BMP

[править | править код]

Материал из Википедии — свободной энциклопедии

Текущая версия страницы пока не проверялась опытными участниками и может значительно отличаться от версии, проверенной 14 октября 2020; проверки требует 1 правка.

Перейти к навигации Перейти к поиску

Windows Bitmap

Расширение	.bmp[1], .dib[1] или .rle[1]
MIME-тип	image/bmp[2]
Разработчик	Майкрософт[3][4]
Тип формата	растровая графика

Медиафайлы на Викискладе

Для термина «Bitmap» см. также другие значения.

У этого термина существуют и другие значения, см. BMP (значения).

BMP (от англ. ***B**it**m**ap* ***P**icture*) — формат хранения растровых изображений, разработанный компанией Microsoft. Файлы формата BMP могут иметь расширения .bmp, .dib и .rle.

С форматом BMP работает огромное количество программ, так как его поддержка интегрирована в операционные системы Windows и OS/2. Кроме того, данные этого формата включаются в двоичные файлы ресурсов RES и в PE-файлы.

В данном формате можно хранить только однослойные растры. На каждый пиксель в разных файлах может приходиться разное количество бит (глубина цвета). Microsoft предлагает битности 1, 2, 4, 8, 16, 24, 32, 48 и 64. В битностях 8 и ниже цвет указывается индексом из таблицы цветов (палитры), а при бóльших — непосредственным значением. Цвет же в любом случае можно задать только в цветовой модели RGB (как при непосредственном указании в пикселе, так и в таблице цветов), но в битностях 16 и 32 можно получить Grayscale с глубиной до 16 и 32 бит, соответственно. Частичная прозрачность реализована альфа-каналом различных битностей, но при этом прозрачность без градаций можно косвенно получить RLE-кодированием.

В большинстве случаев пиксели хранятся в виде относительно простого двумерного массива. Для битностей 4 и 8 доступно RLE-кодирование, которое может уменьшить их размер. Формат BMP также поддерживает встраивание данных в форматах JPEG и PNG. Но последнее скорее больше предназначено не для компактного хранения, а для обхода ограничений архитектуры GDI, которая не предусматривает прямую работу с изображениями отличных от BMP форматов. В последних версиях формата BMP также появились возможности по управлению цветом. В частности, можно указывать конечные точки, производить гамма-коррекцию и встраивать цветовые профили ICC.

Содержание

- 1 DIB и DDB
- 2 Внутреннее строение
  - 2.1 Общая структура
  - 2.2 BITMAPFILEHEADER
  - 2.3 BITMAPINFO
    - 2.3.1 16-битные информационные поля (версия CORE)
    - 2.3.2 32-битные информационные поля (версии 3, 4 и 5)
  - 2.4 Битность изображения (поле BitCount)
  - 2.5 Поле Compression
  - 2.6 Таблица цветов
  - 2.7 Маски для извлечения значений цветовых каналов
  - 2.8 Пиксельные данные
    - 2.8.1 Указание цвета и значения альфа-канала
    - 2.8.2 Двумерный массив
    - 2.8.3 RLE-кодирование
    - 2.8.4 Встраивание данных в форматах JPEG и PNG
  - 2.9 Разрешение изображения
  - 2.10 Цветовое пространство
    - 2.10.1 Конечные точки и значение гаммы
    - 2.10.2 Цветовой профиль
    - 2.10.3 Предпочтения при рендеринге
- 3 Пример программы на C
- 4 См. также

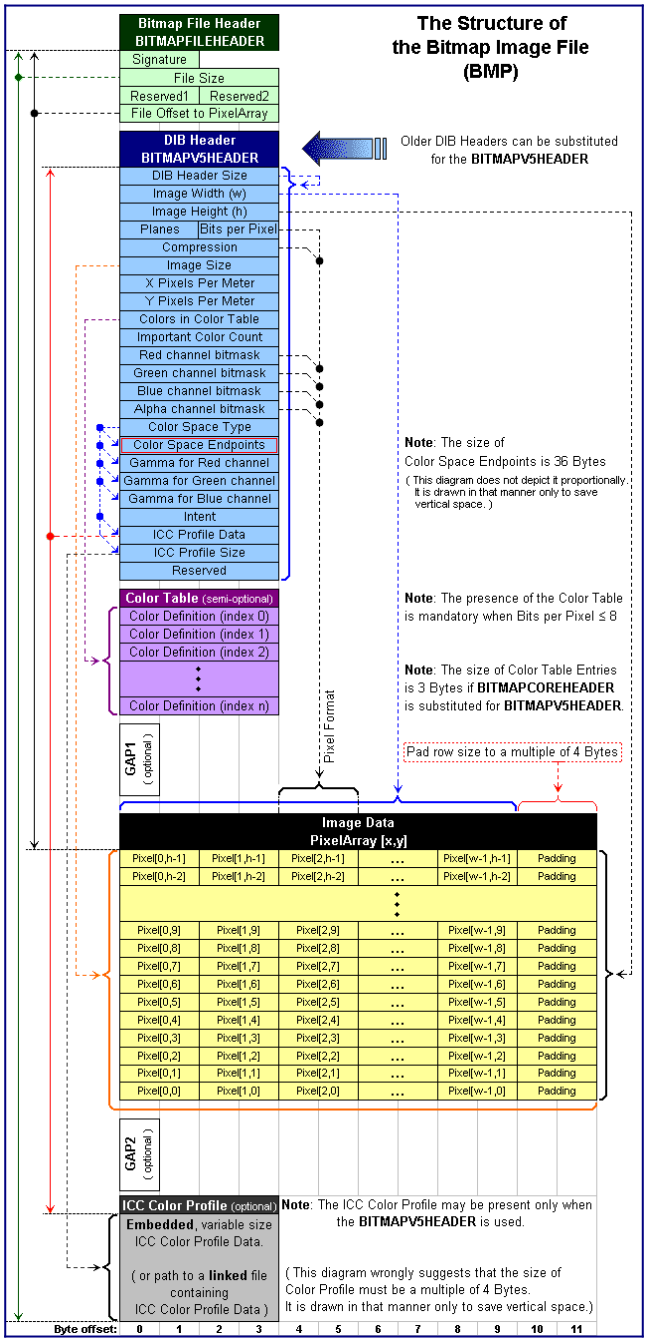
- 5 Примечания
- 6 Литература
  - 6.1 Microsoft (MSDN и SDK)
  - 6.2 Другие

DIB и DDB[править | править код]

При использовании формата DIB (англ. *Device Independent Bitmap*, аппаратно-независимый растр) программист может получить доступ ко всем элементам структур, описывающих изображение, при помощи обычного указателя. Но эти данные не используются для непосредственного управления экраном, так как они всегда хранятся в системной памяти, а не в специализированной видеопамяти. Формат пикселя в оперативной памяти может отличаться от того формата, который должен заноситься в видеопамять для индикации точки такого же цвета. Например, в DIB-формате может использоваться 24 бита для задания пикселя, а графический адаптер в этот момент может работать в режиме HiColor с цветовой глубиной 16 бит. При этом ярко-красная точка в аппаратно-независимом формате будет задаваться тремя байтами $0000FF_{16}$, а в видеопамяти — словом $F800_{16}$. При копировании картинки на экран система будет тратить дополнительное время на преобразование кодов цвета из 24-битного формата в формат видеобуфера.

Формат DDB (англ. *Device Dependent Bitmap*, аппаратно-зависимый растр) всегда содержит цветовые коды, совпадающие с кодами видеобуфера, но храниться он может как в системной, так и в видеопамяти. В обоих случаях он содержит только коды цвета в том формате, который обеспечит пересылку изображения из ОЗУ в видеопамять при помощи простого копирования[5].

Внутреннее строение[править | править код]



Структура файла BMP

Официальную информацию по формату BMP можно найти в MSDN или справке Microsoft Windows SDK (может идти в комплекте с некоторыми IDE). В файле WinGDI.h от компании Microsoft есть все объявления на языке C++, которые касаются данного формата. В данную статью же не были включены объявления типов, так как от этого она может быть слишком

громоздкой. К тому же официальные объявления некоторые разработчики могут посчитать неудобными и поэтому их востребованность сомнительна. Если вам потребуются оригинальные имена констант, структур, типов и их полей, то они все есть в тексте данной статьи.

Максимальный размер неделимых ячеек (исключая поля битовых структур): 32 бита и поэтому формат можно классифицировать как 32-битный. Исключением могут быть 64-битные пиксели, но значения их каналов можно обрабатывать и 16-битными словами. Порядок байт в 16-битных и 32-битных ячейках повсюду от младшего к старшему (little-endian). Целые числа записываются в прямом коде, со знаком — в дополнительном. Если сравнивать с аппаратными архитектурами, то порядок байт и формат чисел соответствует x86.

В данной статье для указания типов используются имена типов WinAPI как в документации Microsoft. Кроме специфических (описаны отдельно в тексте статьи) можно встретить четыре числовых типа:

- BYTE — 8-битное беззнаковое целое.
- WORD — 16-битное беззнаковое целое.
- DWORD — 32-битное беззнаковое целое.
- LONG — 32-битное целое со знаком.

В формате Windows Bitmap под структурами понимается блок с идущими подряд ячейками различного фиксированного размера, у которых есть условные имена (есть во многих языках программирования), а не что-то сложнее (например, поток команд произвольного размера).

У некоторых элементов формата указана версия Windows, начиная с которой он поддерживается. Речь идёт в первую очередь об основных библиотеках WinAPI таких как gdi32.dll, shell32.dll, user32.dll и kernel32.dll. Другие компоненты операционной системы (например, GDI+, .NET, DirectX) могут иметь другие более широкие возможности.

Общая структура[править | править код]

Данные в формате BMP состоят из трёх основных блоков различного размера:

1. Заголовок из структуры BITMAPFILEHEADER и блока BITMAPINFO. Последний содержит:
    1. Информационные поля.
    2. Битовые маски для извлечения значений цветовых каналов (опциональные).
    3. Таблица цветов (опциональная).
2. Цветовой профиль (опциональный).
3. Пиксельные данные.

При хранении в файле все заголовки идут с самого первого байта. Пиксельные данные могут находиться на произвольной позиции в файле (она указывается в поле OffBits структуры BITMAPFILEHEADER), в том числе и в удалении от заголовков. Опциональный цветовой профиль появился в версии 5 и он также может свободно располагаться, но его позиция указывается от начала BITMAPINFO (в поле ProfileData).

В оперативной памяти (например, при взаимодействии с WinAPI-функциями GDI) из заголовков исключается структура BITMAPFILEHEADER. При этом Microsoft рекомендует располагать цветовой профиль сразу за заголовками в едином блоке[6]. Пиксельные данные могут иметь произвольное расположение в памяти и их адрес указывается в параметрах процедур. В любом случае рекомендуется в памяти все блоки содержать по адресам кратным четырём: в заголовках присутствуют 32-битные ячейки, а к пиксельным данным такое требование указано в документации[7]. Это требование справедливо только для оперативной памяти: при хранении в файле его придерживаться не обязательно.

BITMAPFILEHEADER[править | править код]

BITMAPFILEHEADER — 14-байтная структура, которая располагается в самом начале файла. Обратите внимание на то, что с самого начала структуры сбивается выравнивание ячеек. Если для вас оно важно, то в оперативной памяти данный заголовок располагайте по чётным адресам, которые не кратны четырём (тогда 32-битные ячейки попадут на выравненные позиции).

Поз. (hex)	Размер (байты)	Имя	Тип WinAPI	Описание
00	2	bfType	WORD	Отметка для отличия формата от других (сигнатура формата). Может содержать единственное значение $4D42_{16}$/$424D_{16}$ (little-endian/big-endian).
02	4	bfSize	DWORD	Размер файла в байтах.
06	2	bfReserved1	WORD	Зарезервированы и должны содержать ноль.
08	2	bfReserved2	WORD	
0A	4	bfOffBits	DWORD	Положение пиксельных данных относительно начала данной структуры (в байтах).

Сигнатура формата при просмотре содержимого файла текстом в двоичном режиме выглядит как пара ASCII-символов «BM».

BITMAPINFO[править | править код]

BITMAPINFO в файле идёт сразу за BITMAPFILEHEADER. Адрес этого блока в памяти напрямую также передаётся некоторым функциям WinAPI (например, SetDIBitsToDevice или CreateDIBitmap). Кроме этого, этот же блок используется в форматах значков и курсоров Windows, но в данной статье этот момент не рассматривается (см. отдельные описания этих форматов). Данная структура является основной и описательной в формате BMP и поэтому когда просто упомянуто имя поля, то речь идёт о поле в данной структуре.

Блок BITMAPINFO состоит из трёх частей:

1. Структура с информационными полями.
2. Битовые маски для извлечения значений цветовых каналов (присутствуют не всегда).
3. Таблица цветов (присутствует не всегда).

Про битовые маски и таблицу цветов смотрите ниже в отдельных разделах. Здесь далее пойдёт описание структуры с информационными полями.

В момент написания данной статьи структура с информационными полями имела четыре версии[8]: CORE, 3, 4 и 5 (обозначения версий приведены условные в рамках данной статьи для краткости). Для каждой версии Microsoft объявила четыре отдельные структуры с разными именами полей. В данной статье при упоминании поля, которое присутствует в нескольких структурах, берётся общая для всех структур часть в конце имени (например, BitCount вместо bcBitCount, biBitCount, bV4BitCount или bV5BitCount). Версию структуры можно определить по первой 32-битной ячейке (WinAPI-тип DWORD), которая содержит её размер в байтах (беззнаковым целым). Версия CORE отличается от всех своей компактностью и использованием исключительно 16-битных беззнаковых полей. Остальные три содержат идентичные ячейки, к которым в каждой новой версии добавлялись новые.

Ниже представлена обзорная таблица по информационным структурам:

Версия	Размер (байты)	Имя структуры	Версия Windows 9x/NT[9]	Версия Windows CE/Mobile[10]	Примечания
CORE	12	BITMAPCOREHEADER	95/NT 3.1 и старше	CE 2.0/Mobile 5.0 и старше	Содержит только ширину, высоту и битность растра.
3	40	BITMAPINFOHEADER	95/NT 3.1 и старше	CE 1.0/Mobile 5.0 и старше	Содержит ширину, высоту и битность растра, а также формат пикселей, информацию о цветовой таблице и разрешении.
4	108	BITMAPV4HEADER	95/NT 4.0 и старше	не поддерживается	Отдельно выделены маски каналов, добавлена информация о цветовом пространстве и гамме.
5	124	BITMAPV5HEADER	98/2000 и старше	не поддерживается	Добавлено указание предпочтительной стратегии отображения и поддержка профилей ICC.

Из-за идентичности полей в версиях 3, 4 и 5 может показаться, что полем Size можно регулировать количество полей, убирая неиспользуемые. В действительности это не корректно, так как здесь размер играет роль версии (в версии CORE хоть и тоже идентичные поля, но другого размера и типа). Никто не гарантирует, что вам не могут попасться заголовки меньших или больших размеров с другим набором полей. Тем не менее, Adobe Photoshop может при сохранении файлов BMP записывать структуры информационных полей с размерами 52 и 56 байт. По сути это урезанная 4-я версия, которая содержат только битовые маски каналов (56 байт — версия с альфа-каналом).

16-битные информационные поля (версия CORE)[править | править код]

Обратите внимание на то, что здесь поля ширины и высоты содержат беззнаковые целые, в то время как 32-битные структуры хранят значения со знаком.

Позиция в файле (hex)	Позиция в структуре (hex)	Размер (байты)	Имя	Тип WinAPI	Описание
0E	00	4	bcSize	DWORD	Размер данной структуры в байтах, указывающий также на версию структуры (здесь должно быть значение 12).
12	04	2	bcWidth	WORD	Ширина (bcWidth) и высота (bcHeight) растра в пикселях. Указываются целым числом без знака. Значение 0 не документировано.
14	06	2	bcHeight	WORD	
16	08	2	bcPlanes	WORD	В BMP допустимо только значение 1. Это поле используется в значках и курсорах Windows.
18	0A	2	bcBitCount	WORD	Количество бит на пиксель (список поддерживаемых смотрите в отдельном разделе ниже).

32-битные информационные поля (версии 3, 4 и 5)[править | править код]

В таблице ниже поля представлены обзорно. Подробную информацию вы можете найти в разделах далее.

Позиция в файле (hex)	Позиция в структуре (hex)	Размер (байты)	Имя (версии 3/4/5)	Тип WinAPI	Описание
0E	00	4	biSize bV4Size bV5Size	DWORD	Размер данной структуры в байтах, указывающий также на версию структуры (см. таблицу версий выше).
12	04	4	biWidth bV4Width bV5Width	LONG	Ширина растра в пикселях. Указывается целым числом со знаком. Ноль и отрицательные не документированы.
16	08	4	biHeight bV4Height bV5Height	LONG	Целое число со знаком, содержащее два параметра: высота растра в пикселях (абсолютное значение числа) и порядок

					следования строк в двумерных массивах (знак числа). Нулевое значение не документировано.
1A	0C	2	biPlanes bV4Planes bV5Planes	WORD	В BMP допустимо только значение 1. Это поле используется в значках и курсорах Windows.
1C	0E	2	biBitCount bV4BitCount bV5BitCount	WORD	Количество бит на пиксель (список поддерживаемых смотрите в отдельном разделе ниже).
1E	10	4	biCompression bV4V4Compression[11] bV5Compression	DWORD	Указывает на способ хранения пикселей (см. в разделе ниже).
22	14	4	biSizeImage bV4SizeImage bV5SizeImage	DWORD	Размер пиксельных данных в байтах. Может быть обнулено если хранение осуществляется двумерным массивом.
26	18	4	biXPelsPerMeter bV4XPelsPerMeter bV5XPelsPerMeter	LONG	Количество пикселей на метр по горизонтали и вертикали (см. раздел «Разрешение изображения» данной статьи).
2A	1C	4	biYPelsPerMeter bV4YPelsPerMeter bV5YPelsPerMeter	LONG	
2E	20	4	biClrUsed bV4ClrUsed bV5ClrUsed	DWORD	Размер таблицы цветов в ячейках.
32	24	4	biClrImportant bV4ClrImportant bV5ClrImportant	DWORD	Количество ячеек от начала таблицы цветов до последней используемой (включая её саму).
Добавлены в версии 4					
Позиция в файле (hex)	**Позиция в структуре (hex)**	**Размер (байты)**	**Имя (версии 4/5)**	**Тип WinAPI**	**Описание**
36	28	4	bV4RedMask bV5RedMask	DWORD	Битовые маски для извлечения значений каналов: интенсивность красного, зелёного, синего и значение альфа-канала.
3A	2C	4	bV4GreenMask bV5GreenMask	DWORD	
3E	30	4	bV4BlueMask bV5BlueMask	DWORD	
42	34	4	bV4AlphaMask bV5AlphaMask	DWORD	
46	38	4	bV4CSType bV5CSType	DWORD	Вид цветового пространства.
4A	3C	36	bV4Endpoints bV5Endpoints	CIEXYZTRIPLE	Значение этих четырёх полей берётся во внимание только если поле CSType содержит 0 (LCS_CALIBRATED_RGB). Тогда конечные точки и значения гаммы для трёх цветовых компонент указываются в этих полях.
6E	60	4	bV4GammaRed bV5GammaRed	DWORD[12]	
72	64	4	bV4GammaGreen bV5GammaGreen	DWORD[12]	
76	68	4	bV4GammaBlue bV5GammaBlue	DWORD[12]	
Добавлены в версии 5					
Позиция в файле (hex)	**Позиция в структуре (hex)**	**Размер (байты)**	**Имя**	**Тип WinAPI**	**Описание**
7A	6C	4	bV5Intent	DWORD	Предпочтения при рендеринге растра.
7E	70	4	bV5ProfileData	DWORD	Смещение в байтах цветового профиля от начала BITMAPINFO (см. также раздел «Цветовой профиль» ниже в этой статье).
82	74	4	bV5ProfileSize	DWORD	Если в BMP непосредственно включается цветовой профиль, то здесь указывается его размер в байтах.
86	78	4	bV5Reserved	DWORD	Зарезервировано и должно быть обнулено.

Битность изображения (поле BitCount)[править | править код]

Поле BitCount содержит количество бит, которое приходится на каждый пиксель. Если там указано отличное от нуля значение, то это и есть реальная битность. Нулевое же содержимое поля BitCount указывается если пиксели хранятся в формате JPEG или PNG. Тогда действительная битность будет определяться уже этими форматами. Битности чисто BMP формата представлены в таблице ниже:

Бит	Байт	BITMAPINFO		RLE	Маски каналов	Поддержка программным обеспечением			
		CORE	3, 4, 5			Windows 9x и NT	Windows CE и Mobile	GDI+ и .NET	
1	⅛	Да	Да	Нет	Нет	Да	Да	Да	
2	¼	Да	Да	Нет	Нет	Нет	Да	Нет	
4	½	Да	Да	Да	Нет	Да	Да	Да	
8	1	Да	Да	Да	Нет	Да	Да	Да	
16	2	Нет	Да	Нет	Да	Да	Да	Да	
24	3	Да	Да	Нет	Нет	Да	Да	Да	
32	4	Нет	Да	Нет	Да	Да	Да	Да	
48	6	Нет	Да	Нет	Нет	Нет	Нет	Да	
64	8	Нет	Да	Нет	Нет	Нет	Нет	Да	

Примечания к таблице:

1. В колонке «BITMAPINFO» указана поддержка битностей только со стороны Microsoft.
2. Windows CE 1.0 и 1.01 поддерживает только битности 1 и 2[13].
3. GDI+ версии 1.0 16-битные цветовые каналы умеет только считывать, сразу переводя их в 8-битные[14].

В битностях 8 и ниже цвет пикселя указывается индексом в таблице цветов, в высших: непосредственным значением в цветовой модели RGB. Альфа-канал опционально может быть добавлен в битностях 16 и 32. В битности 64 он присутствует перманентно.

В таблице представлены только битности, которые документировала корпорация Microsoft. Сам формат не содержит никаких принципиальных ограничений на использование каких-либо не упомянутых здесь битностей.

1-битные BMP-файлы с чисто чёрным (сброшенный бит) и белым (установленный бит) цветами называют монохромными. Такие файлы обычно используются в качестве масок для других растров. Возможно вам постоянно попадались на глаза именно такие 1-битные растры. В действительности формат Windows Bitmap не накладывает никаких ограничений на то, какие цвета будут использоваться для каждого из значений бит.

Вы можете также встретить следующие названия битностей: CGA для двух бит, EGA для четырёх, HiColor (HighColor) для 16 бит, TrueColor для 24 и 32 бит с полупрозрачностью, DeepColor для больших битностей и возможно другие. Эти названия появились в период развития цветных растровых дисплеев и относятся больше к их глубине цвета. К моменту написания данной статьи (2014 год) такие названия уже давно не используются из-за сильного распространения 24-битных устройств (вместо этого просто указывается глубина цвета в битах или их количество). А BMP-файлы в меньшей битности сохраняются в большей степени не для отображения на CGA или EGA-устройствах, а для компактности по сравнению с 24-битными и 32-битными форматами если используется малое количество цветов.

Поле Compression[править | править код]

Для каждой группы битностей используются отдельные ограниченные значения Compression, но в совокупности их значения уникальны. Microsoft документировала следующие значения (в таблице указаны имена констант от этой корпорации):

Значение	Имя константы	Применимо к BitCount	Хранение пикселей	Знак Height	Версия Windows	
					9x/NT	CE
0	BI_RGB	любым кроме нуля	двумерный массив	+/−	95/NT 3.1	CE 1.0
1	BI_RLE8	8	RLE-кодирование	+	95/NT 3.1	(не подд.)
2	BI_RLE4	4	RLE-кодирование	+	95/NT 3.1	(не подд.)
3	BI_BITFIELDS	16 и 32	двумерный массив с масками цветовых каналов	+/−	95/NT 3.1	CE 2.0
4	BI_JPEG	0	во встроенном JPEG-файле	?/−[15]	98/2000	(не подд.)
5	BI_PNG	0	во встроенном PNG-файле	?/−[15]	98/2000	(не подд.)
6	BI_ALPHABITFIELDS	16 и 32	двумерный массив с масками цветовых и альфа-канала	+/−	(не подд.)	.NET 4.0

Таблица цветов[править | править код]

Таблица цветов является частью блока BITMAPINFO и может использоваться в двух случаях:

1. Она обязательно присутствует при битностях 8 и ниже, в которых цвет пикселей задаётся индексом ячейки из неё.

2. При битностях 8 и выше, в которых цвет указывается непосредственным значением, таблица присутствует если используется заголовок не CORE-версии, у которого поле ClrUsed содержит ненулевое значение. Здесь она задействуется уже для оптимизации цветов при работе с использующими палитры устройствами (как именно производится оптимизация в документации не сказано).

Позиция таблицы цветов указывается от его начала блока BITMAPINFO. По умолчанию она идёт сразу за информационной структурой (её беззнаковый размер в байтах можно прочитать из первого 32-битного поля). Но между структурой с полями и цветовой таблицей могут идти четырёхбайтные битовые маски для извлечения цветовых каналов (касается только битностей 16 и 32). Они там находятся только если используется информационная структура 3-й версии (Size = 40) и поле Compression содержит 3 (BI_BITFIELDS) или 6 (BI_ALPHABITFIELDS). Тогда к размеру информационных полей нужно прибавить 12 (при значении BI_BITFIELDS) или 16 байт (если указано BI_ALPHABITFIELDS)[16]. Получается 6 вариантов расположения таблицы:

Версия заголовка	Позиция (hex)		Примечания
	В файле	В BITMAPINFO	
CORE	1A	0C	маски каналов не поддерживаются
3	36	28	Compression не содержит 3 или 6
	42	34	Compression = 3 (BI_BITFIELDS)
	46	38	Compression = 6 (BI_ALPHABITFIELDS)
4	7A	6C	маски каналов встроены в информационные поля
5	8A	7B	

Количество ячеек в таблице определяется по полям BitCount и ClrUsed. При битностях 8 и ниже максимальное количество ячеек в таблице принимается за $2^{битность}$: 2 в однобитном растре, 4 в двухбитном, 16 в 4-битном и 256 в 8-битном. В данных битностях таблица всегда содержит это максимальное количество ячеек если используется заголовок версии CORE (Size = 12) или если в других версиях поле ClrUsed содержит 0. Во всех остальных случаях, независимо от битности, в таблице находится столько же ячеек, сколько указано в поле ClrUsed[17].

Сама же таблица представляет собой одномерный массив, который может содержать ячейки трёх типов:

1. 32-битная структура **RGBQUAD**. Применяется если в BITMAPINFO использована информационная структура версии 3, 4 или 5. В самой же структуре RGBQUAD указывается цвет в модели RGB в четырёх байтовых ячейках (все имеют WinAPI-тип BYTE): rgbBlue (синий), rgbGreen (зелёный), rgbRed (красный) и rgbReserved (зарезервирована и должна быть обнулена).
2. 24-битная структура **RGBTRIPLE**. Применяется если BITMAPINFO начинается со структуры BITMAPCOREHEADER. RGBTRIPLE состоит из трёх байтовых ячеек (WinAPI-тип BYTE), в которых указывается цвет в модели RGB: rgbtBlue (синий), rgbtGreen (зелёный) и rgbtRed (красный).
3. 16-битные индексы цветов (беззнаковые целые числа) в текущей логической палитре контекста устройства (системные объекты Windows GDI). Этот вид доступен только во время выполнения приложения. Формат BMP не поддерживает явное указание, что используется такая таблица и поэтому приложение само извещает WinAPI-функции об этом в специальных параметрах (как правило константой DIB_PAL_COLORS).

Во всей таблице могут быть задействованы не все ячейки и в поле ClrImportant помещается количество ячеек от начала таблицы до последней используемой (включая её саму). Содержимое 0 поля ClrImportant указывает на то, что используется вся таблица. Задействованные ячейки лучше размещать в самом начале таблицы и рекомендуется при этом отсортировать их по убыванию степени важности (на случай если придётся уменьшить их количество).

Маски для извлечения значений цветовых каналов[править | править код]

Если битность изображения 16 или 32, то могут быть указаны 32-битные маски для извлечения цветовых каналов. Это связано с тем, что 16 не кратно трём и поэтому биты могут быть распределены разными способами. В 32-битных изображениях из-за удобства используют 8-битные каналы и поэтому поддержка для них может показаться избыточной. В действительности здесь маска даёт возможность включить/отключить альфа-канал или установить удобный вам порядок следования компонент, а не только регулировать их разрешение. При применении масок ячейка пикселя считывается целиком как соответствующее машинное слово в litte-endian.

Наличие битовых масок зависит от версии информационных полей структуры BITMAPINFO и поля Compression в ней. Для версии CORE произвольные маски указать нельзя, так как там отсутствует поле Compression и отдельные поля масок. В остальных версиях цветовые маски задействуются если Compression содержит 3 (BI_BITFIELDS). Маска альфа-канала используется всегда в версиях 4 и 5. Так как Windows CE не поддерживает эти две версии, в которых для неё есть специальное поле, для третьей версии было введено значение 6 (BI_ALPHABITFIELDS) поля Compression, которое добавляет сразу цветовые маски и маску альфа-канала (поддерживается начиная с Windows CE .NET 4.0).

Положение битовых масок фиксировано независимо от версии заголовка: 36_h во всём файле или 28_h от начала блока BITMAPINFO. В версиях 4 и 5 на этом месте располагаются предназначенные специально для них поля. В версии же 3 битовые маски должны располагаться сразу за информационными полями и таким образом они точно попадают на позиции соответствующих полей в старших версиях. Обратите внимание на то, что в третьей версии при наличии масок сдвигается таблица цветов на 12 или 16 байт вперёд, располагаясь сразу за ними. 4-байтные маски цветов следуют в порядке красная, зелёная, синяя. Маска альфа-канала располагается уже за ними.

В документации Microsoft к битовым маскам применяется только одно обязательное требование: каждая маска должна быть непрерывной. Про случай пересечения масок там сказано, что желательно этого не делать[18]. Microsoft также говорит о том, что не обязательно задействовать все биты пикселя[19]. Какие-либо требования к содержимому неиспользуемых бит отсутствуют.

Обратите внимание на то, что никто не гарантирует, что могут быть использованы маски шире 8 бит. И ничего не сказано про случай, когда у какого-либо канала будет нулевая маска (например, когда он действительно не используется). Здесь возможна ситуация когда нулевые маски будут у всех компонентов и останется один альфа-канал (который при этом может

занять все биты). Нулевую маску цветового канала можно трактовать двумя способами: его значение принимается за ноль или же при прорисовке пиксели этого канала не затрагиваются. Если взять первый вариант интерпретации с единственным альфа-каналом, то альфа-канал по сути будет задавать степень зачернения пикселя. Кроме неопределённых вариантов есть также и интересный. Так как пересечения не запрещены, то можно все каналы выставить на одну позицию и тем самым получить Grayscale.

Некоторое ПО имеет ограниченный набор поддерживаемых битовых масок. В таблице ниже приведены доступные варианты в таких ограниченных средах:

Битность	*	Значения масок (hex)					Поддержка в ПО	
		Красный	Зелёный	Синий	Альфа	Неиспользуемые	Windows 9x[20]	GDI+[21] и .NET[22]
16	(a)	7C00	03E0	001F	0000	8000	Да	Да
		7C00	03E0	001F	8000	0000	Нет	Да
		F800	07E0	001F	0000	0000	Да	Да
	(b)	FFFF	FFFF	FFFF	0000	0000	Нет	Да
32	(a)	00FF:0000	0000:FF00	0000:00FF	0000:0000	FF00:0000	Да	Да
		00FF:0000	0000:FF00	0000:00FF	FF00:0000	0000:0000	Нет	Да

Примечания к таблице:

(a) Эти наборы используются по умолчанию в битностях 16 и 32, если маски для извлечения цветов не заданы.

(b) Данный набор масок по своей сути реализует 16-битный Grayscale.

Пиксельные данные[править | править код]

В файле положение пиксельных данных можно узнать из поля OffBits структуры BITMAPFILEHEADER. Во время выполнения приложение хранит адрес пиксельных данных там, где удобней. В документации Microsoft также упоминаются так называемые упакованные (англ.*packed*) битмапы, которые указываются одним адресом блока BITMAPINFO. У таких битмапов пиксельные данные следуют сразу за заголовком (включая помимо информационных полей битовые маски и таблицу цветов)[23].

Размер пиксельных данных в байтах записывается в поле SizeImage структуры BITMAPINFO. Туда записываются именно «сырой» размер того непрерывного блока, который содержит данные для формирования пикселей (независимо от формата), а не какой-нибудь распакованный. По умолчанию это поле обязательно должно содержать актуальное значение, так как по нему можно точно узнать сколько именно байт нужно считать из файла для получения пикселей. Тем не менее, допустимо содержать в этом поле ноль при хранении пикселей двумерными массивами (когда поле Compression содержит значение 0 (BI_RGB), 3 (BI_BITFIELDS) или 6 (BI_ALPHABITFIELDS)[24]). Тогда размер пикселей при необходимости можно относительно быстро рассчитать исходя из битности, ширины и высоты растра.

В формате Windows Bitmap доступно три способа хранения пикселей (см. также раздел «Поле Compression» данной статьи):

1. Двумерный массив.
2. RLE-кодирование (только для битностей 4 и 8).
3. В форматах JPEG или PNG.

В подразделах далее отдельно описан каждый из них.

Указание цвета и значения альфа-канала[править | править код]

Для указания цвета при хранении в формате BMP независимо от способа задания используются только беззнаковые целые числа. Сам же цвет пикселя может задаваться двумя способами:

1. Индексом в таблице цветов (при битностях 8 и ниже).
2. Непосредственным значением в цветовой модели RGB (при битностях выше 8).

Второй целесообразно использовать когда набор цветов довольно большой или непредсказуем (например, во время обработки изображения). Первый же способ обеспечивает как компактную компоновку при малом наборе цветов, так и некоторое удобство в управлении используемыми цветами (достаточно изменить значение цвета в палитре). В самой таблице цветов указываются или 16-битные беззнаковые индексы в системной палитре (см. раздел «Таблица цветов» в данной статье), или же в RGB как в пикселе, но исключительно 8-битными значениями каналов.

Индекс в таблице цветов — это номер ячейки в ней от начала таблицы (используется непрерывная нумерация начиная с нуля). Для каждой битности максимальный индекс принципиально ограничен значением $2^{битность} - 1$. В действительности же он ограничен ещё и количеством элементов в таблице (подробности в разделе «Таблица цветов» данной статьи). Microsoft не документировала поведение в случае когда указывается индекс за пределами таблицы, но GDI в этом случае берёт чёрный цвет.

В битностях выше 8 цвет пикселя указывается непосредственно в цветовой модели RGB: отдельно указывается уровень красного цвета, зелёного и синего. Нулевое значение любого из каналов означает полное отсутствие соответствующего оттенка, а максимальное: полное его присутствие. Разрешение же значений каналов переменное и в каждой битности оно своё (конкретные значения смотрите в разделе про хранение пикселей в двумерном массиве этой статьи). При этом в битностях 16 и 32 может быть задано не только произвольное разрешение, но и индивидуальное для каждого канала (например, 5 бит для красной и синей, но 6 бит для зелёной). Несмотря на большое количество вариантов задания

разрешения значений, в документации Microsoft не сказано как производить изменение разрешения значения. Из-за этого у разных производителей программного обеспечения могут получаться различные результаты при смене битности.

При непосредственном задании цвета пикселя кроме значений RGB формат Windows Bitmap опционально позволяет ещё задать значения **альфа-канала**. В плане битности и кодировании значений он идентичен цветовым каналам: у него произвольная битность и используются беззнаковые целые. Что же касается сопоставления значений, то ноль соответствует полной прозрачности, а максимальное доступное число — полной заполненности.

Двумерный массив[править | править код]

В двумерном массиве можно хранить пиксели любой битности. При таком способе хранения поле Compression содержит значение 0 (BI_RGB), 3 (BI_BITFIELDS) или 6 (BI_ALPHABITFIELDS). Если используется заголовок версии CORE, то пиксели в любом случае хранятся только двумерным массивом.

В данной компоновке пиксели растра записываются однопиксельными горизонтальными полосками, которые Microsoft в своей документации часто называет «*scans*» (в русском языке наиболее близкое слово: *строки*). В памяти эти ряды записываются по-порядку, но при положительном Height: начиная с самого нижнего (англ. *bottom-up bitmap*), а при отрицательном: с самого верхнего (англ. *top-down bitmap*). Внутри каждого горизонтального ряда пиксели записываются строго только от левого к правому. Пиксели меньше 8 бит размещаются в байтах, заполняя биты от старших к младшим, в результате чего шестнадцатиричные/двоичные числовые значения пикселей более похожи на выводимое изображение. Если битность 16 или 32, то обработка осуществляется цельными машинными словами аналогичного размера с порядком бит от младшего к старшему (little-endian). **Ряды, независимо от размера ячеек, обязательно должны дополняться нулями до кратного четырём байтам размера**[7]. Из-за этого, при некратной ширине изображения, в конце рядов могут оказываться неиспользуемые биты или целые байты. Но благодаря гарантированной кратности размера ряда, обработку можно производить 8-, 16- или 32-битными машинными словами, по выбору. И у Microsoft ещё прослеживается следующая тенденция в битностях больше 8: компонент Blue (синий цвет) размещается в младших битах/первых байтах, Green (зелёный) в последующих, а Red (красный) старше/дальше всех, и если есть альфа-канал, то он находится в самых старших битах/последних байтах.

Диаграмма ниже отображает расположение пикселей в **битностях меньше 8**:

Биты	7	6	5	4	3	2	1	0
1 бит	0	1	2	3	4	5	6	7
2 бита	0		1		2		3	
4 бита	0				1			

В **битностях 16** и **32** пиксели обрабатываются машинными словами аналогичного размера (предполагается порядок байт little-endian), которым применяются битовые маски каналов. Если индивидуальные битовые маски не заданы, то структура будет следующая. При 16 бит на каждый канал отводится по 5 бит. Синий располагается в младших битах (маска $001F_{16}$), зелёный на позиции 5 (маска $03E0_{16}$), красный: начиная с 10-го бита (маска $7C00_{16}$), а старший оставшийся бит 15 не используется. Если используется битность 32, то по умолчанию на каждый канал отводится уже по байту (8 бит). Компоненты располагаются аналогично: синий в младших битах (маска $0000{:}00FF_{16}$), зелёный начиная с бита 8 (маска $0000{:}FF00_{16}$), красный начинается с бита 16 (маска $00FF{:}0000_{16}$), а старший байт не используется (он используется в качестве альфа-канала только если прямо это показать)[25]. Так как предполагается чтение в порядке байта little-endian, то если читать значения из памяти по-байтово, они будут располагать в таком же порядке (синий будет идти первым).

При **битности 24** на каждый канал приходится по байту, а в **битностях 48** и **64**: по 16-битному машинному слову. Во всех трёх случаях в памяти цветовые компоненты идут в порядке: синий, зелёный, красный. В 64-битных BMP за цветами дополнительно следует 16-битный альфа-канал. Если захотите 64-битный пиксель обработать цельным машинным словом, то в little-endian синий окажется в младших 16 битах, а альфа-канал: в старших. Зелёный, соответственно, будет рядом к красным, а синий — рядом с альфа. И можно заметить что в 24 битах формат пикселя соответствует структуре RGBTRIPLE из таблицы цветов.

RLE-кодирование[править | править код]

Применение RLE-кодирования компанией Microsoft документировано только для битностей 4 и 8. При её использовании в BITMAPINFO поле Compression должно содержать 2 (BI_RLE4) при битности 4 или 1 (BI_RLE8) с восьмибитными пикселями. Высота растра при этом должна быть указана положительным числом.

В формате Windows Bitmap RLE-кодирование можно сравнить с прорисовкой простыми командами. Прорисовка начинается с *левого нижнего* пикселя (будьте внимательней: в растрах в целом привычней может быть верхний левый) и осуществляется вправо и вверх. Пиксели за пределами размера растра не прорисовываются (об этом в документации не сказано, но GDI проявляет такое поведение). Инструкции RLE позволяют прерывать прорисовку горизонтали, всего изображения, а также перемещать курсор прорисовки на другую позицию. В результате некоторые пиксели могут оказаться не зарисованными. Формат не предусматривает цвета для незарисованных пикселей, в документации ничего про них не сказано, а системные функции просто не трогают незакрашенные пиксели. Так как нет веской причины в начале закрашивать прямоугольник под растром неопределённым цветом, то можно говорить о том что RLE косвенно поддерживает прозрачность.

Формирование изображения при RLE-кодировании осуществляется командами. Каждая команда обязательно должна начинаться с чётного адреса (выравнена на 16-битную границу). Существует пять команд, которые определяются парой байт:

Байт 1 (hex)	**Байт 2 (hex)**	**Описание**
01..FF	00..FF	Начиная с текущей позиции и далее вправо прорисовать столько пикселей, сколько указано в первом байте. Значения для пикселей берутся из второго байта. В 8-битных BMP весь байт целиком является значением. В 4-битных из него по-очереди берётся сначала старший, а потом младший ниббл (четвёрка бит).
00	00	Переместить курсор в начало (самое лево) следующей (верхней) горизонтали.

00	01	Прекратить прорисовку (достигнут конец).
00	02	Переместить курсор вправо и вверх на указанные в следующих далее двух байтах значения. В первом следующем байте содержится значение для горизонтального сдвига, а в следующем — для вертикального. Оба значения: целые беззнаковые числа (влево и вниз сдвинуть нельзя).
00	03..FF	С текущей позиции и далее вправо зарисовать пиксели значениями, которые идут после этой пары байт. Во втором байте команды содержится количество пикселей, которое нужно закрасить (именно пикселей, а не байт). В 8-битном растре берётся поток байт как есть. В 4-битном считывается уже нибблы: старшие 4 бита из байта для первого пикселя, младшие 4 бита — для следующего, и так из последующих байт. Данный поток может закончиться нечётным количеством байт, а команды требуют 16-битного выравнивания. Если это произошло, то дописывается дополнительный байт (его содержимое значения не имеет).

Когда достигается правый край горизонтали, то на следующую перевод не производится. Поэтому нужно специально вставлять команду окончания ряда. И как видно из таблицы, набор команд не позволяют двигаться вниз или вернуться назад в горизонтали. Поэтому можно прекращать прорисовку если будет достигнут верхний край.

Встраивание данных в форматах JPEG и PNG[править | править код]

Начиная с версий Windows 98/ME и 2000/XP системные функции позволяют хранить пиксели в форматах JPEG и PNG. Про степень поддержки этих двух форматов системой ничего не известно.

Для встраивания JPEG или PNG нужно в BITMAPINFO обнулить поле BitCount, а в Compression указать значение 4 (BI_JPEG) или 5 (PI_PNG). Значение поля SizeImage в данном случае будет равно размеру JPEG или PNG-файла, который встраивается на место пиксельных данных как есть. Ширина же с высотой в заголовке указываются уже для раскодированного изображения. Про знак поля Height именно для этого случая в документации напрямую ничего не сказано, но судя по всему нужно записывать отрицательное значение[15].

Разрешение изображения[править | править код]

Для сопоставления безразмерных пикселей с материальными размерами используются поля XPelsPerMeter и YPelsPerMeter. В этих полях целым числом указывается сколько в данном изображении пикселей приходится на один линейный метр, отдельно по горизонтали (XPelsPerMeter) и вертикали (YPelsPerMeter). Microsoft объявила эти два поля числовым типом со знаком, но в документации ничего не сказано про отрицательные значения. Про значение ноль также ничего не сказано, но логичней его принимать за неопределённое разрешение, когда оно неизвестно или не имеет значения.

Разрешение часто указывается с привязкой не к метрическим размерностям, а в точках на дюйм (DPI/PPI). Для перевода туда и обратно дюйм принимается равным 25,4 мм (английский дюйм). Математические формулы для перевода пиксели/дюйм (PPI) в пиксели/метр (PPM) и наоборот:

Если интересует точный целочисленный перевод, то выходят следующие выражения:

PPM = (*PPI* * 5000 + 64) / 127

PPI = (*PPM* * 127 + 2500) / 5000

После них округление будет произведено к ближайшему целому. Если хотите округление вниз, то не прибавляйте половину делителя. Если хотите вверх, то прибавляйте уменьшенный на единицу делитель.

Ниже представлены заранее вычисленные значения PPM для некоторых PPI/DPI:

- 96 ppi ≈ 3780 ppm (для мониторов у Microsoft)
- 72 ppi ≈ 2835 ppm (Apple для мониторов)
- 150 dpi ≈ 5906 ppm
- 300 dpi ≈ 11811 ppm
- 600 dpi ≈ 23622 ppm

Цветовое пространство[править | править код]

В информационных полях основным полем задающим цветовое пространство является поле CSType. Допустимые его значения приведены в таблице ниже:

Значение		**Версия BITMAPINFO[26]**	**Имя константы**	**Описание**
Hex	**Текст**			
0	(нет)	4	LCS_CALIBRATED_RGB	Корректировка исходя из значений Endpoints, GammaRed, GammaGreen и GammaBlue.
73524742	'sRGB'	4	LCS_sRGB	Используется цветовое пространство sRGB.
57696E20	'Win '[27]	4	LCS_WINDOWS_COLOR_SPACE	Системное пространство по умолчанию (sRGB).
4C494E4B	'LINK'	5	PROFILE_LINKED	Цветовой профиль в другом файле.
4D424544	'MBED'	5	PROFILE_EMBEDDED	Включённый в данный файл цветовой профиль.

Microsoft объявила значения констант не числовыми значениями, а текстовыми из четырёх символов[28]. В данном случае коды символов формируют байты 32-битного значения (ASCII-код самого правого символа является младшим байтом, код самого левого — старшим). При просмотре двоичного содержимого файла в виде текста такие значения в кодировке ASCII будут отображаться задом-наперёд (например, «KNIL», а не «LINK»).

Конечные точки и значение гаммы[править | править код]

Формат Windows Bitmap позволяет производить цветокоррекцию указанием конечных точек для красного, зелёного и синего цветов, а также значений гаммы. Для этого поле CSType должно содержать значение 0 (LCS_CALIBRATED_RGB). Корректирующие же значения записываются в поля Endpoints, GammaRed, GammaGreen и GammaBlue (при других значениях CSType эти четыре поля игнорируются).

36-байтное поле EndPoints является структурой CIEXYZTRIPLE, которая состоит из трёх полей ciexyzRed (конечная точка красного), ciexyzGreen (точка зелёного) и ciexyzBlue (синяя). Эти три поля в свою очередь также являются структурами CIEXYZ с тремя полями ciexyzX, ciexyzY и ciexyzZ типа FXPT2DOT30. PXPT2DOT30 — это 32-битное беззнаковое число с фиксированной запятой, у которого 2 старших бита отводятся под целую часть, а 30 младших — под дробную.

Значение гаммы записывается в соответствующие поля для каждого цветового канала отдельно: GammaRed (красный), GammaGreen (зелёный) и GammaBlue (синий). В объявлении информационных структур Microsoft указала у данных полей тип DWORD. В это же время в файле WinGDI.h есть более подходящее объявление типа FXPT16DOT16 (на основе типа long), который представляет собой 32-битное беззнаковое число с дробной частью в младших 16 битах и целой — в 16 старших. Следует отметить, что в MSDN на страницах про структуры BITMAPV4HEADER и BITMAPV5HEADER только это и сказано. В статье же про структуру LOGCOLORSPACE сказано, что у неё в аналогичных полях должны быть обнулены старший и младший байт (по сути вместо формата 16.16 используется формат 8.8, который располагается в середине 32-битной ячейки)[29].

Ниже приведены значения указанных выше четырёх полей в соответствии с цветовым пространством sRGB[30]:

Поле	Значение	
	Дробное	Hex
EndPoints.ciexyzRed.ciexyzX	0,64	28F5C28F
EndPoints.ciexyzRed.ciexyzY	0,33	151EB852
EndPoints.ciexyzRed.ciexyzZ	0,03	01EB851F
EndPoints.ciexyzGreen.ciexyzX	0,30	13333333
EndPoints.ciexyzGreen.ciexyzY	0,60	26666666
EndPoints.ciexyzGreen.ciexyzZ	0,10	06666666
EndPoints.ciexyzBlue.ciexyzX	0,15	0999999A
EndPoints.ciexyzBlue.ciexyzY	0,06	03D70A3D
EndPoints.ciexyzBlue.ciexyzZ	0,79	328F5C29
GammaRed GammaGreen GammaBlue	2,20	0002199A[31]

Цветовой профиль[править | править код]

В файле BMP при необходимости может быть указан цветовой профиль как непосредственным включением, так и ссылкой на другой файл. Профили появились в пятой версии BMP, и пока только здесь есть специальные для них поля. Поддерживаются же цветовые профили только в формате ICC[32][33].

При использовании цветовых профилей в первую очередь нужно указать следующие значения поля CSType:

- $4C494E4B_{16}$ (PROFILE_LINKED) — если используется профиль в другом файле.
- $4D424544_{16}$ (PROFILE_EMBEDDED) — если профиль непосредственно встраивается в BMP.

В любом случае в поле ProfileData указывается смещение профиля в байтах от начала блока BITMAPINFO. Если профиль встроенный, то в ProfileSize нужно указать его размер в байтах (если он подключаемый, то это поле должно быть обнулено). Независимо от варианта, Microsoft рекомендует размещать профиль при хранении в файле за пиксельными данными, а в оперативной памяти при взаимодействии с WinAPI-функциями: сразу за заголовком[34].

Формат ICC в своём заголовке использует преимущественно 32-битные или кратные этому размеру ячейки[35]. Исходя из этого, если профиль непосредственно включается в BMP, то в оперативной памяти его рекомендуется хранить по кратному четырём байтам адресу.

Когда профиль внешний, то вместо его содержимого в BMP размещается текстовая строка с путём к файлу. Он обязательно должен быть в однобайтовой кодировке Windows 1252 (стандартная кодировка для западноевропейских языков) и заканчиваться нулевым байтом. Про разделители компонентов пути в документации ничего не сказано и поэтому скорее всего можно использовать как левые слэши «\», так и «правые» «/». Путь же может быть как относительным, так и полным и сетевым[34]. И так как в указании пути используется однобайтовая кодировка, то эту строку в оперативной памяти выравнивать не обязательно.

Предпочтения при рендеринге[править | править код]

Предпочтения при рендеринге (англ. *rendering intents*) были введены Международным концорциумом по цвету (International Color Consortium) и определяют приоритеты в случае когда при переходе из цветового подпространства, поддерживаемого одним устройством (англ. *gamut*), в подпространство другого, в изображении использованы цвета, отсутствующие в целевом. Также есть определение от ICC, которое определяется предпочтения при рендеринге как стиль сопоставления цветовых значений из одного описателя изображения в другое (оригинал на английском языке: *«style of mapping colour values from one image description to another»*)[36]. Корпорация Microsoft включила в формат BMP специальное поле Intent, которое может принимать значения полностью по спецификации ICC. Поэтому за подробной информации обращайтесь к документации консорциума, последнюю версию которой можно скачать с сайта color.org[37]. У Microsoft же эти предпочтения коротко описаны в статье «Rendering Intents» на MSDN.

Предпочтение указывается в поле Intent блока BITMAPINFO и доступны только с 5-й версией информационных полей. Значения же могут быть следующими:

Значение	Имя константы для BMP	Название ICC	Название Microsoft	Константа из файла Icm.h	Константа для DEVMODE
1	LCS_GM_BUSINESS	saturation	Graphic	INTENT_SATURATION (2)	DMICM_SATURATE (1)
2	LCS_GM_GRAPHICS	media-relative colorimetric	Proof	INTENT_RELATIVE_COLORIMETRIC (1)	DMICM_COLORIMETRIC (3)
4	LCS_GM_IMAGES	perceptual	Picture	INTENT_PERCEPTUAL (0)	DMICM_CONTRAST (2)
8	LCS_GM_ABS_COLORIMETRIC	ICC-absolute colorimetric (relative colometric)	Match	INTENT_ABSOLUTE_COLORIMETRIC (3)	DMICM_ABS_COLORIMETRIC (4)

Microsoft для данной характеристики объявила как минимум три набора констант, которые отличаются своими значениями и используются в разных местах[38]. Здесь они приведены на случай если вам нужно будет быстро их сопоставить. Значения констант с префиксом «INTENT_» полностью совпадают с теми значениями, которые используются в файлах профилей ICC[39]. Константы с префиксом «DMICM_» объявлены в файле WinGDI.h для структуры DEVMODE. Константы «LCS_GM_», которые используются в BMP, объявлены там же и предназначены в первую очередь для структуры LOGCOLORSPACE. Есть также названия для свойств принтеров. Они аналогичны тем, что в колонке «Название Microsoft», но с «Graphics» и «Pictures».

За значение по умолчанию, которое в первую очередь подходит для фотографий и картинок, можно принимать 4 (LCS_GM_IMAGES). В таком качестве его рекомендует как Microsoft[40], так и ICC[41].

Пример программы на C[править | править код]

Следующая программа открывает 24 битный BMP файл в окне XWindow, глубина цвета должна составлять 32 бита, на меньшей цветопередаче не работает, так как это усложняет пример:

```
/* Компилируется строкой: cc -o xtest xtest.c -I/usr/X11R6/include -L/usr/X11R6/lib -lX11 -lm  */
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xatom.h>
#include <X11/keysym.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
#include <math.h>
#include "bitmap.h" /* Здесь определения заголовков BMP как было описано выше в этой статье (структуры должны быть упакованы на 2 байта!) */

static XImage *CreateImageFromBuffer(Display*, unsigned char *, int, int);

main(int argc, char *argv[])
{
    Display *dis;
    Window win;/* Наше окно */
    XEvent event;/* События */
    GC gc;/* Графический контекст */

    XImage *image;
    int  n, width, height, fd, size;
    unsigned char *data;
    BITMAPFILEHEADER bmp;
    BITMAPINFOHEADER inf;
    char* buf;

    if (argc < 2) {
        perror("use: xtest file.bmp\n");
        exit(1);
    }

  if ((fd = open(argv[1], O_RDONLY)) == -1) {
        printf("Error open bitmap\n");
        exit(1);
    }

  read(fd, &bmp, sizeof(BITMAPFILEHEADER));
  read(fd, &inf,sizeof(BITMAPINFOHEADER));

  width = inf.biWidth;
  height = inf.biHeight;

    if ((dis = XOpenDisplay(getenv("DISPLAY"))) == NULL) {
        printf("Can't connect X server:%s\n", strerror(errno));
        exit(1);
    }

    win = XCreateSimpleWindow(dis, RootWindow(dis, DefaultScreen(dis)), 0, 0, width, height, 5,
                BlackPixel(dis, DefaultScreen(dis)), WhitePixel(dis, DefaultScreen(dis)));

    XSetStandardProperties(dis, win, argv[1], argv[0], None, argv, argc, NULL);
```

```
    gc = DefaultGC(dis, DefaultScreen(dis));

 /* Иногда в структуре это место не заполнено */
    if(inf.biSizeImage == 0)  {
    /* Вычислим размер */
     size = width * 3 + width % 4;
     size = size * height;
    } else {
      size = inf.biSizeImage;
     }

    buf = malloc(size);
    if(buf == NULL) {
        perror("malloc");
        exit(1);
    }
    printf("size =%d байтов выделено\n", size);

     /* Сместимся на начало самого изображения */
    lseek(fd, bmp.bfOffBits, SEEK_SET);

    /* Читаем в буфер */
    n = read(fd, buf, size);
    printf("size =%d байт прочитано\n", n);

   image = CreateImageFromBuffer(dis, buf, width, height);

   /* Удалим буфер - он нам больше не нужен */
   free(buf);

    XMapWindow(dis, win);
    XSelectInput(dis, win, ExposureMask | KeyPressMask);
    while (1) {
        XNextEvent(dis, &event);
        if (event.xany.window == win) {
            switch (event.type) {
            case Expose:
                XPutImage(dis, win, gc, image, 0, 0, 0, 0, image->width, image->height);
                break;

            case KeyPress:
                if (XLookupKeysym(&event.xkey, 0) == XK_q) {
                    XDestroyImage(image);
                    XCloseDisplay(dis);
                    close(fd);
                    exit(EXIT_SUCCESS);
                }
                break;

            default:
                break;
            }
        }
    }
}

/* Создает Ximage из файла BMP, так как изображение BMP хранится первернутым
 * и зеркальным-в цикле это исправляется */
XImage *CreateImageFromBuffer(Display * dis, unsigned char *buf, int width,  int height)
{
    int depth, screen;
    XImage *img = NULL;
    int i, j;
    int numBmpBytes;
    size_t numImgBytes;
    int32_t *imgBuf;
    int ind = 0;
    int line;
    int temp;
    int ih, iw; /* Номера строки и столбца для отражения  */
    int new_ind; /* Новый индекс */

    screen = DefaultScreen(dis);
    depth = DefaultDepth(dis, screen);
    temp = width * 3;
    line = temp + width % 4; /* Длина строки с учетом выравнивания */
    numImgBytes = (4 * (width * height));
    imgBuf = malloc(numImgBytes);

    /* Размер, отведенный на BMP в файле с учетом выравнивания */
    numBmpBytes = line * height;
    for (i = 0; i < numBmpBytes; i++) {
        unsigned int r, g, b;

        /* Пропускаем padding */
        if (i >= temp && (i % line) >= temp)
            continue;

        b = buf[i];
```

```
        i++;
        g = buf[i];
        i++;
        r = buf[i];


        /* Вычисляем новый индекс для отражения по вертикали */
        iw = ind % width;
        ih = ind / width;
        new_ind = iw + (height - ih - 1) * width;

        imgBuf[new_ind] = (r | g << 8 | b << 16) << 8;
        ind++;
    }

    img = XCreateImage(dis, CopyFromParent, depth, ZPixmap, 0, (char *) imgBuf, width, height, 32, 0);
    XInitImage(img);

    /* Порядок битов и байтов на PC должен быть таким */
    img->byte_order = MSBFirst;

    img->bitmap_bit_order = MSBFirst;
    return img;
}
```