

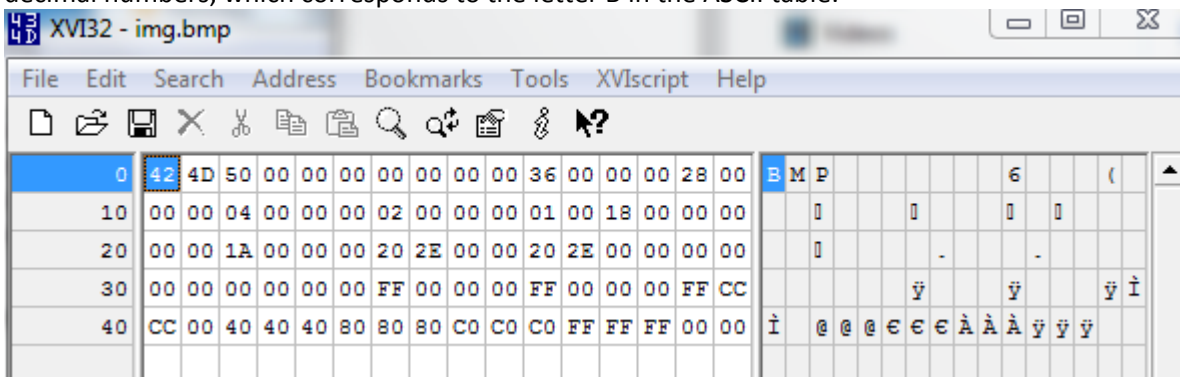
Binary Files - Background:

Data files are often stored in binary format. It is much more compact to store the number 200000 as a 32-bit integer instead of as 6 characters (each of which take 16-bits to store). It is also faster to read in those raw bits instead of reading strings and converting those to integers.

Binary files are not designed to be read by humans or text editors. If you open one in word or notepad you get a ton of garbage. You can view one in a HEX editor - a program that reads a binary file and displays each byte as a 2-digit hexadecimal (base 16) number. In hexadecimal each digit can be 0-9,a,b,c,d,e, or f (a=10, b=11,... f=15). Hexadecimal is used because it is much more compact than binary, but converts very easily to binary.

XVI32 is a free PC hex editor (similar programs are available for Mac/Linux). Use it to open the file **img.bmp**. Resize the window's width until it looks like the picture below.

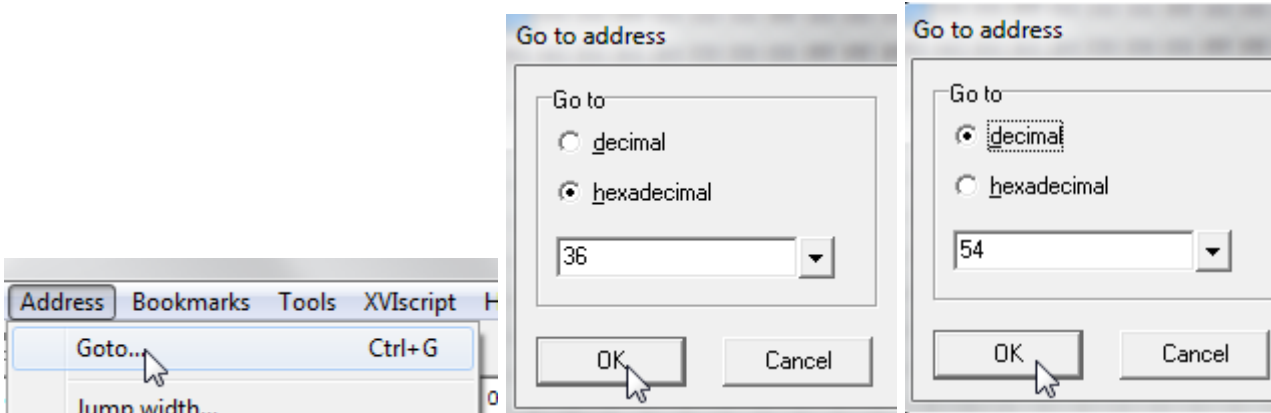
This is what the file **img.bmp** looks like opened in the hex editor. The first byte is 42 in hex (01000010 in binary) or 66 in decimal numbers, which corresponds to the letter B in the ASCII table:

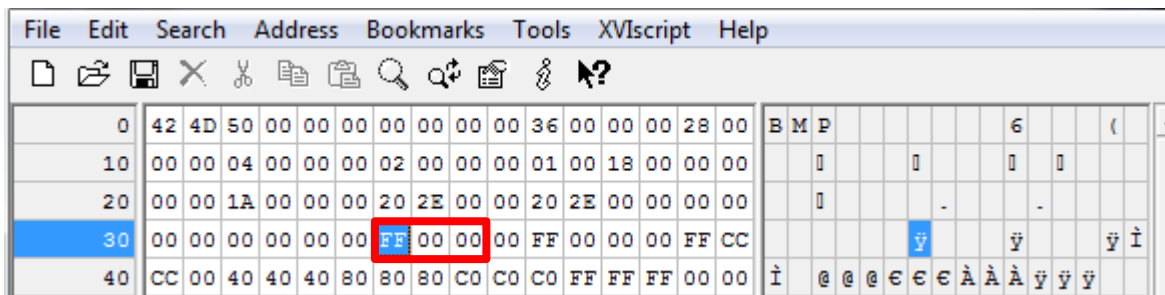


Each square is one byte - two hex digits representing a binary number between 0 and 255. In BMP files the first three bytes are the ASCII codes for B M P.

To read the rest of the binary files we need to know how to interpret the 1's and 0's. Are the digits 000000000000000000001110011000011011 to be interpreted as one 32-bit integer (58907 in decimal)? It is supposed to be two 16-bit characters (0000000000000000 and 1110011000011011)? Is it supposed to be four 8-bit bytes? (00000000, 00000000, 11100110, 00011011)

For details on how BMP files are formatted, see the appendix. The short version is that there are a bunch of values at the start that indicate things like the size and the encoding scheme. Then there is the image data. In the sample file, the image data starts at hex address 36 (or 54 in decimal). Use the Goto command and enter either 36 hex or 54 decimal:





This is the start of the image data. Each pixel is represented by 3 bytes (hex pairs):

BLUE : 1 byte	GREEN : 1 byte	RED : 1 byte
---------------	----------------	--------------

So the first pixel is the FF and the next two 00's:

BLUE : 1 byte	GREEN : 1 byte	RED : 1 byte
FF (255 decimal)	00 (0 decimal)	00 (0 decimal)

It is all blue, no red, no green.

If you open the image in either a web browser or image viewer and zoom in (it is only 4 pixels by 2 pixels), you will see it looks like this:



See the blue pixel? (Lower left-corner).

The next pixel starts right after the first:

0	42	4D	50	00	00	00	00	00	00	00	36	00	00	00	28	00
10	00	00	04	00	00	00	02	00	00	00	01	00	18	00	00	00
20	00	00	1A	00	00	00	20	2E	00	00	20	2E	00	00	00	00
30	00	00	00	00	00	00	FF	00	00	00	FF	00	00	00	FF	CC
40	CC	00	40	40	40	80	80	80	C0	C0	C0	FF	FF	FF	00	00

BLUE : 1 byte	GREEN : 1 byte	RED : 1 byte
00 (0 decimal)	FF (255 decimal)	00 (0 decimal)

This is the green pixel.

Each row is written from left to right, but the rows are in reverse order. The first row in the file is actually the bottom row in the image.

The top row, with gray pixels comes after the bottom row. To find gray pixels look for three identical values right after each other - the same amount of red/green/blue make a shade of gray. 40/40/40 is a dark gray, while C0/C0/C0 is a brighter shade. 00/00/00 would be black and FF/FF/FF is white.

0	42	4D	50	00	00	00	00	00	00	00	36	00	00	00	28	00
10	00	00	04	00	00	00	02	00	00	00	01	00	18	00	00	00
20	00	00	1A	00	00	00	20	2E	00	00	20	2E	00	00	00	00
30	00	00	00	00	00	00	FF	00	00	00	FF	00	00	00	FF	CC
40	CC	00	40	40	40	80	80	80	C0	C0	C0	FF	FF	FF	00	00

Making Changes

Identify the data for a pixel, click on a square and type in a new hex value. You always have to type two digits at a time (FF or 00 or 3C). Don't modify values from the beginning of the file before the image data starts, you will almost certainly break the file (if you do, just start with a new copy).

After making a change, hit save, then go reload the image in whatever program you are viewing it in.

Appendix –

BMP File Background:

File formats specify how a particular kind of file is to be written - how to interpret the 1's and 0's. Below is a simplified description of the file format for .bmp (bitmap) image files. (*The full format for .bmp image files can be found at: http://en.wikipedia.org/wiki/BMP_file_format)*

Byte Address	Description	Sample value from above screen shot
0	Magic Number - verifies really is bmp data - always 66 (42 hex)	42 hex (66 in decimal) - represents character 'B'
1	Magic Number - verifies really is bmp data - always 77 (4d hex)	4d hex (77 in decimal) - represents character 'M'
2	Magic Number - verifies really is bmp data - always 80 (50 hex)	50 hex (80 in decimal) - represents character 'P'
10	Byte address image data starts	10 is <i>a</i> in hex. So look at the 11 th column (first one is 0). The value is 36hex or 54 in decimal. The image data starts at byte #54.
18	Width of bitmap	18 is 12 in hex. So look in column '2' in the second row. The value is 04hex which is 4 in decimal. The image is 4 pixels wide.
22	Height of bitmap	22 is 16 in hex. So look in column '6' in the second row. The value is 02hex which is 2 in decimal. The image is 2 pixels tall.

Notes: in the full file format, there are other pieces of information that are stored at other addresses (we are ignoring the real messy details). Also, the width and height are stored as a 32-bit integers (not just as one byte). But there are some hairy issues with reading values more than one byte long. (Google "little endian")