introduction

this document describes the microsoft windows and ibm os/2 picture bitmap files, called bitmaps or bmp files. most of the descriptions of the bmp file concentrate on the microsoft windows bmp structures like `bmpinfoheader` and `bmpcoreinfo` , but only a few describe the file contents on byte level. this information is therefor only intended to be used in applications where direct reading and writing of a bmp file is required.

bitmap file format

the following chapters contain the detailed information on the contents of the bmp file. first more general information will be given regarding the byte order and file alignment. the second chapter will concentrate on the byte-level contents of a bmp file. the third chapter will elaborate on this chapter and explain some of the concepts - like compression - and/or values in detail.

general

the bmp file has been created by microsoft and ibm and is therefor very strictly bound to the architecture of the main hardware platform that both companies support: the ibm compatible pc. this means that all values stored in the bmp file are in the intel format, sometimes also called the little endian format because of the byte order that an intel processor uses internally to store values.

the bmp files are the way, windows stores bit mapped images. the bmp image data is bit packed but every line must end on a dword boundary - if that 担 not the case, it must be padded with zeroes. bmp files are stored bottom-up, that means that the first scan line is the bottom line.

the bmp format has four incarnations, two under windows (new and old) and two under os/2, all are described here.

bmp contents

the following table contains a description of the contents of the bmp file. for every field, the file offset, the length and the contents will be given. for a more detailed discussion, see the following chapters.

offset	field	size	contents
0000h	identifier	2 bytes	the characters identifying the bitmap. the following entries are possible: 態 m’ - windows 3.1x, 95, nt, ⋯ 態 a’ - os/2 bitmap array 慍 i’ - os/2 color icon 慍 p’ - os/2 color pointer 慨 c’ - os/2 icon 憧 t’ - os/2 pointer
0002h	file size	1 dword	complete file size in bytes.
0006h	reserved	1 dword	reserved for later use.
000ah	bitmap data offset	1 dword	offset from beginning of file to the beginning of the bitmap data.
000eh	bitmap header size	1 dword	length of the bitmap info header used to describe the bitmap colors, compression, ⋯ the following sizes are possible: 28h - windows 3.1x, 95, nt, ⋯ 0ch - os/2 1.x f0h - os/2 2.x
0012h	width	1 dword	horizontal width of bitmap in pixels.
0016h	height	1 dword	vertical height of bitmap in pixels.
001ah	planes	1 word	number of planes in this bitmap.
001ch	bits per pixel	1 word	bits per pixel used to store palette entry information. this also identifies in an indirect way the number of possible colors. possible values are: 1 - monochrome bitmap

			4 - 16 color bitmap 8 - 256 color bitmap 16 - 16bit (high color) bitmap 24 - 24bit (true color) bitmap 32 - 32bit (true color) bitmap
001eh	compression	1 dword	compression specifications. the following values are possible: 0 - none (also identified by bi_rgb) 1 - rle 8-bit / pixel (also identified by bi_rle4) 2 - rle 4-bit / pixel (also identified by bi_rle8) 3 - bitfields (also identified by bi_bitfields)
0022h	bitmap data size	1 dword	size of the bitmap data in bytes. this number must be rounded to the next 4 byte boundary.
0026h	hresolution	1 dword	horizontal resolution expressed in pixel per meter.
002ah	vresolution	1 dword	vertical resolution expressed in pixels per meter.
002eh	colors	1 dword	number of colors used by this bitmap. for a 8-bit / pixel bitmap this will be 100h or 256.
0032h	important colors	1 dword	number of important colors. this number will be equal to the number of colors when every color is important.
0036h	palette	n * 4 byte	the palette specification. for every entry in the palette four bytes are used to describe the rgb values of the color in the following way: 1 byte for blue component 1 byte for green component 1 byte for red component

			1 byte filler which is set to 0 (zero)
0436h	bitmap data	x bytes	depending on the compression specifications, this field contains all the bitmap data bytes which represent indices in the color palette.

note: *the following sizes were used in the specification above:*

size	# bytes	sign
char	1	signed
word	2	unsigned
dword	4	unsigned

field details

some of the fields require some more information. the following chapters will try to provide this information:

height field

the *height* field identifies the height of the bitmap in pixels. in other words, it describes the number of scan lines of the bitmap. if this field is negative, indicating a top-down dib, the *compression* field must be either bi_rgb or bi_bitfields. top-down dibs cannot be compressed.

bits per pixel field

the *bits per pixel* (bbp) field of the bitmap file determines the number of bits that define each pixel and the maximum number of colors in the bitmap.

- **when this field is equal to 1.**

  the bitmap is monochrome, and the palette contains two entries. each bit in the bitmap array represents a pixel. if the bit is clear, the pixel is displayed with the color of the first entry in the

palette; if the bit is set, the pixel has the color of the second entry in the table.

- **when this field is equal to 4.**

  the bitmap has a maximum of 16 colors, and the palette contains up to 16 entries. each pixel in the bitmap is represented by a 4-bit index into the palette. for example, if the first byte in the bitmap is 1fh, the byte represents two pixels. the first pixel contains the color in the second palette entry, and the second pixel contains the color in the sixteenth palette entry.

- **when this field is equal to 8.**

  the bitmap has a maximum of 256 colors, and the palette contains up to 256 entries. in this case, each byte in the array represents a single pixel.

- **when this field is equal to 16.**

  the bitmap has a maximum of 2^16 colors. if the *compression* field of the bitmap file is set to bi_rgb, the *palette* field does not contain any entries. each word in the bitmap array represents a single pixel. the relative intensities of red, green, and blue are represented with 5 bits for each color component. the value for blue is in the least significant 5 bits, followed by 5 bits each for green and red, respectively. the most significant bit is not used.

  if the *compression* field of the bitmap file is set to bi_bitfields, the *palette* field contains three dword color masks that specify the red, green, and blue components, respectively, of each pixel. each word in the bitmap array represents a single pixel.

  windows nt specific: when the *compression* field is set to bi_bitfields, bits set in each dword mask must be contiguous and should not overlap the bits of another mask. all the bits in the pixel do not have to be used.

windows 95 specific: when the *compression* field is set to bi_bitfields, windows 95 supports only the following 16bpp color masks: a 5-5-5 16-bit image, where the blue mask is 0x001f, the green mask is 0x03e0, and the red mask is 0x7c00; and a 5-6-5 16-bit image, where the blue mask is 0x001f, the green mask is 0x07e0, and the red mask is 0xf800.

- **when this field is equal to 24.**

  the bitmap has a maximum of 2^24 colors, and the *palette* field does not contain any entries. each 3-byte triplet in the bitmap array represents the relative intensities of blue, green, and red, respectively, for a pixel.

- **when this field is equal to 32.**

  the bitmap has a maximum of 2^32 colors. if the *compression* field of the bitmap is set to bi_rgb, the *palette* field does not contain any entries. each dword in the bitmap array represents the relative intensities of blue, green, and red, respectively, for a pixel. the high byte in each dword is not used.

  if the *compression* field of the bitmap is set to bi_bitfields, the palette field contains three dword color masks that specify the red, green, and blue components, respectively, of each pixel. each dword in the bitmap array represents a single pixel.

  windows nt specific: when the *compression* field is set to bi_bitfields, bits set in each dword mask must be contiguous and should not overlap the bits of another mask. all the bits in the pixel do not have to be used.

  windows 95 specific: when the *compression* field is set to bi_bitfields, windows 95 supports only the following 32bpp color mask: the blue mask is 0x000000ff, the green mask is 0x0000ff00, and the red mask is 0x00ff0000.

compression field

the *compression* field specifies the way the bitmap data is stored in the file. this information together with the *bits per pixel* (*bpp*) field identifies the compression algorithm to follow.

the following values are possible in this field:

value	meaning
bi_rgb	an uncompressed format.
bi_rle4	an rle format for bitmaps with 4 bits per pixel. the compression format is a two-byte format consisting of a count byte followed by two word-length color indices. for more information, see the following remarks section.
bi_rle8	a run-length encoded (rle) format for bitmaps with 8 bits per pixel. the compression format is a two-byte format consisting of a count byte followed by a byte containing a color index. for more information, see the following remarks section.
bi_bitfields	specifies that the bitmap is not compressed and that the color table consists of three double word color masks that specify the red, green, and blue components, respectively, of each pixel. this is valid when used with 16- and 32- bits-per-pixel bitmaps.

when the compression field is bi_rle8, the bitmap is compressed by using a run-length encoding (rle) format for an 8-bit bitmap. this format can be compressed in encoded or absolute modes. both modes can occur anywhere in the same bitmap.

- **encoded mode consists of two bytes:**

  the first byte specifies the number of consecutive pixels to be drawn using the color index contained in the second byte. in addition, the first byte of the pair can be set to zero to indicate an escape that denotes an end of line, end of bitmap, or delta. the interpretation of the escape depends on the value of the second byte of the pair, which can be one of the following:

0 end of line.

1 end of bitmap.

2 delta. the two bytes following the escape contain unsigned values

indicating the horizontal and vertical offsets of the next pixel from the current position.

- **absolute mode.**

  the first byte is zero and the second byte is a value in the range 03h through ffh. the second byte represents the number of bytes that follow, each of which contains the color index of a single pixel. when the second byte is 2 or less, the escape has the same meaning as in encoded mode. in absolute mode, each run must be aligned on a word boundary.

the following example shows the hexadecimal values of an 8-bit compressed bitmap.

```
03 04 05 06 00 03 45 56 67 00 02 78 00 02 05 01 02 78 00 00
09 1e 00 01
```

this bitmap would expand as follows (two-digit values represent a color index for a single pixel):

```
04 04 04

06 06 06 06 06

45 56 67

78 78

move current position 5 right and 1 down

78 78

end of line

1e 1e 1e 1e 1e 1e 1e 1e 1e

end of rle bitmap
```

when the compression field is bi_rle4, the bitmap is compressed by using a run-length encoding format for a 4-bit bitmap, which also uses encoded and absolute modes:

- **in encoded mode.**

  the first byte of the pair contains the number of pixels to be drawn using the color indices in the second byte. the second byte contains two color indices, one in its high-order four bits and one in its low-order four bits. the first of the pixels is drawn using the color specified by the high-order four bits, the second is drawn using the color in the low-order four bits, the third is drawn using the color in the high-order four bits, and so on, until all the pixels specified by the first byte have been drawn.

- **in absolute mode.**

  the first byte is zero, the second byte contains the number of color indices that follow, and subsequent bytes contain color indices in their high- and low-order four bits, one color index for each pixel. in absolute mode, each run must be aligned on a word boundary.

  the end-of-line, end-of-bitmap, and delta escapes described for bi_rle8 also apply to bi_rle4 compression.

the following example shows the hexadecimal values of a 4-bit compressed bitmap.

```
03 04 05 06 00 06 45 56 67 00 04 78 00 02 05 01 04 78 00 00
09 1e 00 01
```

this bitmap would expand as follows (single-digit values represent a color index for a single pixel):

```
0 4 0

0 6 0 6 0

4 5 5 6 6 7

7 8 7 8

move current position 5 right and 1 down
```

```
7 8 7 8

end of line

1 e 1 e 1 e 1 e 1

end of rle bitmap
```

colors field

the *colors* field specifies the number of color indices in the color table that are actually used by the bitmap. if this value is zero, the bitmap uses the maximum number of colors corresponding to the value of the *bbp* field for the compression mode specified by the *compression* field.

if the *colors* field is nonzero and the *bbp* field less than 16, the *colors* field specifies the actual number of colors the graphics engine or device driver accesses.

if the *bbp* field is 16 or greater, then *colors* field specifies the size of the color table used to optimize performance of windows color palettes.

if *bbp* equals 16 or 32, the optimal color palette starts immediately following the three double word masks.

if the bitmap is a packed bitmap (a bitmap in which the bitmap array immediately follows the bitmap header and which is referenced by a single pointer), the *colors* field must be either 0 or the actual size of the color table.

important colors field

the *important colors* field specifies the number of color indices that are considered important for displaying the bitmap. if this value is zero, all colors are important.