1．位图和调色板的概念

如今 Windows（3.x 以及 95，NT）系列已经成为决大多数用户使用的操作系统。它比 DOS 成功的一个重要因素是它可视化的漂亮界面，

例如你可以在桌面上铺上你喜欢的墙纸。那么 Windows 是如何显示图象的呢？这就要谈到位图（Bitmap）。

我们知道，普通的显示器屏幕是由许许多多的点构成的，我们称之为象素。显示时采用扫描的方法：电子枪每次从左到右扫描一行，

为每个象素着色，然后从上到下这样扫描若干行，就扫过了一屏。为了防止闪烁，每秒要重复上述过程几十次。例如我们常说的屏

幕分辨率为 640*480，刷新

频率为 70Hz，意思是说每行要扫描 640 个象素，一共有 480 行，每秒重复扫描屏幕 70 次。我们称这种显示器为位映象设备。所谓位映

象，就是指一个二维的象素矩阵，而位图就是采用位映象方法显示和存储的图象。举个例子，下图 1 是一幅普通的黑白位图，图 2 是

被放大后的图，图中每个方

那么，彩色图是怎么回事呢？

我们先来说说三元色 RGB 概念。我们知道，自然界中的所有颜色都可以由红，绿，蓝（R，G，B）组合而成。有的颜色含有红色成分

多一些，如深红；有的含有红色成分少一些，如淡红。针对含有红色成分的多少，可以分成 0 到 255 共 256 个等级，0 级表示不含红色

成分，255 级表示含有 100%的

红色成分。同样，绿色和蓝色也被分成 256 级。这种分级的概念被称作量化。这样，根据红，绿，蓝各种不同的组合我们就能表示出

256*256*256，约 1 千 6 百万种颜色。这么多颜色对于我们人眼来已经足够了。

下表是常见的一些颜色的 RGB 组合值。

颜色	R	G	B
红	255	0	0
蓝	0	0	255
绿	0	255	0
黄	255	255	0
紫	255	0	255
青	0	255	255
白	255	255	255
黑	0	0	0
灰	128	128	128

你大概已经明白了，当一幅图中每个象素赋予不同的 RGB 值时，就能呈现出五彩缤纷的颜色了，这样就形成了彩色图。对，是这样的

，但实际上的做法还有些差别。

让我们来看看下面的例子。

有一个长宽各为 200 个象素，颜色数为 16 色的彩色图，每一个象素都用 R，G，B 三个分量表示，因为每个分量有 256 个级别，要用 8 位（
bit），即一个字节（byte）来表示，所以每个象素需要用 3 个字节。整个图象要用 200*200*3，约 120k 字节，可不是一个小数目呀！
如果我们用下面的方法，
就能省的多。
因为是一个 16 色图，也就是说这幅图中最多只有 16 种颜色，我们可以用一个表：表中的每一行记录一种颜色的 R，G，B 值。这样当我们
表示一个象素的颜色时，只需要指出该颜色是在第几行，即该颜色在表中的索引值。举个例子，如果表的第 0 行为 255，0，0（红色）
，那么当某个象素为

色时，只需要标明 0 即可。 让我们再来计算一下：16 种状态可以用 4 位（bit）表示，所以一个象素要用半个字节。整个图象要用 200*2
00*0.5，约 20k 字节，再加上表占用的字节为 3*16=48 字节.整个占用的字节数约为前面的 1/6，省很多吧。
这张 RGB 的表，即是我们常说的调色板(Palette)，另一种叫法是颜色查找表 LUT(LookUpTable),似乎更确切一些。Windows 位图中便用

到了调色板技术.其实是不光是 Windows 位图，许多图象文件格式如 pcx，tif，gif 等都用到了。所以很好地掌握调色板的概念是十分重
要的.
有一种图，它的颜色数高达 256*256*256 种，也就是说包含我们上述提到的 R，G，B 颜色表示方法中所有的颜色，这种图叫做真彩色图(
TrueColor)。真彩色图并不是说一幅图包含了所有的颜色，而是说它具有显示所有颜色的能力，即最多可以包含所有的颜色。表示真彩

色图时，每个象素直
接用 R，G，B 三个分量字节表示，而不采用调色板技术，原因很明显：如果用调色板，表示一个象素也要用 24 位，这是因为每种颜色的
索引要用 24 位（因为总共有 2 的 24 次方种颜色，即调色板有 2 的 24 次方行)，和直接用 R，G，B 三个分量表示用的字节数一样，不但没有任
何便宜，还要加上一

个 256*256*256*3 个字节的大调色板。所以真彩色图直接用 R，G，B 三个分量表示，它又叫做 24 位色图。
2．Bmp 文件格式
介绍完位图和调色板的概念，下面就让我们来看一看 Windows 的位图文件(.bmp 文件）的格式是什么样子的。 bmp 文件大体上分成四个部
分，如图 3 所示。

图 3.Windows 位图文件结构示意图(右)

第一部分为位图文件头 BITMAPFILEHEADER，是一个结构，其定义如下：
typedefstructtagBITMAPFILEHEADER{
WORD bfType;
DWORD bfSize;
WORD bfReserved1;
WORD bfReserved2;
DWORD bfOffBits;
} BITMAPFILEHEADER;
这个结构的长度是固定的，为 14 个字节（WORD 为无符号 16 位整数，DWORD 为无符号 32 位整数），各个域的说明如下：
bfType
指定文件类型，必须是 0x424D，即字符串"BM"，也就是说所有.bmp 文件的头两个字节都是"BM"
bfSize
指定文件大小，包括这 14 个字节
bfReserved1，bfReserved2
为保留字，不用考虑
bfOffBits
为从文件头到实际的位图数据的偏移字节数，即图 3 中前三个部分的长度之和。
第二部分为位图信息头 BITMAPINFOHEADER，也是一个结构，其定义如下：
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; 这个结构的长度是固定的，为 40 个字节（WORD 为无符号 16 位整数，DWORD 无符号 32 位整数,LONG 为 32 位整数），各个域的说明如下：
biSize
指定这个结构的长度，为 40
biWidth
指定图象的宽度，单位是象素
biHeight
指定图象的高度，单位是象素
biPlanes
必须是 1，不用考虑
biBitCount
指定表示颜色时要用到的位数，常用的值为 1（黑白二色图）,4（16 色图）,8（256 色）,24

（真彩色图）（新的.bmp 格式支持 32 位色，这里就不做讨论了）。

biCompression

指定位图是否压缩，有效的值为 BI_RGB，BI_RLE8，BI_RLE4，BI_BITFIELDS（都是一些 Windows 定义好的常量）。要说明的是，Windows 位图可以采用 RLE4，和 RLE8 的压缩格式，但用的不多。我们今后所讨论的只有第一种不压缩的情况，即 biCompression 为 BI_RGB 的情况。

biSizeImage

指定实际的位图数据占用的字节数，其实也可以从以下的公式中计算出来：

biSizeImage=biWidth'*biHeight

要注意的是：上述公式中的 biWidth'必须是 4 的整倍数（所以不是 biWidth，而是 biWidth'，表示大于或等于 biWidth 的，离 4 最近的整倍数。举个例子，如果 biWidth=240，则 biWidth'=240；如果 biWidth=241，biWidth'=244）如果 biCompression 为 BI_RGB，则该项可能为零

biXPelsPerMeter

指定目标设备的水平分辨率，单位是每米的象素个数，关于分辨率的概念，我们将在打印部分详细介绍。

biYPelsPerMeter

指定目标设备的垂直分辨率，单位同上。

biClrUsed

指定本图象实际用到的颜色数，如果该值为零，则用到的颜色数为 2 的 biBitCount 次方。

biClrImportant

指定本图象中重要的颜色数，如果该值为零，则认为所有的颜色都是重要的。

第三部分为调色板(Palette)，当然，这里是对那些需要调色板的位图文件而言的。有些位图，如真彩色图，前面已经讲过，是不需要调色板的，BITMAPINFOHEADER 后直接是位图数据。

调色板实际上是一个数组，共有 biClrUsed 个元素（如果该值为零，则有 2 的 biBitCount 次方个元素）。数组中每个元素的类型是一个 RGBQUAD 结构，占 4 个字节，其定义如下：

```
typedef struct tagRGBQUAD{
BYTE rgbBlue; //该颜色的蓝色分量
BYTE rgbGreen; //该颜色的绿色分量
BYTE rgbRed; //该颜色的红色分量
BYTE rgbReserved; //保留值
} RGBQUAD;
```

第四部分就是实际的图象数据了。对于用到调色板的位图，图象数据就是该像素颜在调色板中的索引值，对于真彩色图，图象数据就是实际的 R,G,B 值。下面就 2 色，16 色，256 色位图和真彩色位图分别介绍。

对于 2 色位图，用 1 位就可以表示该像素的颜色（一般 0 表示黑，1 表示白），所以一个字节可以表示 8 个像素。

对于 16 色位图，用 4 位可以表示一个像素的颜色，所以一个字节可以表示 2 个像素。

对于 256 色位图，一个字节刚好可以表示 1 个像素。

对于真彩色图，三个字节才能表示 1 个像素。

要注意两点：

1．每一行的字节数必须是 4 的整倍数，如果不是，则需要补齐。这在前面介绍 biSizeImage 时已经提到了。

2．一般来说，.BMP 文件的数据从下到上，从左到右的。也就是说，从文件中最先读到的是图象最下面一行的左边第一个像素，然后是左边第二个像素…接下来是倒数第二行左边第一个像素，左边第二个像素…依次类推，最后得到的是最上面一行的最右一个像素。
好了，终于介绍完 bmp 文件结构了，是不是觉得头有些大

--

标　题: bmp 图像文件格式说明(2)

BMP 文件格式分析

本来不想写这篇东西，因为介绍 BMP 文件结构的资料太多了，都有些滥了。但刚写完 BMP 的
读写模块，又不想不留下点什么，所以就写了，全当是学习笔记吧。自己以后查资料时 也
方便一些，也许对某些初哥还会有点用^_^

注：本文参考了林福宗老师的有关 BMP 文件格式的文章，在此声明。

简介

BMP(Bitmap-File)图形文件是 Windows 采用的图形文件格式，在 Windows 环境下运行的所有
图象处理软件都支持 BMP 图象文件格式。Windows 系统内部各图像绘制操作都是以 BMP 为基础
的。Windows
3.0 以前的 BMP 图文件格式与显示设备有关，因此把这种 BMP 图象文件格式称为
设备相关位图 DDB(device-dependent bitmap)文件格式。Windows
3.0 以后的 BMP 图象文件
与显示设备无关，因此把这种 BMP 图象文件格式称为设备无关位图 DIB(device-independ en
t bitmap)格式（注：Windows
3.0 以后，在系统中仍然存在 DDB 位图，象 BitBlt()这种函数
就是基于 DDB 位图的，只不过如果你想将图像以 BMP 格式保存到磁盘文件中时，微软极力
推
荐你以 DIB 格式保存），目的是为了让 Windows 能够在任何类型的显示设备上显示所存储的
图象。BMP 位图文件默认的文件扩展名是 BMP 或者 bmp（有时它也会以.DIB 或.RLE 作扩展名）
。

6.1.2 文件结构

位图文件可看成由 4 个部分组成：位图文件头(bitmap-file
header)、位图信息头(bitmap
-information header)、彩色表(color
table)和定义位图的字节阵列，它具有如下所示的
形式。

位图文件的组成
结构名称
符号

位图文件头(bitmap-file header) BITMAPFILEHEADER bmfh
位图信息头(bitmap-information header) BITMAPINFOHEADER bmih
彩色表(color table) RGBQUAD aColors[]
图象数据阵列字节 BYTE aBitmapBits[]

位图文件结构可综合在表 6-01 中。

表 01 位图文件结构内容摘要

偏移量
域的名称
大小
内容

图象文件

头
0000h 文件标识 2 bytes 两字节的内容用来识别位图的类型：
‘BM’ ： Windows 3.1x, 95, NT, …

‘BA’ ：OS/2 Bitmap Array

‘CI’ ：OS/2 Color Icon

‘CP’ ：OS/2 Color Pointer

‘IC’ ： OS/2 Icon

‘PT’ ：OS/2 Pointer

注：因为 OS/2 系统并没有被普及开，所以在编程时，你只需判断第一个标识“BM”就行
。

0002h File Size 1 dword 用字节表示的整个文件的大小
0006h Reserved 1 dword 保留，必须设置为 0
000Ah Bitmap Data Offset 1 dword
从文件开始到位图数据开始之间的数据(bitmap
data)之间的偏移量
000Eh Bitmap Header Size 1 dword 位图信息头(Bitmap Info
Header)的长度，用来
描述位图的颜色、压缩方法等。下面的长度表示：
28h - Windows 3.1x, 95, NT, …

0Ch - OS/2 1.x

F0h - OS/2 2.x

注：在 Windows95、98、2000 等操作系统中，位图信息头的长度并不一定是 28h，因为微软
已经制定出了新的 BMP 文件格式，其中的信息头结构变化比较大，长度加长。所以最好不要
直接使用常数 28h，而是应该从具体的文件中读取这个值。这样才能确保程序的兼容性。

0012h Width 1 dword 位图的宽度，以象素为单位
0016h Height 1 dword 位图的高度，以象素为单位
001Ah Planes 1 word 位图的位面数（注：该值将总是 1）

图象
信息

头

001Ch Bits Per Pixel 1 word 每个象素的位数
1 -
单色位图（实际上可有两种颜色，缺省情况下是黑色和白色。你可以自己定义这两种
颜色）

4 - 16 色位图

8 - 256 色位图

16 - 16bit 高彩色位图

24 - 24bit 真彩色位图

32 - 32bit 增强型真彩色位图

001Eh Compression 1 dword 压缩说明：
0 - 不压缩 (使用 BI_RGB 表示)

1 - RLE 8-使用 8 位 RLE 压缩方式(用 BI_RLE8 表示)

2 - RLE 4-使用 4 位 RLE 压缩方式(用 BI_RLE4 表示)

3 - Bitfields-位域存放方式(用 BI_BITFIELDS 表示)

0022h Bitmap Data Size 1 dword
用字节数表示的位图数据的大小。该数必须是 4 的倍
数
0026h HResolution 1 dword 用象素/米表示的水平分辨率
002Ah VResolution 1 dword 用象素/米表示的垂直分辨率
002Eh Colors 1 dword 位图使用的颜色数。如 8-比特/象素表示为 100h 或者
256.
0032h Important Colors 1 dword
指定重要的颜色数。当该域的值等于颜色数时（或
者等于 0 时），表示所有颜色都一样重要
调色板数据 根据 BMP 版本的不同而不同 Palette N * 4 byte
调色板规范。对于调色板中
的每个表项，这 4 个字节用下述方法来描述 RGB 的值： 1 字节用于蓝色分量
1 字节用于绿色分量
1 字节用于红色分量
1 字节用于填充符(设置为 0)

图象数据 根据 BMP 版本及调色板尺寸的不同而不同 Bitmap Data xxx bytes
该域的大小取
决于压缩方法及图像的尺寸和图像的位深度，它包含所有的位图数据字节，这些数据可
能
是彩色调色板的索引号，也可能是实际的 RGB 值，这将根据图像信息头中的位深度值来决
定
。

构件详解

1. 位图文件头

位图文件头包含有关于文件类型、文件大小、存放位置等信息，在 Windows 3.0 以上版本的
位图文件中用 BITMAPFILEHEADER 结构来定义：

```
typedef struct tagBITMAPFILEHEADER { /* bmfh */

UINT bfType;
DWORD bfSize;
UINT bfReserved1;
UINT bfReserved2;
DWORD bfOffBits;

} BITMAPFILEHEADER;
```

其中：

bfType
 说明文件的类型.（该值必需是 0x4D42，也就是字符'BM'。我们不需要判断 OS/2 的位图标
识，这么做现在来看似乎已经没有什么意义了，而且如果要支持 OS/2 的位图，程序将变得
很繁琐。所以，在此只建议你检察'BM'标识）

bfSize
 说明文件的大小，用字节为单位

bfReserved1
 保留，必须设置为 0

bfReserved2
 保留，必须设置为 0

bfOffBits
 说明从文件头开始到实际的图象数据之间的字节的偏移量。这个参数是非常有用的，因为位图信息头和调色板的长度会根据不同情况而变化，所以你可以用这个偏移值迅速的从文件中读取到位数据。

2. 位图信息头

位图信息用 BITMAPINFO 结构来定义，它由位图信息头(bitmap-information
header)和彩色
表(color
table)组成，前者用 BITMAPINFOHEADER 结构定义，后者用 RGBQUAD 结构定义。BI
TMAPINFO 结构具有如下形式：

typedef struct tagBITMAPINFO { /* bmi */

BITMAPINFOHEADER bmiHeader;
RGBQUAD bmiColors[1];

} BITMAPINFO;

其中：

bmiHeader
 说明 BITMAPINFOHEADER 结构，其中包含了有关位图的尺寸及位格式等信息

bmiColors
 说明彩色表 RGBQUAD 结构的阵列，其中包含索引图像的真实 RGB 值。

BITMAPINFOHEADER 结构包含有位图文件的大小、压缩类型和颜色格式，其结构定义为：

typedef struct tagBITMAPINFOHEADER { /* bmih */

DWORD biSize;
LONG biWidth;
LONG biHeight;
WORD biPlanes;
WORD biBitCount;
DWORD biCompression;
DWORD biSizeImage;
LONG biXPelsPerMeter;
LONG biYPelsPerMeter;
DWORD biClrUsed;
DWORD biClrImportant;

} BITMAPINFOHEADER;

其中：

biSize
说明 BITMAPINFOHEADER 结构所需要的字数。注：这个值并不一定是 BITMAPINFOHEADER 结构
的尺寸，它也可能是 sizeof(BITMAPV4HEADER)的值，或是 sizeof(BITMAPV5HEADER)的值
。
这要根据该位图文件的格式版本来决定，不过，就现在的情况来看，绝大多数的 BMP 图像都
是 BITMAPINFOHEADER 结构的（可能是后两者太新的缘故吧:-）。

biWidth
说明图象的宽度，以象素为单位

biHeight
说明图象的高度，以象素为单位。注：这个值除了用于描述图像的高度之外，它还有另一个用处，就是指明该图像是倒向的位图，还是正向的位图。如果该值是一个正数，说明图
像是倒向的，如果该值是一个负数，则说明图像是正向的。大多数的 BMP 文件都是倒向的位
图，也就是时，高度值是一个正数。（注：当高度值是一个负数时（正向图像），图像将
不能被压缩（也就是说 biCompression 成员将不能是 BI_RLE8 或 BI_RLE4）。

biPlanes
为目标设备说明位面数，其值将总是被设为 1

biBitCount
说明比特数/象素，其值为 1、4、8、16、24、或 32

biCompression
说明图象数据压缩的类型。其值可以是下述值之一：
BI_RGB：没有压缩；

BI_RLE8：每个象素 8 比特的 RLE 压缩编码，压缩格式由 2 字节组成(重复象素计数和颜色
索
引)；

BI_RLE4：每个象素 4 比特的 RLE 压缩编码，压缩格式由 2 字节组成

BI_BITFIELDS：每个象素的比特由指定的掩码决定。

biSizeImage
 说明图象的大小，以字节为单位。当用 BI_RGB 格式时，可设置为 0
biXPelsPerMeter
 说明水平分辨率，用象素/米表示
biYPelsPerMeter
 说明垂直分辨率，用象素/米表示
biClrUsed
 说明位图实际使用的彩色表中的颜色索引数（设为 0 的话，则说明使用所有调色板项）

biClrImportant
 说明对图象显示有重要影响的颜色索引的数目，如果是 0，表示都重要。

现就 BITMAPINFOHEADER 结构作如下说明：

(1) 彩色表的定位

应用程序可使用存储在 biSize 成员中的信息来查找在 BITMAPINFO 结构中的彩色表，如下所
示：

pColor = ((LPSTR) pBitmapInfo + (WORD) (pBitmapInfo->bmiHeader.biSize))

(2) biBitCount

biBitCount=1
表示位图最多有两种颜色，缺省情况下是黑色和白色，你也可以自己定义这
两种颜色。图像信息头装调色板中将有两个调色板项，称为索引 0 和索引 1。图象数据阵列
中的每一位表示一个象素。如果一个位是 0，显示时就使用索引 0 的 RGB 值，如果位是 1，则
使用索引 1 的 RGB 值。

biBitCount=4
表示位图最多有 16 种颜色。每个象素用 4 位表示，并用这 4 位作为彩色表的表
项来查找该象素的颜色。例如，如果位图中的第一个字节为 0x1F，它表示有两个象素，第一象素的颜色就在彩色表的第 2 表项中查找，而第二个象素的颜色就在彩色表的第 16 表项中
查找。此时，调色板中缺省情况下会有 16 个 RGB 项。对应于索引 0 到索引 15。

biBitCount=8
表示位图最多有 256 种颜色。每个象素用 8 位表示，并用这 8 位作为彩色表的

表项来查找该象素的颜色。例如，如果位图中的第一个字节为 0x1F，这个象素的颜色就在彩色表的第 32 表项中查找。此时，缺省情况下，调色板中会有 256 个 RGB 项，对应于索引 0 到
索引 255。

biBitCount=16
表示位图最多有 216 种颜色。每个色素用 16 位（2 个字节）表示。这种格式
叫作高彩色，或叫增强型 16 位色，或 64K 色。它的情况比较复杂，当 biCompression 成员的
值是 BI_RGB 时，它没有调色板。16 位中，最低的 5 位表示蓝色分量，中间的 5 位表示绿色
分
量，高的 5 位表示红色分量，一共占用了 15 位，最高的一位保留，设为 0。这种格式也被称
作 555
16 位位图。如果 biCompression 成员的值是 BI_BITFIELDS，那么情况就复杂了，首先
是原来调色板的位置被三个 DWORD 变量占据，称为红、绿、蓝掩码。分别用于描述红、绿、
蓝分量在 16 位中所占的位置。在 Windows
95（或 98）中，系统可接受两种格式的位域：55
5 和 565，在 555 格式下，红、绿、蓝的掩码分别是：0x7C00、0x03E0、0x001F，而在 565 格
式下，它们则分别为：0xF800、0x07E0、0x001F。你在读取一个像素之后，可以分别用掩
码“与”上像素值，从而提取出想要的颜色分量（当然还要再经过适当的左右移操作）。
在 NT 系统中，则没有格式限制，只不过要求掩码之间不能有重叠。（注：这种格式的图像
使用起来是比较麻烦的，不过因为它的显示效果接近于真彩，而图像数据又比真彩图像小
的多，所以，它更多的被用于游戏软件）。

biBitCount=24
表示位图最多有 224 种颜色。这种位图没有调色板（bmiColors 成员尺寸为
0），在位数组中，每 3 个字节代表一个象素，分别对应于颜色 R、G、B。

biBitCount=32
表示位图最多有 232 种颜色。这种位图的结构与 16 位位图结构非常类似，当
biCompression 成员的值是 BI_RGB 时，它也没有调色板，32 位中有 24 位用于存放 RGB 值，顺
序是：最高位—保留，红 8 位、绿 8 位、蓝 8 位。这种格式也被成为 888
32 位图。如果 biCo
mpression 成员的值是 BI_BITFIELDS 时，原来调色板的位置将被三个 DWORD 变量占据，成

为
红、绿、蓝掩码，分别用于描述红、绿、蓝分量在 32 位中所占的位置。在 Windows 95(or
98)中，系统只接受 888 格式，也就是说三个掩码的值将只能是：0xFF0000、0xFF00、0x FF
。而在 NT 系统中，你只要注意使掩码之间不产生重叠就行。（注：这种图像格式比较规整
，因为它是 DWORD 对齐的，所以在内存中进行图像处理时可进行汇编级的代码优化（简单）
）。

(3) ClrUsed

BITMAPINFOHEADER 结构中的成员 ClrUsed 指定实际使用的颜色数目。如果 ClrUsed 设置成
0，
位图使用的颜色数目就等于 biBitCount 成员中的数目。请注意，如果 ClrUsed 的值不是可用颜色的最大值或不是 0，则在编程时应该注意调色板尺寸的计算，比如在 4 位位图中，调色
板的缺省尺寸应该是 16＊sizeof(RGBQUAD)，但是，如果 ClrUsed 的值不是 16 或者不是 0，那
么调色板的尺寸就应该是 ClrUsed＊sizeof(RGBQUAD)。

(4) 图象数据压缩

①
BI_RLE8：每个象素为 8 比特的 RLE 压缩编码，可使用编码方式和绝对方式中的任何一种进行压缩，这两种方式可在同一幅图中的任何地方使用。

编码方式：由 2 个字节组成，第一个字节指定使用相同颜色的象素数目，第二个字节指定使用的颜色索引。此外，这个字节对中的第一个字节可设置为 0，联合使用第二个字节的值表示：

第二个字节的值为 0：行的结束。

第二个字节的值为 1：图象结束。

第二个字节的值为 2：其后的两个字节表示下一个象素从当前开始的水平和垂直位置的偏
移量。

绝对方式：第一个字节设置为 0，而第二个字节设置为 0x03～0xFF 之间的一个值。在这种

方
式中，第二个字节表示跟在这个字节后面的字节数，每个字节包含单个象素的颜色索引
。
压缩数据格式需要字边界(word
boundary)对齐。下面的例子是用 16 进制表示的 8-位压缩图
象数据：

03 04 05 06 00 03 45 56 67 00 02 78 00 02 05 01 02 78 00 00 09 1E 00 01
这些压缩数据可解释为 ：

压缩数据
 扩展数据

03 04 04 04 04
05 06 06 06 06 06 06
00 03 45 56 67 00 45 56 67
02 78 78 78
00 02 05 01 从当前位置右移 5 个位置后向下移一行
02 78 78 78
00 00 行结束
09 1E 1E 1E 1E 1E 1E 1E 1E 1E 1E
00 01 RLE 编码图象结束

②
BI_RLE4：每个象素为 4 比特的 RLE 压缩编码，同样也可使用编码方式和绝对方式中的任何一种进行压缩，这两种方式也可在同一幅图中的任何地方使用。这两种方式是：

编码方式：由 2 个字节组成，第一个字节指定象素数目，第二个字节包含两种颜色索引，一个在高 4 位，另一个在低 4 位。第一个象素使用高 4 位的颜色索引，第二个使用低 4 位的颜色
索引，第 3 个使用高 4 位的颜色索引，依此类推。

绝对方式：这个字节对中的第一个字节设置为 0，第二个字节包含有颜色索引数，其后续字节包含有颜色索引，颜色索引存放在该字节的高、低 4 位中，一个颜色索引对应一个象素。此外，BI_RLE4 也同样联合使用第二个字节中的值表示：

 第二个字节的值为 0：行的结束。

 第二个字节的值为 1：图象结束。

 第二个字节的值为 2：其后的两个字节表示下一个象素从当前开始的水平和垂直位置的偏
移量。

下面的例子是用 16 进制数表示的 4-位压缩图象数据：

03 04 05 06 00 06 45 56 67 00 04 78 00 02 05 01 04 78 00 00 09 1E 00 01

这些压缩数据可解释为 ：

压缩数据
 扩展数据

03 04 0 4 0
05 06 0 6 0 6 0
00 06 45 56 67 00 4 5 5 6 6 7
04 78 7 8 7 8
00 02 05 01 从当前位置右移 5 个位置后向下移一行
04 78 7 8 7 8
00 00 行结束
09 1E 1 E 1 E 1 E 1
00 01 RLE 图象结束

3. 彩色表

彩色表包含的元素与位图所具有的颜色数相同，象素的颜色用 RGBQUAD 结构来定义。对于 2
4-位真彩色图象就不使用彩色表（同样也包括 16 位、和 32 位位图），因为位图中的 RGB 值就
代表了每个象素的颜色。彩色表中的颜色按颜色的重要性排序，这可以辅助显示驱动程序
为不能显示足够多颜色数的显示设备显示彩色图象。RGBQUAD 结构描述由 R、G、B 相对强
度
组成的颜色，定义如下：

typedef struct tagRGBQUAD { /* rgbq */

BYTE rgbBlue;
BYTE rgbGreen;
BYTE rgbRed;
BYTE rgbReserved;

} RGBQUAD;

其中：

rgbBlue
 指定蓝色强度

rgbGreen
 指定绿色强度

rgbRed
 指定红色强度

rgbReserved
 保留，设置为 0

4. 位图数据

紧跟在彩色表之后的是图象数据字节阵列。图象的每一扫描行由表示图象象素的连续的字
节组成，每一行的字节数取决于图象的颜色数目和用象素表示的图象宽度。扫描行是由底
向上存储的，这就是说，阵列中的第一个字节表示位图左下角的象素，而最后一个字节表
示位图右上角的象素。（只针对与倒向 DIB，如果是正向 DIB，则扫描行是由顶向下存储的
），倒向 DIB 的原点在图像的左下角，而正向 DIB 的原点在图像的左上角。同时，每一扫描
行的字节数必需是 4 的整倍数，也就是 DWORD 对齐的。如果你想确保图像的扫描行 DWORD 对齐
，可使用下面的代码：

(((width*biBitCount)+31)>>5)<<2

5. 参考书目

《图象文件格式(上、下)—Windows 编程》
《图像文件格式大全》
《Programming Windows by Charles Petzold》

6. 相关站点

各种格式：http://www.wotsit.org/← [m
各种格式：http://www.csdn.net/← [m
位图格式：http://www.cica.indiana.edu/graphics/image_specs/bmp.format.txt← [m