BMP image format

Written by Paul Bourke
July 1998

Introduction

BMP files are an historic (but still commonly used) file format for the historic (but still commonly used) operating system called "Windows". BMP images can range from black and white (1 bit per pixel) up to 24 bit colour (16.7 million colours). While the images can be compressed, this is rarely used in practice and won't be discussed in detail here.

Structure

A BMP file consists of either 3 or 4 parts as shown in the diagram on the right. The first part is a header, this is followed by a information section, if the image is indexed colour then the palette follows, and last of all is the pixel data. The position of the image data with respect to the sart of the file is contained in the header. Information such as the image width and height, the type of compression, the number of colours is contained in the information header.

Header

The header consists of the following fields. Note that we are assuming short int of 2 bytes, int of 4 bytes, and long int of 8 bytes. Further we are assuming byte ordering as for typical (Intel) machines. The header is 14 bytes in length.

```
typedef struct {
   unsigned short int type;                /* Magic identifier            */
   unsigned int size;                      /* File size in bytes          */
   unsigned short int reserved1, reserved2;
   unsigned int offset;                    /* Offset to image data, bytes */
} HEADER;
```

The useful fields in this structure are the type field (should be 'BM') which is a simple check that this is likely to be a legitimate BMP file, and the offset field which gives the number of bytes before the actual pixel data (this is relative to the start of the file). Note that this struct is not a multiple of 4 bytes for those machines/compilers that might assume this, these machines will generally pad this struct by 2 bytes to 16 which will unalign the future fread() calls - be warned.

Information

The image info data that follows is 40 bytes in length, it is described in the struct given below. The fields of most interest below are the image width and height, the number of bits per pixel (should be 1, 4, 8 or 24), the number of planes (assumed to be 1 here), and the compression type (assumed to be 0 here).

```
typedef struct {
   unsigned int size;               /* Header size in bytes      */
   int width,height;                /* Width and height of image */
   unsigned short int planes;       /* Number of colour planes   */
   unsigned short int bits;         /* Bits per pixel            */
   unsigned int compression;        /* Compression type          */
   unsigned int imagesize;          /* Image size in bytes       */
   int xresolution,yresolution;     /* Pixels per meter          */
   unsigned int ncolours;           /* Number of colours         */
   unsigned int importantcolours;   /* Important colours         */
} INFOHEADER;
```

The compression types supported by BMP are listed below :

- 0 - no compression
- 1 - 8 bit run length encoding
- 2 - 4 bit run length encoding
- 3 - RGB bitmap with mask

Only type 0 (no compression will be discussed here.

24 bit Image Data

The simplest data to read is 24 bit true colour images. In this case the image data follows immediately after the information header, that is, there is no colour palette. It consists of three bytes per pixel in b,g,r order. Each byte gives the saturation for that colour component, 0 for black and 1 for white (fully saturated).

Indexed Colour Data

If the image is indexed colour then immediately after the information header there will be a table of infoheader.ncolours colours, each of 4 bytes. The first three bytes correspond to b,g,r components, the last byte is reserved/unused but could obviously represent the alpha channel. For 8 bit greyscale images this colour index will generally just be a greyscale ramp. If you do the sums....then the length of the header plus the length of the information block plus 4 times the number of palette colours should equal the image data offset. In other words

$$14 + 40 + 4 * \text{infoheader.ncolours} = \text{header.offset}$$

Source code

Here's source provided by Michael Sweet, BITMAP.H, BITMAP.C, and BMPVIEW.C.

And some example code by myself, parse.c. Note that neither of these code segments will handle all types of BMP files, in particular, they don't handle compressed BMP files. They should be a good starting point to variations encountered and to those who wish to write BMP compliant files. On the other hand if you have or write a better BMP handler then you are welcome to submit for addition here.

Contribution by Adam Majewski that writes a one bit per pixel BMP file: pf1bit_bmp.c.