Šaltinis

<http://www.ece.ualberta.ca/~elliott/ee552/studentAppNotes/2003_w/misc/bmp_file_format/bmp_file_format.htm>

**THE BMP FILE FORMAT**

Compiled by Nathan Liesch of **Imperium Accelero 9000**

Increasingly the power of FPGAs is being utilized for DSP applications and a common source for digital signals to process is images. The first step in implementing any sort of image processing algorithm is accessing the raw pixel data.

The MS-Windows standard format is BMP and was developed as a device-independent bitmap (DIB) format that will allow Windows to display the bitmap on any type of display device. The term "device independent" means that the bitmap specifies pixel color in a form independent of the method used by a display to represent color. This file format can be stored uncompressed, so reading BMP files is fairly simple; most other graphics formats are compressed, and some, like GIF, are difficult to decompress.

The file format consists of the following structures:

|  |  |  |
| --- | --- | --- |
| **Structure** | **Corresponding Bytes** | **Description** |
| Header | 0x00 - 0x0D | contains information about the type, size, and layout of a device-independent bitmap file |
| InfoHeader | 0x0E - 0x35 | specifies the dimensions, compression type, and color format for the bitmap |
| ColorTable | 0x36 - variable | contains as many elements as there are colors in the bitmap, but is not present for bitmaps with 24 color bits because each pixel is represented by 24-bit red-green-blue (RGB) values in the actual bitmap data area |
| Pixel Data | variable | an array of bytes that defines the bitmap bits. These are the actual image data, represented by consecutive rows, or "scan lines," of the bitmap. Each scan line consists of consecutive bytes representing the pixels in the scan line, in left-to-right order. The system maps pixels beginning with the bottom scan line of the rectangular region and ending with the top scan line. |

Below is a more detailed table of the contents of each of these structures.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Size** | **Offset** | **Description** |
| **Header** | | | 14 bytes |  | Windows Structure: BITMAPFILEHEADER |
|  | Signature | | 2 bytes | 0000h | 'BM' |
| FileSize | | 4 bytes | 0002h | File size in bytes |
| reserved | | 4 bytes | 0006h | unused (=0) |
| DataOffset | | 4 bytes | 000Ah | Offset from beginning of file to the beginning of the bitmap data |
| **InfoHeader** | | | 40 bytes |  | Windows Structure: BITMAPINFOHEADER |
|  | Size | | 4 bytes | 000Eh | Size of InfoHeader =40 |
| Width | | 4 bytes | 0012h | Horizontal width of bitmap in pixels |
| Height | | 4 bytes | 0016h | Vertical height of bitmap in pixels |
| Planes | | 2 bytes | 001Ah | Number of Planes (=1) |
| Bits Per Pixel | | 2 bytes | 001Ch | 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 palette. NumColors = 1    4 = 4bit palletized. NumColors = 16    8 = 8bit palletized. NumColors = 256   16 = 16bit RGB. NumColors = 65536 24 = 24bit RGB. NumColors = 16M |
| Compression | | 4 bytes | 001Eh | Type of Compression    0 = BI\_RGB   no compression    1 = BI\_RLE8 8bit RLE encoding    2 = BI\_RLE4 4bit RLE encoding |
| ImageSize | | 4 bytes | 0022h | (compressed) Size of Image   It is valid to set this =0 if Compression = 0 |
| XpixelsPerM | | 4 bytes | 0026h | horizontal resolution: Pixels/meter |
| YpixelsPerM | | 4 bytes | 002Ah | vertical resolution: Pixels/meter |
| Colors Used | | 4 bytes | 002Eh | Number of actually used colors. For a 8-bit / pixel bitmap this will be 100h or 256. |
| Important Colors | | 4 bytes | 0032h | Number of important colors   0 = all |
| **ColorTable** | | | 4 \* NumColors bytes | 0036h | present only if Info.BitsPerPixel less than 8    colors should be ordered by importance |
|  |  | Red | 1 byte |  | Red intensity (reikia sukeisti raudoną su melyna!!!!!) |
| Green | 1 byte |  | Green intensity |
| Blue | 1 byte |  | Blue intensity |
| reserved | 1 byte |  | unused (=0) |
| repeated NumColors times | | | |  |
| **Pixel Data** | | | InfoHeader.ImageSize bytes |  | The image data |

Bits Per Pixel Field

|  |  |
| --- | --- |
| Value | Description |
| 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. |
| 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. |
| 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. |
| 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 4 byte color masks that specify the red, green, and blue components, respectively, of each pixel.  Each 2 bytes in the bitmap array represents a single pixel. |
| 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. |

Additional Info

Each scan line is zero padded to the nearest 4-byte boundary. If the image has a width that is not divisible by four, say, 21 bytes, there would be 3 bytes of padding at the end of every scan line.

Scan lines are stored bottom to top instead of top to bottom.

RGB values are stored backwards i.e. BGR.

4 bit & 8 bit BMPs can be compressed. BMPs use a very simple form of compression called Run Length Encoded (RLE). Instead of storing a value for each pixel RLE stores a number, N, followed by an index. This means that the next N pixels are of the color for this index.

For additional information refer to:  
[MSDN Library: Bitmap Storage](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdi/bitmaps_4v1h.asp)  
[The Graphics File Formats Page - BMP](http://www.dcs.ed.ac.uk/home/mxr/gfx/2d/BMP.txt)

Compiled by Nathan Liesch of **Imperium Accelero 9000**