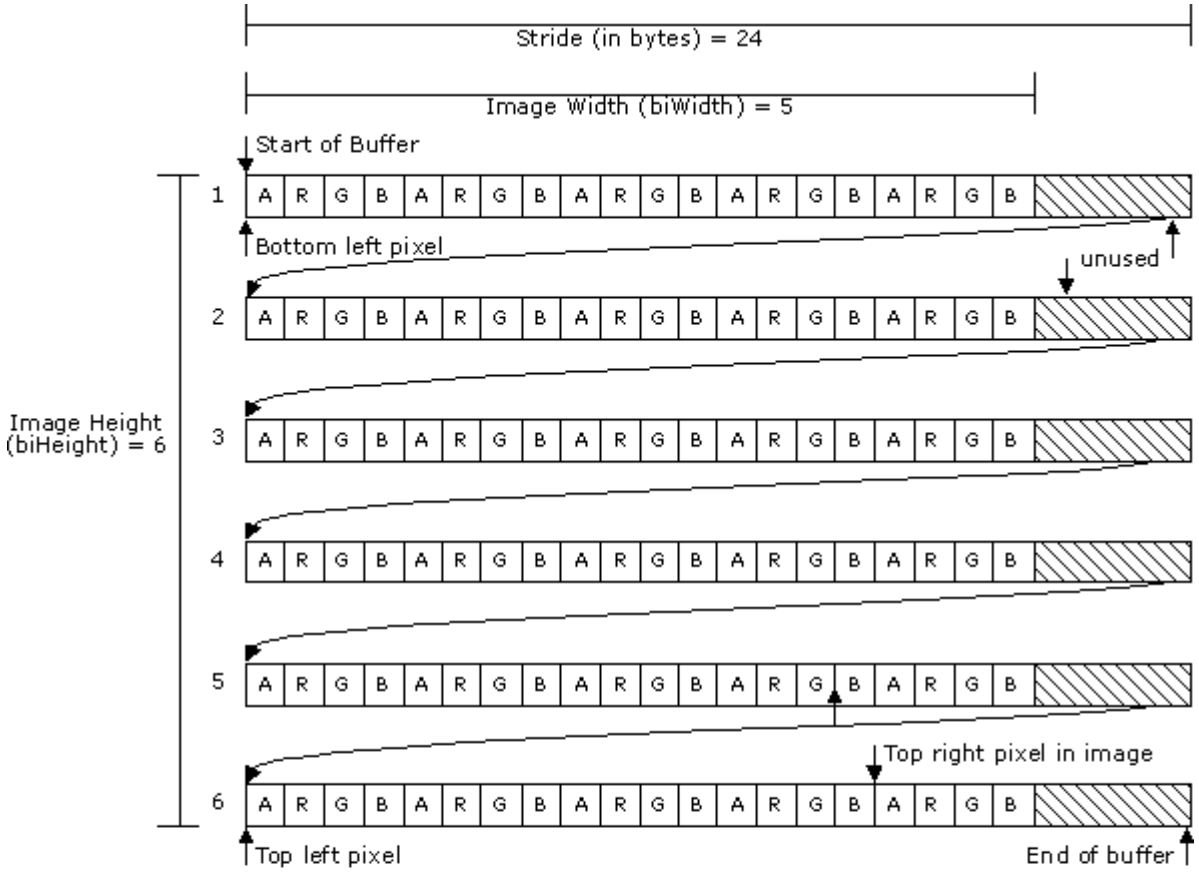
Top-Down vs. Bottom-Up DIBs

Microsoft DirectShow 9.0

Top-Down vs. Bottom-Up DIBs

If you are new to graphics programming, you might expect that a bitmap would be arranged in memory so that the top row of the image appeared at the start of the buffer, followed by the next row, and so forth. However, this is not necessarily the case. In Windows, device-independent bitmaps (DIBs) can be placed in memory in two different orientations, bottom-up and top-down.

In a *bottom-up* DIB, the image buffer starts with the *bottom* row of pixels, followed by the next row up, and so forth. The top row of the image is the last row in the buffer. Therefore, the first byte in memory is the *bottom-left* pixel of the image. In GDI, all DIBs are bottom-up. The following diagram shows the physical layout of a bottom-up DIB.



Physical layouts of bytes in a 5x6 bottom-up ARGB bitmap
Note: the actual number of unused alignment bytes at the end of each row will vary depending on the graphics hardware and/or other factors. In some cases this number may be zero.

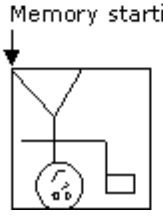
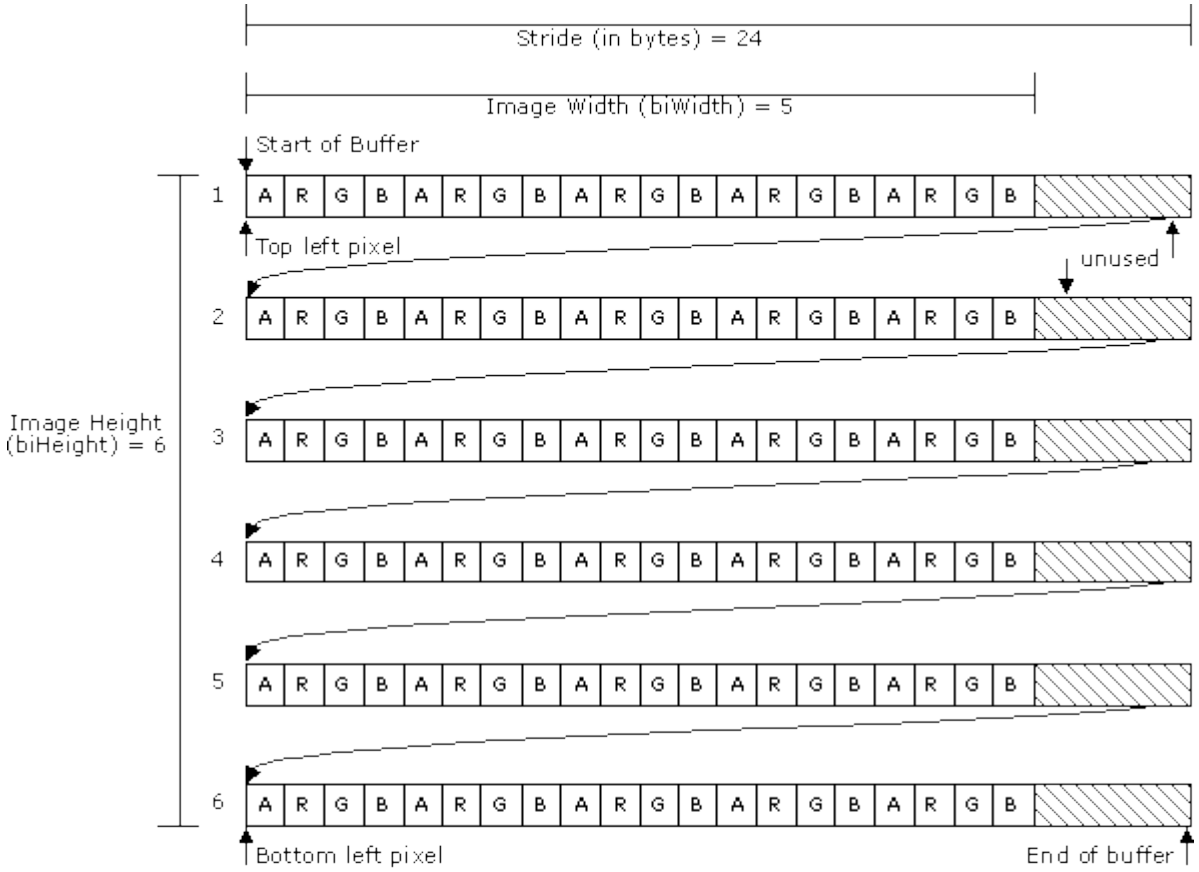


Image orientation of a bottom-up bitmap
biHeight is positive, stride is negative

In a top-down DIB, the order of the rows is reversed. The *top* row of the image is the first row in memory, followed by the next row down. The bottom row of the image is the last row in the buffer. With a top-down DIB, the first byte in memory is the *top-left* pixel of the image. DirectDraw uses top-down DIBs. The following diagram shows the physical layout of a top-down DIB:



Physical layouts of bytes in a 5x6 top-down ARGB bitmap
Note: the actual number of unused alignment bytes at the end of each row will vary depending on the graphics hardware and/or other factors. In some cases this number may be zero.

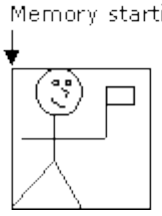


Image orientation of a top-down bitmap
biHeight is negative, stride is positive

For RGB DIBs, the image orientation is indicated by the **biHeight** member of the **BITMAPINFOHEADER** structure. If **biHeight** is positive, the image is bottom-up. If **biHeight** is negative, the image is top-down.

DIBs in YUV formats are always top-down, and the sign of the **biHeight** member is ignored. Decoders should offer YUV formats with positive **biHeight**, but they should also accept YUV formats with negative **biHeight** and ignore the sign.

Also, any DIB type that uses a **FOURCC** in the **biCompression** member, should express its **biHeight** as a positive number no matter what its orientation is, since the **FOURCC** itself identifies a compression scheme whose image orientation should be understood by any compatible filter.