Article • 08/23/20225 minutes to read

The **BITMAPINFOHEADER** structure contains information about the dimensions and color format of a device-independent bitmap (DIB).

> **Note** This structure is also described in the GDI documentation. However, the semantics for video data are slightly different than the semantics used for GDI. If you are using this structure to describe video data, use the information given here.

Syntax

C++

```
typedef struct tagBITMAPINFOHEADER {
  DWORD biSize;
  LONG  biWidth;
  LONG  biHeight;
  WORD  biPlanes;
  WORD  biBitCount;
  DWORD biCompression;
  DWORD biSizeImage;
  LONG  biXPelsPerMeter;
  LONG  biYPelsPerMeter;
  DWORD biClrUsed;
  DWORD biClrImportant;
} BITMAPINFOHEADER, *LPBITMAPINFOHEADER, *PBITMAPINFOHEADER;
```

Members

`biSize`

Specifies the number of bytes required by the structure. This value does not include the size of the color table or the size of the color masks, if they are appended to the end of structure. See Remarks.

`biWidth`

Specifies the width of the bitmap, in pixels. For information about calculating the stride of the bitmap, see Remarks.

`biHeight`

Specifies the height of the bitmap, in pixels.

- For uncompressed RGB bitmaps, if **biHeight** is positive, the bitmap is a bottom-up DIB with the origin at the lower left corner. If **biHeight** is negative, the bitmap is a top-down DIB with the origin at the upper left corner.
- For YUV bitmaps, the bitmap is always top-down, regardless of the sign of **biHeight**. Decoders should offer YUV formats with positive **biHeight**, but for backward compatibility they should accept YUV formats with either positive or negative **biHeight**.
- For compressed formats, **biHeight** must be positive, regardless of image orientation.

`biPlanes`

Specifies the number of planes for the target device. This value must be set to 1.

`biBitCount`

Specifies the number of bits per pixel (bpp). For uncompressed formats, this value is the average number of bits per pixel. For compressed formats, this value is the implied bit depth of the uncompressed image, after the image has been decoded.

`biCompression`

For compressed video and YUV formats, this member is a FOURCC code, specified as a **DWORD** in little-endian order. For example, YUYV video has the FOURCC 'VYUY' or 0x56595559. For more information, see FOURCC Codes.

For uncompressed RGB formats, the following values are possible:

Value	Meaning
BI_RGB	Uncompressed RGB.
BI_BITFIELDS	Uncompressed RGB with color masks. Valid for 16-bpp and 32-bpp bitmaps.

See Remarks for more information. Note that **BI_JPG** and **BI_PNG** are not valid video formats.

For 16-bpp bitmaps, if **biCompression** equals **BI_RGB**, the format is always RGB 555. If **biCompression** equals **BI_BITFIELDS**, the format is either RGB 555 or RGB 565. Use the subtype GUID in the AM_MEDIA_TYPE structure to determine the specific RGB type.

`biSizeImage`

Specifies the size, in bytes, of the image. This can be set to 0 for uncompressed RGB bitmaps.

`biXPelsPerMeter`

Specifies the horizontal resolution, in pixels per meter, of the target device for the bitmap.

`biYPelsPerMeter`

Specifies the vertical resolution, in pixels per meter, of the target device for the bitmap.

`biClrUsed`

Specifies the number of color indices in the color table that are actually used by the bitmap. See Remarks for more information.

`biClrImportant`

Specifies the number of color indices that are considered important for displaying the bitmap. If this value is zero, all colors are important.

Remarks

Color Tables

The **BITMAPINFOHEADER** structure may be followed by an array of palette entries or color masks. The rules depend on the value of **biCompression**.

- If **biCompression** equals **BI_RGB** and the bitmap uses 8 bpp or less, the bitmap has a color table immediately following the **BITMAPINFOHEADER** structure. The color table consists of an array of **RGBQUAD** values. The size of the array is given by the **biClrUsed** member. If **biClrUsed** is zero, the array contains the maximum number of colors for the given bitdepth; that is, 2^**biBitCount** colors.
- If **biCompression** equals **BI_BITFIELDS**, the bitmap uses three **DWORD** color masks (red, green, and blue, respectively), which specify the byte layout of the pixels. The 1 bits in each mask indicate the bits for that color within the pixel.

- If **biCompression** is a video FOURCC, the presence of a color table is implied by the video format. You should not assume that a color table exists when the bit depth is 8 bpp or less. However, some legacy components might assume that a color table is present. Therefore, if you are allocating a **BITMAPINFOHEADER**

structure, it is recommended to allocate space for a color table when the bit depth is 8 bpp or less, even if the color table is not used.

When the **BITMAPINFOHEADER** is followed by a color table or a set of color masks, you can use the BITMAPINFO structure to reference the color table of the color masks. The **BITMAPINFO** structure is defined as follows:

syntax

```
typedef struct tagBITMAPINFO {
    BITMAPINFOHEADER bmiHeader;
    RGBQUAD          bmiColors[1];
} BITMAPINFO;
```

If you cast the **BITMAPINFOHEADER** to a BITMAPINFO, the **bmiHeader** member refers to the **BITMAPINFOHEADER** and the **bmiColors** member refers to the first entry in the color table, or the first color mask.

Be aware that if the bitmap uses a color table or color masks, then the size of the entire format structure (the **BITMAPINFOHEADER** plus the color information) is not equal to `sizeof(BITMAPINFOHEADER)` or `sizeof(BITMAPINFO)`. You must calculate the actual size for each instance.

Calculating Surface Stride

In an uncompressed bitmap, the stride is the number of bytes needed to go from the start of one row of pixels to the start of the next row. The image format defines a minimum stride for an image. In addition, the graphics hardware might require a larger stride for the surface that contains the image.

For uncompressed RGB formats, the minimum stride is always the image width in bytes, rounded up to the nearest **DWORD**. You can use the following formula to calculate the stride:

```
stride = ((((biWidth * biBitCount) + 31) & ~31) >> 3)
```

For YUV formats, there is no general rule for calculating the minimum stride. You must understand the rules for the particular YUV format. For a description of the most common YUV formats, see Recommended 8-Bit YUV Formats for Video Rendering. Decoders and video sources should propose formats where biWidth is the width of the image in pixels. If the video renderer requires a surface stride that is larger than the default image stride, it modifies the proposed media type by setting the following values:

- It sets **biWidth** equal to the surface stride in pixels.
- It sets the **rcTarget** member of the VIDEOINFOHEADER or VIDEOINFOHEADER2 structure equal to the image width, in pixels.

Then the video renderer proposes the modified format by calling IPin::QueryAccept on the upstream pin. For more information about this mechanism, see Dynamic Format Changes.

If there is padding in the image buffer, never dereference a pointer into the memory that has been reserved for the padding. If the image buffer has been allocated in video memory, the padding might not be readable memory.

Requirements

Header	wingdi.h

See also

DirectShow Structures

VIDEOINFOHEADER Structure

VIDEOINFOHEADER2 Structure

Working with Video Frames