extensions.org/filetype/extensions/name/Bitmap+image/) . *Search File-Extensions.org*. http://www.file-extensions.org/filetype/extensions/name/Bitmap+image/.

External links

- For a table view of the bitmap file format (http://atlc.sourceforge.net/bmp.html#_toc381201084) at sourceforge
- Bitmap File Structure (http://www.digicamsoft.com/bmp/bmp.html)
- An introduction to DIBs (Device Independent Bitmaps) (http://www.herdsoft.com/ti/davincie/imex3j8i.htm)
- BMP test images (http://wvnvaxa.wvnet.edu/vmswww/bmp.html)
- Simple bitmap loader C++ class (http://www.kalytta.com/bitmap.h)

Retrieved from "http://en.wikipedia.org/wiki/BMP_file_format"
Categories: Graphics file formats | Microsoft Windows multimedia technology