BMP文件格式，又称为Bitmap（位图）或是DIB(Device-Independent Device，设备无关位图)，是Windows系统中广泛使用的图像文件格式。由于它可以不作任何变换地保存图像像素域的数据，因此成为我们取得RAW数据的重要来源

BMP文件的数据按照从文件头开始的先后顺序分为四个部分：

Ø **bmp文件头(bmp file header)**：提供文件的格式、大小等信息

Ø **位图信息头(bitmap information)**：提供图像数据的尺寸、位平面数、压缩方式、颜色索引等信息

Ø **调色板(color palette)**：可选，如使用索引来表示图像，调色板就是索引与其对应的颜色的映射表

Ø **位图数据(bitmap data)**：就是图像数据啦，原始数据

一、bmp文件头

Windows为bmp文件头定义了如下结构体：

```
 typedef struct tagBITMAPFILEHEADER
{
UINT16 bfType;
DWORD bfSize;
UINT16 bfReserved1;
UINT16 bfReserved2;
DWORD bfOffBits;
} BITMAPFILEHEADER;
```

其中：

变量名	地址偏移	大小	作用
bfType	0000h	2 bytes	说明文件的类型，可取值为： • 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
bfSize	0002h	4 bytes	说明该位图文件的大小，用字节为单位
bfReserved1	0006h	2 bytes	保留，必须设置为0
bfReserved2	0008h	2 bytes	保留，必须设置为0
bfOffBits	000Ah	4 bytes	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， 所以我们可以用这个[illegible] 读取到位图数据。

\

二、位图信息头

同样地，Windows为位图信息头定义了如下结构体：

代码

变量名	地址偏移	大小	作用
biSize	000Eh	4 bytes	BITMAPINFOHEADER结构所需要的字数。
biWidth	0012h	4 bytes	说明图像的宽度，用像素为单位
biHeight	0016h	4 bytes	说明图像的高度，以像素为单位。 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，也就是高度值是一个正数。
biPlanes	001Ah	2 bytes	为目标设备说明颜色平面数， 其值将总是被设为1。
biBitCount	001Ch	2 bytes	说明比特数/像素，其值为1、4、8、16、24或32。
biCompression	001Eh	4 bytes	说明图像数据压缩的类型。取值范围： 0 BI_RGB 不压缩（最常用） 1 BI_RLE8 8比特游程编码（RLE），只用于8位位图 2 BI_RLE4 4比特游程编码（RLE），只用于4位位图 3 BI_BITFIELDS 比特域，用于16/32位位图 4 BI_JPEG JPEG 位图含JPEG图像（仅用于打印机） 5 BI_PNG PNG 位图含PNG图像（仅用于打印机）
biSizeImage	0022h	4 bytes	说明图像的大小， 以字节为单位。当用BI_RGB格式时，可设置为0。
biXPelsPerMeter	0026h	4 bytes	说明水平分辨率，用像素/米表示，有符号整数
biYPelsPerMeter	002Ah	4 bytes	说明垂直分辨率，用像素/米表示，有符号整数
biClrUsed	002Eh	4 bytes	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[illegible]板项）
biClrImportant	0032h	4 bytes	说明对图像显示有重要影响的[illegible] 如果是0，表示都重要。

\

大小	文件头	标识	兼容性（被哪些GDI支持）
12	OS/2 V1	BITMAPCOREHEADER	OS/2以及Windows 3.0以上的Windows版本
64	OS/2 V2	BITMAPCOREHEADER2	
40	Windows V3	BITMAPINFOHEADER	Windows 3.0以上的Windows版本
108	Windows V4	BITMAPV4HEADER	Windows 95/NT4以上的Windows版本
124	Windows V5	BITMAPV5HEADER	Windows 98/2000及其新版本

三、调色板

下面的数据就是调色板了。前面也已经提过，调色板其实是一张映射表，标识颜色索引号与其代表的颜色的对应关系。它在文件中的布局就像一个二维数组palette[N][4],其中N表示总的颜色索引数，每行的四个元素分别表示该索引对应的B、G、R和Alpha的值，每个分量占一个字节。如不设透明通道时，Alpha为0。因为前面知道，本图有256个颜色索引，因此N＝256。索引号就是所在行的行号，对应的颜色就是所在行的四个元素。这里截取一些数据来说明：

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+a	+b	+c	+d	+e	+f	Dump
0000	42	4d	36	04	01	00	00	00	00	00	36	04	00	00	28	00	BM6.....6...(.
0010	00	00	00	01	00	00	00	01	00	00	01	00	08	00	00	00	.....
0020	00	00	00	00	01	00	00	00	00	00	00	00	00	00	00	01	.....
0030	00	00	00	01	00	00	fe	fa	fd	00	fd	f3	fc	00	f4	f3	......þúý.ýóü.óó
0040	fc	00	fc	f2	f4	00	f6	f2	f2	00	fb	f9	f6	00	ea	f3	ü.üòô.öòò.ûùö.êó
0050	f8	00	fb	ee	fa	00	fb	ee	f3	00	f4	ed	f2	00	f4	ea	ø.ûîú.ûîó.ôíò.ôê

\

索引：(蓝，绿，红，Alpha)

0号：(fe，fa，fd，00)

1号：(fd，f3，fc，00)

2号：(f4，f3，fc，00)

3号：(fc，f2，f4，00)

4号：(f6，f2，f2，00)

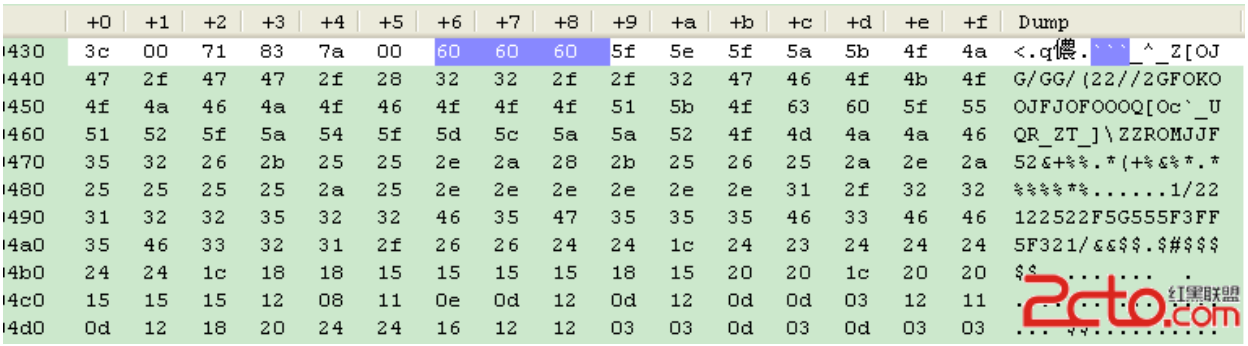
5号：(fb，f9，f6，00) 等等。

一共有256种颜色，每个颜色占用4个字节，就是一共1024个字节，再加上前面的文件信息头和位图信息头的54个字节加起来一共是1078个字节。也就是说在位图数据出现之前一共有1078个字节，与我们在文件信息头得到的信息：文件头到文图数据区的偏移为1078个字节一致！

四、位图数据

下面就是位图数据了，每个像素占一个字节，取得这个字节后，以该字节为索引查询相应的颜色，并显示到相应的显示设备上就可以了。

注意：由于位图信息头中的图像高度是正数，所以位图数据在文件中的排列顺序是从左下角到右上角，以行为主序排列的。



	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+a	+b	+c	+d	+e	+f	Dump
1430	3c	00	71	83	7a	00	60	60	60	5f	5e	5f	5a	5b	4f	4a	<.q僾.```_^_Z[OJ
1440	47	2f	47	47	2f	28	32	32	2f	2f	32	47	46	4f	4b	4f	G/GG/(22//2GFOKO
1450	4f	4a	46	4a	4f	46	4f	4f	4f	51	5b	4f	63	60	5f	55	OJFJOFOOOQ[Oc`_U
1460	51	52	5f	5a	54	5f	5d	5c	5a	5a	52	4f	4d	4a	4a	46	QR_ZT_]\ZZROMJJF
1470	35	32	26	2b	25	25	2e	2a	28	2b	25	26	25	2a	2e	2a	52&+%%.*(+%&%*.*
1480	25	25	25	25	2a	25	2e	2e	2e	2e	2e	2e	31	2f	32	32	%%%%*%.....1/22
1490	31	32	32	35	32	32	46	35	47	35	35	35	46	33	46	46	122522F5G555F3FF
14a0	35	46	33	32	31	2f	26	26	24	24	1c	24	23	24	24	24	5F321/&&$$.$#$$$
14b0	24	24	1c	18	18	15	15	15	15	18	15	20	20	1c	20	20	$$..... .
14c0	15	15	15	12	08	11	0e	0d	12	0d	12	0d	0d	03	12	11	....
14d0	0d	12	18	20	24	24	16	12	12	03	03	0d	03	0d	03	03	... $$.....

\

也即我们见到的第一个像素60是图像最左下角的数据，第二个人像素60为图像最后一行第二列的数据，…一直到最后一行的最后一列数据，后面紧接的是倒数第二行的第一列的数据，依此类推。

如果图像是24位或是32位数据的位图的话，位图数据区就不是索引而是实际的像素值了。

下面说明一下，此时位图数据区的每个像素的RGB颜色阵列排布：

24位RGB按照BGR的顺序来存储每个像素的各颜色通道的值，一个像素的所有颜色分量值都存完后才存下一个下一个像素，不进行交织存储。

32位数据按照BGRA的顺序存储，其余与24位位图的方式一样。

像素的排布规则与前述一致。

对齐规则

讲完了像素的排列规则以及各像素的颜色分量的排列规则，最后我们谈谈数据的对齐规则。

我们知道Windows默认的扫描的最小单位是4字节，如果数据对齐满足这个值的话对于数据的获取速度等都是有很大的增益的。因此，BMP图像顺应了这个要求，要求每行的数据的长度必须是4的倍数，如果不够需要进行比特填充（以0填充），这样可以达到按行的快速存取。这时，位图数据区的大小就未必是 图片宽×每像素字节数×图片高 能表示的了，因为每行可能还需要进行比特填充。

填充后的每行的字节数为：

$$RowSize = 4*\left\lceil \frac{BPP*Width}{32} \right\rceil$$

\

，其中BPP（Bits Per Pixel）为每像素的比特数。

在程序中，我们可以表示为：

int iLineByteCnt = (((m_iImageWidth * m_iBitsPerPixel) + 31) >> 5) << 2;

这样，位图数据区的大小为：

m_iImageDataSize = iLineByteCnt * m_iImageHeight;

我们在扫描完一行数据后，也可能接下来的数据并不是下一行的数据，可能需要跳过一段填充数据：

skip = 4 - ((m_iImageWidth * m_iBitsPerPixel)>>3) & 3;

五、拾遗

至此，我们通过分析一个具体的位图文件例子详细地剖析了位图文件的组成。需要注意的是：我们讲的主要是PC机上的位图文件的构成，对于嵌入式平台，可能在调色板数据段与PC机的不同。如在嵌入式平台上常见的16位r5g6b5位图实际上采用的掩模的方式而不是索引的方式来表示图像。此时，在调色板数据段共有四个部分，每个部分为四个字节，实际表示的是彩色版规范。即：

第一个部分是红色分量的掩模

第二个部分是绿色分量的掩模

第三个部分是蓝色分量的掩模

第四个部分是Alpha分量的掩模（缺省为0）

典型的调色板规范在文件中的顺序为为：

00F8 0000 E007 0000 1F00 0000 0000 0000

其中

00F8 0000为FB00h=1111100000000000（二进制），是蓝红分量的掩码。

E007 0000为 07E0h=0000011111100000（二进制），是绿色分量的掩码。

1F00 0000为001Fh=0000000000011111（二进制），是蓝色分量的掩码。

0000 0000设置为0。

将掩码跟像素值进行“与”运算再进行移位操作就可以得到各色分量值。看看掩码，就可以明白事实上在每个像素值的两个字节16位中，按从高到低取5、6、5位分别就是r、g、b分量

值。取出分量值后把r、g、b值分别乘以8、4、8就可以补齐每个分量为一个字节，再把这三个字节按BGR组合，放入存储器，就可以转换为24位标准BMP格式了。

这样我们假设在位图数据区有一个像素的数据在文件中表示为02 F1。这个数据实际上应为F102：

r = (F102 AND F800) >> 8 = F0h = 240

g= (F102 AND 07E0）>> 3 = 20h = 32

b=(F102 AND 001F) << 3 = 10h =16