

HD Photo

Photographic Still Image File Format

*This file format is also known under the name
Windows Media™ Photo*

Device Porting Kit Specification

Copyright © 2005-2006 Microsoft Corporation. All rights reserved. Any use, distribution or public discussion of, and any feedback related to these materials are subject to the terms of the attached license.

Windows Media™ is a registered trademark of Microsoft Corporation. All rights reserved.

Version	1.0
Status	RTM

Microsoft Corporation Technical Documentation License Agreement for the specification "HD Photo Device Porting Kit"

READ THIS! THIS IS A LEGAL AGREEMENT BETWEEN MICROSOFT CORPORATION ("MICROSOFT") AND THE RECIPIENT OF THE ABOVE REFERENCED MATERIALS, WHETHER AN INDIVIDUAL OR AN ENTITY ("YOU"). IF YOU HAVE ACCESSED THIS AGREEMENT IN THE PROCESS OF DOWNLOADING THESE MATERIALS ("MATERIALS") FROM A MICROSOFT WEB SITE, BY CLICKING "I ACCEPT", DOWNLOADING, USING OR PROVIDING FEEDBACK ON THE MATERIALS, YOU AGREE TO THESE TERMS. IF THIS AGREEMENT IS ATTACHED TO MATERIALS, BY ACCESSING, USING OR PROVIDING FEEDBACK ON THE ATTACHED MATERIALS, YOU AGREE TO THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS, YOU ARE NOT AUTHORIZED TO ACCESS, DOWNLOAD, USE OR REVIEW THE MATERIALS.

For good and valuable consideration, the receipt and sufficiency of which are acknowledged, You and Microsoft agree as follows:

1. You may review these Materials only (a) as a reference to assist You in planning and designing Your product, service or technology ("Product") to interface with a Microsoft product, specification, service or technology ("Microsoft Product") as described in these Materials; and (b) to provide feedback on these Materials to Microsoft. All other rights are retained by Microsoft; this Agreement does not give You rights under any Microsoft patents. You may not (i) duplicate any part of these Materials, (ii) remove this Agreement or any notices from these Materials, or (iii) give any part of these Materials, or assign or otherwise provide Your rights under this Agreement, to anyone else.
2. These Materials may contain preliminary information or inaccuracies, and may not correctly represent any associated Microsoft Product as commercially released. All Materials are provided entirely "AS IS." To the extent permitted by law, MICROSOFT MAKES NO WARRANTY OF ANY KIND, DISCLAIMS ALL EXPRESS, IMPLIED AND STATUTORY WARRANTIES, AND ASSUMES NO LIABILITY TO YOU FOR ANY DAMAGES OF ANY TYPE IN CONNECTION WITH THESE MATERIALS OR ANY INTELLECTUAL PROPERTY IN THEM.
3. If You are an entity and (a) merge into another entity or (b) a controlling ownership interest in You changes, Your right to use these Materials automatically terminates and You must destroy them.
4. You have no obligation to give Microsoft any suggestions, comments or other feedback ("Feedback") relating to these Materials. However, any Feedback you voluntarily provide may be used in Microsoft Products and related specifications or other documentation (collectively, "Microsoft Offerings") which in turn may be relied upon by other third parties to develop their own products, services or technology ("Third Party Products"). Accordingly, if You do give Microsoft Feedback on any version of these Materials or the Microsoft Offerings to which they apply, You agree: (a) Microsoft may freely use, reproduce, license, distribute, and otherwise commercialize Your Feedback in any Microsoft Offering; (b) You also grant third parties, without charge, only those patent rights necessary to enable Third Party Products to use, implement or interface with any specific parts of a Microsoft Product that incorporate Your Feedback; and (c) You will not give Microsoft any Feedback (i) that You have reason to believe is subject to any patent, copyright or other intellectual property claim or right of any third party; or (ii) subject to license terms which seek to require any Microsoft Offering incorporating or derived from such Feedback, or other Microsoft intellectual property, to be licensed to or otherwise shared with any third party.
5. Microsoft has no obligation to maintain the confidentiality of any Microsoft Offering, or the confidentiality of Your Feedback, including Your identity as the source of such Feedback.
6. This Agreement is governed by the laws of the State of Washington. Any dispute involving it must be brought in the federal or state superior courts located in King County, Washington, and You waive any defenses allowing the dispute to be litigated elsewhere. If there is litigation, the losing party must pay the other party's reasonable attorneys' fees, costs and other expenses. If any part of this Agreement is unenforceable, it will be considered modified to the extent necessary to make it enforceable, and the remainder shall continue in effect. This Agreement is the entire agreement between You and Microsoft concerning these Materials; it may be changed only by a written document signed by both You and Microsoft.

Contents

Contents	ii
Preface.....	iii
Chapter 1. OVERVIEW	1
1.1 Format Identification	1
1.2 Documentation	1
1.3 Device Porting Kit Contents	2
Chapter 2. Sample Applications	4
2.1 Introduction	4
2.2 WMPEncApp Parameter Overview	4
2.3 Standard Uncompressed File Formats	6
2.4 Fixed Point Pixel Formats	7
2.5 Unsupported Pixel Formats	7
2.6 Compression Choices: -q, -d & -l Options	8
2.6.1 -q Image Quantization (1-255)	8
2.6.2 -d Chroma Sub-sampling (0, 1, 2 or 3)	8
2.6.3 -l Overlap Processing (0, 1, 2)	9
2.7 Image Organization Choices: -f, -U, -V, -H & -a Options	10
2.7.1 -a Alpha Channel Structure.....	10
2.7.2 -Q Planar Alpha Quantization (1-255)	10
2.7.3 -f Frequency Order vs. Spatial Order.....	10
2.8 Image Tiling: -U, -V, -H	11
2.9 Encoder Status Reporting: -v, -t.....	11
2.9.1 -v Verbose mode	11
2.9.2 -t Timing information	11
2.10 WMPDecApp Command Line Decoder	12
Chapter 3. Additional Utilities.....	14
3.1 ImageComp	14
3.2 HDR2HDR	14
Chapter 4. Encoder and Decoder Internal Interfaces and Data Structures	15
4.1 ImageInfo Structure	15
4.2 cfColorFormat and bdDepth Combinations	16
4.3 Encoder cfColorFormat and Decoder cfColorFormat Combinations.....	17
4.4 CodecParam Structure	18
4.5 External cfColorFormat and Internal cfColorFormat Combinations.....	19
4.6 ImageBufInfo Structure.....	20

Preface

About This Specification

HD Photo is a file format and associated codec specifically designed to for use with all types of continuous tone photographic content. This document describes the contents of the HD Photo Device Porting Kit 1.0, including various technical details about the software components provided that enable the basic encoder and decoder reference source code to be access via a basic set of programmatic interfaces or by using the included sample command line applications.

The information contained in this specification is subject to change. Every effort has been made to ensure accuracy at the time of publication.

This bit stream specification is written for device manufacturers implementing embedded support for the HD Photo file format encoding and/or decoding and should be used in conjunction with the reference source code and other information provided in the HD Photo Device Porting Kit.

Additional information covering the HD Photo file container format and metadata tags can be found in the "HD Photo Feature Specification." This separate document also contains information required by Windows software developers to implement support for HD Photo encoding and/or decoding.

Formatting Conventions

This specification uses the following formatting conventions:

Terms are formatted like **this**.

Important comments, typically highlighting unimplemented or preliminary features look like this.

Code looks like this.

Raw text and editorial notes look like this.

Language Notes

In this specification, the words that are used to define the significance of each particular requirement are capitalized. These words are used in accordance with their definitions in RFC 2119 and their meaning is reproduced here for convenience:

- **MUST**. This word, or the adjective "REQUIRED," means that the item is an absolute requirement of the specification.
- **SHOULD**. This word, or the adjective "RECOMMENDED," means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
- **MAY**. This word, or the adjective "OPTIONAL," means that this item is truly optional. For example, one implementation may choose to include the item because a particular marketplace or scenario requires it or because it enhances the product. Another implementation may omit the same item.

Chapter 1. OVERVIEW

1.1 Format Identification

Throughout the development of the HD Photo still image file format, the project was first known (internal to Microsoft) by its code name "Photon". Throughout its Beta release, and when publicly introduced at WinHEC 2006, the file format was named "Windows Media™ Photo." With the introduction of the released version of the Device Porting Kit, the format name has been officially changed to "HD Photo."

"Windows Media™ Photo" is still used as the name for the Windows implementation of the "HD Photo" format, in both Windows Vista and the Windows Image Component (WIC) and .NET Frameworks 3.0 redistributable packages for down-level versions of Windows. "Windows Media™ Photo" is still used internally with this release of the Device Porting Kit.

Windows Media™ Photo and HD Photo refer to exactly the same file format. There are no features or changes that differentiate one from the other. They are two names for the same format.

We have extended the specification to allow two different file extensions – either .wdp or .hdp – to be used to identify an HD Photo (or Windows Media™ Photo) file. With the 1.0 release of Windows Vista, only the .wdp extension is recognized. This will be updated at some point in the future. Applications that support the HD Photo file format should recognize either extension, and ideally offer the option to create files with either extension.

While we recognize the potential confusion and issues that this name change can create, we believe the new name, initiated in part based on feedback from our partners, provides a better identification for broad, cross-platform support for this new still image file format.

1.2 Documentation

This document provides supplemental information about the HD Photo Device Porting Kit (DPK) not covered in the other specifications. This includes documentation on the command line encoder and decoder utilities that are provided here as sample reference source code and additional technical details about the application program interfaces (API's) and data types and structures used to interface the core encoder and decoder modules with the command line utility application code.

The "HD Photo Feature Specification" document contains information about the HD Photo file container format, metadata tags, and the API's for HD Photo support on Windows.

The "HD Photo Bitstream Specification" document contains complete technical details on the encoder and decoder compression algorithms and the compressed data (elementary) bitstream format.

1.3 Device Porting Kit Contents

The complete contents of this Device Porting Kit are as follows:

Directory	File	Description
bin\	hdr2hdr.exe	Unsupported utility programs
	imagecomp.exe	
common\include\	guiddef.h	Shared header files
	wmsal.h	
	wmspecstring.h	
	wmspecstrings_adt.h	
	wmspecstrings_strict.h	
doc\	readme.txt	HD Photo Device Porting Kit documentation files (including this document.)
	HDPhoto_Bitstream_Spec_1.0.doc	
	HDPhoto_Feature_Spec_1.0.doc	
	HDPhoto_DPK_Spec_1.0.doc	
	hdphoto_dpk_eula.txt	
image\decode\	decode.c	ANSI C reference source code for the HD Photo decoder
	decode.h	
	huffman.c	
	postprocess.c	
	segdec.c	
	strdec.c	
	strInvTransform.c	
	strPredQuantDec.c	
	WMPTranscode.c	
image\encode\	encode.c	ANSI C reference source code for the HD Photo encoder
	encode.h	
	segenc.c	
	strenc.c	
	strFwdTransform.c	
	strPredQuantEnc.c	
image\sys\	adapthuff.c	ANSI C reference source code and additional functions used for development and testing that are shared by both the encoder and the decoder
	ansi.h	
	common.h	
	image.c	
	perfTimer.h	
	perfTimerANSI.c	
	perfTimerWin32.c	
	strcodec.c	
	strcodec.h	
	strPredQuant.c	
	strTransform.c	
	strTransform.h	
	windowsmediaphoto.h	
	xplatform_image.h	

Directory	File	Description
image\VC8Projects\	CommonLib.vcproj	Project files to build the various device porting kit components
	DecodeLib.vcproj	
	EncodeLib.vcproj	
	MemtraceLib.vcproj	
image\x86\	strenc_x86.c	X86 specific support code, not directly related to the encoder or decoder
	x86.h	
Network\client\debughlp\memtrace\	heaptrace.cpp	A Microsoft developed memory trace utility provided to assist developers in optimizing and debugging implementations of the encoder and decoder
	heaptrace.h	
	memory_trace_lib_doc.htm	
	memtrace.cpp	
	memtrace.h	
	memtrace_com.h	
	regtrace.cpp	
	regtrace.h	
	sysheap.cpp	
	sysMalloc.cpp	
sample\	Shortcut to WMP.sln.lnk	Link to build the sample apps
Systems\tools\SampleCodeNew\WMPEncDecoder\	WmpDecApp.c	Source code for the sample command line decoder app
Systems\tools\SampleCodeNew\WMPEncDecoder\	WmpEncApp.c	Source code for the sample command line encoder app
Systems\tools\VC8Projects\	WMP.sln	Project files to build the various components of the command line sample applications
	WMPDecApp.vcproj	
	WMPEncApp.vcproj	
	WMPGlueLib.vcproj	
	WMPMetaLib.vcproj	
Systems\tools\WMPGlueLib\	WMPGlue.c	Sample source code to provide minimal image file format handling as required by the sample command line applications
	WMPGlue.h	
	WMPGlueBmp.c	
	WMPGlueHdr.c	
	WMPGluePFC.c	
	WMPGluePnm.c	
	WMPGlueTif.c	
	WMPGlueWmp.c	
	WMPGlueYUV.c	
Systems\tools\WMPMetaLib\	WMPMeta.c	Sample source code to provide minimal metadata handling as required by the sample command line applications.
	WMPMeta.h	
xplat\	Makefile	Example make file to build the DPK components on a Unix/Linux system, including support for big-endian hardware architecture

Chapter 2. Sample Applications

2.1 Introduction

This device porting kit includes the source code for command line utilities to convert files between HD Photo format and other common, uncompressed file formats. These utilities are provided as example applications for calling the HD Photo encoder and decoder. This section documents the usage of these utilities, including a complete explanation of the command line parameters. These are not full-featured file conversion applications. While they are useful development tools, they are provided only as sample applications for exercising the HD Photo encoder and decoder.

2.2 WMPEncApp Parameter Overview

This command line utility converts certain uncompressed file formats into equivalent HD Photo files. It provides a complete set of command line options to control all supported HD Photo Encoder options. Here is a summary of the usage of WMPEncApp and the various command line options. All of these options will be discussed in detail in later sections.

wmpencapp [options]...

-i input.bmp/tif/hdr	Input image file name bmp: <=8bpc, BGR tif: >=8bpc, RGB hdr: 24bppRGBE only
-o output.wdp	Output HD Photo file name
-q quality	[1 - 255] Default = 1, lossless
-c format	Required to define uncompressed source pixel format
	0: 24bppBGR
	1: BlackWhite
	2: 8bppGray
	3: 16bppGray
	4: 16bppGrayFixedPoint
	5: 16bppGrayHalf
	7: 32bppGrayFixedPoint
	8: 32bppGrayFloat
	9: 24bppRGB
	10: 48bppRGB
	11: 48bppRGBFixedPoint
	12: 48bppRGBHalf
	14: 96bppRGBFixedPoint
	15: 128bppRGBFloat
	16: 32bppRGBE
	17: 32bppCMYK
	18: 64bppCMYK
	22: 32bppBGRA
	23: 64bppRGBA
	24: 64bppRGBAFixedPoint
	25: 64bppRGBAHalf
	27: 128bppRGBAFixedPoint

	28: 128bppRGBAFloat
	29: 16bppBGR555
	30: 16bppBGR565
	31: 32bppBGR101010
	32: 40bppCMYKA
	33: 80bppCMYKA
	34: 32bppBGR
-d chroma sub-sampling	0: Y-only 1: YCoCg 4:2:0 2: YCoCg 4:2:2 3: YCoCg 4:4:4 (default)
-l overlapping	0: No overlapping 1: One level overlapping (default) 2: Two level overlapping
-f	Frequency order bit stream (default is spatial)
-t	Display timing information
-v	Display verbose encoder information
-V tile_wd0 tile_wd1 ...	Macro block rows per tile
-H tile_ht0 tile_ht1 ...	Macro block columns per tile
-U num_h_tiles num_v_tiles	Horizontal & vertical tile count for uniform tiling
-b Black/White	Applies to 1bpp black/white images 0: 0 = black (default) 1: 0 = white
-a alpha channel format	Required for any pixel format with an alpha channel 2: Planar alpha 3: Interleaved alpha Other: Reserved, do not use
-Q quality for planar alpha	[1 - 255] Default: same as image quality
-F trimmed flexbits	[0 - 15] 0: no trimming (default) 15: trim all
-s skip subbands	0: All subbands included (default) 1: Skip flexbits 2: Skip highpass 3: Skip highpass & lowpass (DC only)

So for example to create a HD Photo file from a typical 24-bit .bmp using reasonably high quality lossy compression, the command line would be:

```
wmpencapp -i input.bmp -o output.wdp -q 10
```

This scenario uses the default settings for most of the encoder options. Obviously, we'd like to take full control of these options to choose exactly how the HD Photo file is created. The following sections describe these options in detail.

2.3 Standard Uncompressed File Formats

The DPK Tools are certainly not general purpose file format conversion utilities. They provide the absolute minimum support for uncompressed source and destination file formats; the specific file formats supported are TIFF, BMP and HDR. Only certain variations of these formats are supported and they are, for the most part, tied to the pixel format being converted to or from. The DPK Tools do not perform any pixel format conversion. Therefore the source uncompressed image must be in the desired pixel format for the encoded HD Photo file. Only minimal data validation is performed; if the source pixel format is incorrect, you will most likely create a bad HD Photo file.

Here are the uncompressed source file formats supported by the DPK Tools, and the specifics for each pixel format supported. Each of the file formats and specific image formats listed below correspond to a mode that can be created using Adobe PhotoShop CS2.

TIFF The image should be flattened; it should not contain any layers. If it does contain layers, the "Discard Layers and Save a Copy" option should be selected under Layer Compression. Image Compression must be set to "None"; any compressed TIFF format will cause an error or convert to a bad HD Photo file. TIFF images must always be stored in "Interleaved" pixel order; "Per Channel" pixel order is not supported. Byte order should be set to "IBM PC." "Save Image Pyramid" should be unchecked.

The specific pixel format created depends on the correct combination of Image Mode in PhotoShop, the specific TIFF save options specified, and the encoder pixel format option specified for WMPEncApp.exe. The following table lists all the possible combinations

<i>PhotoShop Mode</i>	<i>Bit Depth</i>	<i>Encodes as</i>	<i>-c</i>	<i>Notes</i>
RGB/8		24bppRGB	9	
RGB/16		48bppRGB	10	
RGB/16 w/ alpha		64bppRGBA	23	
RGB/32	16 bit (Half)	48bppRGBHalf	12	
RGB/32 w/ alpha	16 bit (Half)	64bppRGBAHalf	25	
RGB/32 w/ alpha	32 bit (Float)	128bppRGBFloat	15	Fill alpha black
RGB/32 w/ alpha	32 bit (Float)	128bppRGBAFloat	28	
CMYK/8		32bppCMYK	17	
CMYK/8 w/ alpha		40bppCMYKA	32	
CMYK/16		64bppCMYK	18	
CMYK/16 w/ alpha		80bppCMYKA	33	
Gray/8		8bppGray	2	
Gray/16		16bppGray	3	
Gray/32	16 bit (Half)	16bppGrayHalf	5	
Gray/32	32 bit (Float)	32bppGrayFloat	8	
Bitmap		1bppBlackWhite	1	

For 1bppBlackWhite encoding, the -b option allows you to specify how interpret black vs. white values. When encoding from TIFF files, the default will generate the correct results and this option should not be required.

BMP This file format is only used for 8 bit per channel (bpc) or smaller bit depths. It differs from the equivalent TIFF format because RGB data is stored in the uncompressed bit stream in BGR rather than RGB channel order. When creating the uncompressed image in PhotoShop, only RGB/8

mode should be used. Under BMP Options, the File Format should always be set to "Windows". "Compress (RLE)" and "Flip row order" should never be checked. The Basic or Advanced Modes under the BMP Save options should be set in combination with the appropriate encoder options to achieve the desired pixel format according to the following table.

<i>PhotoShop Mode</i>	<i>BMP Options</i>	<i>Encodes as</i>	<i>-c</i>	<i>Notes</i>
RGB/8	Basic: 24 bit	24bppBGR	0	
RGB/8 w/ alpha	Basic: 32 bit	32bppBGR	34	Fill alpha black
RGB/8 w/ alpha	Basic: 32 bit	32bppBGRA	22	
RGB/8	Adv: X1 R5 G5 B5	16bppBGR555	29	
RGB/8	Adv: R5 G6 B5	16bppBGR565	30	

HDR This file format is only used for encoding to the HD Photo 24bppRGBE pixel format. This special floating point pixel format uses a shared exponent and three independent mantissas to encode the entire pixel. Unfortunately, while PhotoShop CS 2 supports saving in .HDR file mode, it only saves using the compressed option. WMPEncApp.exe only supports uncompressed .HDR files. So, we've provided a simple command line utility called HDR2HDR that only does one thing: It reads and decompresses a compressed .HDR file and saves it as an uncompressed .HDR file. HDR2HDR.EXE is included in the DPK.

<i>PhotoShop Mode</i>	<i>PhotoShop Save As...</i>	<i>Encodes as</i>	<i>-c</i>
RGB/32	Radiance (*.HDR)	24bppRGBE	16

2.4 Fixed Point Pixel Formats

The one significant omission from the pixel formats listed in the tables above is the set of fixed point formats. The fixed point pixel formats provide an excellent solution for retaining the full content of an image source while still providing efficient storage and processing.

These fixed point formats are one of the innovations introduced with HD Photo and the new graphics infrastructure in Windows Vista and .NET Frameworks 3.0. No other file format supports fixed point encoding. Because WMPEncApp.exe has no built-in capability for pixel format conversion, there is no way we can encode fixed point HD Photo files from standard TIFF or BMP files.

In reality, if we write a little software, we could convert floating point image data to fixed point and still store it in a TIFF file. While the resulting TIFF file containing fixed point image data would not display correctly, it could be converted to a fixed point HD Photo file using WMPEncApp.exe. That's why there are -c option values defined for the various fixed point pixel formats. But for this to work, it's your responsibility to first convert the uncompressed image data to fixed point.

2.5 Unsupported Pixel Formats

There are several other pixel formats that can't easily be encoded using WMPEncApp.exe

It is possible to encode images in any of the pre-multiplied alpha pixel formats, but the uncompressed image will have to first be pre-processed to multiply the RGB channels by the alpha channel value. This can be done using the appropriate blending operations in Photoshop. Like fixed point pixel formats, a TIFF file in a pre-multiplied pixel format will not display correctly, but can be converted to an equivalent HD Photo file using WMPEncApp.exe.

While WMPEncApp.exe includes a -c option value for 32bpcRGB101010 pixel format, there is no way to create this pixel format in an uncompressed file using PhotoShop. You can write your own code to create

this pixel format within a TIFF file. While WMPEncApp.exe will be able to convert this to an HD Photo file, the TIFF file you create will not display correctly in any other application.

WMPEncApp.exe does not support encoding any of the n-Channel pixel formats.

2.6 Compression Choices: -q, -d & -l Options

There are three parameters that control the tradeoffs between image quality and compressed file size. In addition to their individual effect, it's also important to understand how these three parameters interact with each other.

2.6.1 -q Image Quantization (1-255)

One of the principal ways lossy compression is achieved is to "quantize" a set of continuous values into a smaller set of representative values. In this way, "loss" is achieved by mapping values that are close together to the same value. Only the remaining set of values needs to be coded and saved, reducing the amount of storage required. The greater the degree of quantization, the more the content can be compressed, but in doing so, more of the small differences among similar values are lost.

The quantization level, specified by the -q option basically defines the amount of similarity that can be discarded. It is an arbitrary range from 1-255; the quantization value does not correspond directly to any specific difference amount. Quantization determines the desired image quality rather than the desired compression ratio. The actual compression ratio is a function of both the quantization and the image content; images with less complex content will have fewer differences among values and will achieve better compression without the need for greater quantization. Additionally, the quantization is also highly dependent on the specific pixel format, most importantly the bit depth. Higher values will be required to greater bit depths to achieve a comparable compression ratio, since larger bit depths provide a greater range of possible values and therefore will need more quantization.

HD Photo provides the unique capability of preserving all data values during quantization, effectively providing mathematically lossless compression. When the quantization is set to 1, no values are discarded and all encoded pixel values will be returned with absolutely no loss. This is the default setting if no value for the -q option is specified.

2.6.2 -d Chroma Sub-sampling (0, 1, 2 or 3)

We can choose to reduce the resolution of the chrominance of an image prior to the quantization process. Reducing the chrominance resolution, or chroma sub-sampling, has long been understood as an effective way to reduce image content with very little perceptible degradation. In fact, virtually all television or video you watch, whether analog or digital, takes advantage of chroma sub-sampling to reduce the required bandwidth. The JPEG compression format always uses chroma sub-sampling as well. In fact, the unique capability of HD Photo is not that we provide chroma sub-sampling, but that we provide a mechanism for you to reduce or eliminate this technique to improve image quality. Of course, this only applies to RGB color images.

An image is first reorganized from RGB into a channel for luminance and two channels to describe the color information (or chrominance.) If all chrominance is discarded, what's left is a monochrome image. Typically, we don't want to go that far!

Many video systems, as well as the JPEG compression format (or at least the most common variant of it that we all use) discards 75% of the chrominance information. The resolution of the color information is reduced by a factor of two in both dimensions. So every four pixels in an image are represented by four luminance values but only two (one for each chroma channel) chrominance values. What started out as 12 values (four pixels with three channels each) has been cut in half; only 6 values (four luminance values and two chrominance values) have to be saved.

In the world of digital imaging, this is referred to as 4:2:0 chroma sub-sampling, or more simply as 4:2:0. When all chrominance information is retained (no values are discarded), this is referred to as 4:4:4. Another popular approach, particularly for professional video applications, is to only discard 50% of the chroma values; two values for each chroma channel, or four values in total are retained. This is referred to as 4:2:2. Finally, if we discard all color information, retaining only the luminance, this is described as 4:0:0. HD Photo supports all these modes.

- c 3 (4:4:4) All color information is retained, assuring full resolution of the chrominance information. This is the default and is the recommended setting to achieve the best overall image quality. Whenever an image is stored as an intermediate format and further editing is anticipated, it is highly recommended to use 4:4:4.
- c 2 (4:2:2) The color information is encoded at $\frac{1}{2}$ the resolution of the luminance information. Four each set of four pixels, four luminance values are used and the eight chrominance values are reduced down to four (two for each chroma channel.) This provides perceptively lossless color encoding for the final delivery of an image. However, if further editing of the image is anticipated, it's recommended than any chroma sub-sampling be avoided.
- c 1 (4:2:0) The color information is encoded at $\frac{1}{4}$ the resolution of the luminance information. Four each set of four pixels, four luminance values are used and the eight chrominance values are reduced down to two (one for each chroma channel.) This is the same sub-sampling used by JPEG. When converting a JPEG file to HD Photo, there is no need to specify a higher chroma sub-sampling mode than 4:2:0.
- c 0 (4:0:0) All color information is discarded and only the luminance information is retained, effectively creating a monochrome image. For performance reasons, HD Photo uses a non-traditional method to calculate luminance. Therefore, the resulting monochrome image will not appear identical to a monochrome version of the image created using other tools. Additionally, although all color information is discarded, the pixel format is not changed, so the image is still stored using an RGB pixel format. It is strongly recommended that if you want to create a monochrome image, the image should first be converted to monochrome using an appropriate image editing application to achieve the desired result, and then this monochrome image should be encoded using the appropriate Gray pixel format.

2.6.3 -I Overlap Processing (0, 1, 2)

HD Photo uses an advanced version of a macro-block based compression scheme. To achieve the best performance and minimize the amount of memory required to encode or decode an image, the overall image is subdivided into a set of 16x16 pixel macro blocks. Each macro block is further divided into four 4x4 pixel blocks. All image encoding and decoding operations are performed on these blocks and macro-blocks. As a result, for high quantization values (when we are discarding a higher amount of similar pixel values), the steps between blocks and macro blocks may become visible as artifacts in the compressed image. This is very common with JPEG (which also uses macro blocks) and significantly reduces the amount of compression that can be used without creating these visible artifacts.

HD Photo addresses this problem through a combination of better quantization and an additional step of overlap processing. This overlap processing takes into account the values of pixels in neighboring blocks and macro blocks when choosing the quantization values that represent similar adjacent pixels. By doing so, the visible differences among adjacent blocks and macro blocks are dramatically reduced.

Two levels of optional overlap processing can be specified via the -I parameter. Single level overlap processing (-I 1) is performed at the 4x4 block level. For all pixels in the block, bordering pixels in adjacent blocks are also evaluated when choosing the quantization values for that block. Double level overlap processing (-I 2) also analyzes neighboring adjacent pixels when choosing quantization values at the 16x16 macro-block level.

The default value for the `-l` parameter is 1 and single level overlap processing should be used for most typical encoding scenarios. For very high quantization levels, double-level overlap processing may be appropriate, but this will trade off the potential for macro block artifacts for a loss of image detail. Setting the `-l` parameter to 0 suppresses any overlap processing. This can speed performance, but is only recommended for very low quantization values. Specific quantization thresholds for choosing the appropriate level of overlap processing are highly dependent on the image content and cannot be predicted. Trial and error will be your best guide. But when in doubt, stick with the default.

2.7 Image Organization Choices: `-f`, `-U`, `-V`, `-H` & `-a` Options

In addition to controlling the quality vs. the size of the image, HD Photo provides a number of choices on exactly how the image information is structured or organized within the file. This includes the alpha channel structure, image tiling, and the overall data order of frequency vs. spatial.

2.7.1 `-a` Alpha Channel Structure

Obviously, this option only applies to images with alpha channels. The `-a` option is required for any uncompressed source image that contains an alpha channel and it will be ignored if the image does not include an alpha channel.

HD Photo supports both interleaved and planar alpha channels. An interleaved alpha channel is stored in sequence with the channels that describe the image contents (RGB or CMYK). It simply adds an additional channel to each pixel. A planar alpha channel is stored as a completely separate image within the HD Photo file container. The alpha channel is encoded separately from the image RGB or CMYK data. The decoder can decode both and re-interleave the channels to deliver a bitmap with alpha channel. Or if only one element (the image content or the alpha channel) is required, a decoder can return just that portion with no need for all the additional processing required for the other portion.

Setting the `-a` parameter to 3 specifies that the image be encoded with an interleaved alpha channel. Conversely, setting this option to 2 will encode an image with a planar alpha channel. Any other value is illegal and will generate an error.

The default value is 2, and the parameter is only meaningful when encoding an image with alpha.

2.7.2 `-Q` Planar Alpha Quantization (1-255)

This option set the quantization level for the planar alpha image, and the values are interpreted the same as the Image Quantization (`-q`) parameter. If omitted, it is set to the same value as the Image Quantization (`-q`). This parameter is only meaningful when encoding planar alpha channels.

2.7.3 `-f` Frequency Order vs. Spatial Order

HD Photo makes it possible to organize the compressed image data sequentially in either spatial or frequency order. Spatial order is the typical choice for encoding by a device. The sequential compressed data stream represents the image in macro block rows starting at the upper left corner, from left to right and from top to bottom. It allows the image to be encoded sequentially in rows of pixels, minimizing the total memory required. Frequency order groups the data in three different frequencies and places it sequentially in the file starting with the low frequency information, followed by the middle frequency, and finally by the high frequency details. Frequency order makes it much more efficient to decode a low resolution version of the image, minimizing the amount of compressed image data that must be parsed to find the required low frequency content. When encoding on a typical personal computer, the performance difference between encoding in frequency order vs. spatial order is insignificant.

Including the `-f` option specifies that the image be encoded in frequency order. This is typically preferred because of the performance benefits when decoding the image to lower resolutions. Omitting the `-f` option will encode the image in spatial order. This is (obviously) the default.

2.8 Image Tiling: -U, -V, -H

HD Photo allows an image to be subdivided into individual rectangular tiles. Each tile is stored in the compressed bit stream as a fully self-describing sub-picture. This makes it possible to decode a tile without ever having to process the compressed data for any other tile. The main purpose for this feature is to optimize an image for region decoding. The request to decode an arbitrary region only needs to process the tiles that represent that region.

Both uniform tiling and non-uniform tiling are supported. With uniform tiling, all tiles (with the potential exception of the right-most column and bottom-most row) share the same width and height. With non-uniform tiling, the desired width and height for each tile row and column can be specified. Tiles always have uniform height within each tile row, and uniform width within each tile column.

The -U option, followed by the column and row count, specifies uniform tiling. The image is sliced into the requested number of columns and rows, spacing them as evenly as possible. Tiles are always a multiple of macro blocks (16x16 pixels.) If the image width or height is not evenly divisible in macro block increments by the requested column and row count, the right-most column and/or bottom-most row will be re-sized accordingly. The remaining columns and rows will always be of uniform width and height. Columns cannot be less than one macro block in width and rows cannot be less than one macro block in height. If the requested column or row count results in tiles smaller than this, the appropriate column or row count will be adjusted accordingly.

Instead of using the -U option for uniform tiling, the -H and -V options can be used to specify a vector of non-uniform tile widths and heights. The vector of space-delimited values following each parameter expresses the tile dimension in macro blocks (multiples of 16 pixels.) If insufficient values are specified to describe the entire width or height of the image, the right-most column and/or bottom-most row will be sized to contain the remaining pixels. If the vector of macro block sizes exceeds the dimension of the image, the extra values in the vector will be ignored and the right-most column and/or bottom-most row will be resized to match the remaining pixels in that image dimension.

In general, image tiling is not required. Its use, and the appropriate choice of tile size, is application dependent. It's recommended that to minimize the performance penalties associated with tiling, tiles smaller than 256x256 pixels should be avoided.

2.9 Encoder Status Reporting: -v, -t

The following are some additional encoding options that control the encoding process itself.

2.9.1 -v Verbose mode

When present, option enables the output of extended status and results information via the STDOUT output. This information can be piped to a file or other destination using the standard command line conventions (> or >>) for STDOUT piping. Most of the reported information is self-explanatory.

2.9.2 -t Timing information

When present, this option enables the output of encoder timing information via the STDOUT output. As above, it can also be redirected. This timing information was something we included for our own testing. It does not use a very accurate method to measure performance and while it may be informative, it should not be relied on as an precise performance indicator. Also, please remember that the DPK Tools do not include the platform optimization code that is implemented in version for Microsoft Windows.

2.10 WMPDecApp Command Line Decoder

This sample application can convert HD Photo files to different uncompressed file formats. This utility is provided as sample code, and is not designed to be a full-fledged application. It only supports the specific uncompressed file formats necessary to receive data in the various HD Photo pixel formats. It only processes the image data and does not attempt to transfer or convert metadata other than the tags specifically required to define the image geometry, structure and format. It has only minimal error checking or resiliency to bad source data or incorrect parameters.

wmpdecapp [options]...

-i input.wdp	Input HD Photo file name
-o output.bmp/tif/wdp	Output image file name bmp: <=8bpc, BGR tif: >=8bpc, RGB wdp: for compressed domain transcode
-c format	Specifies the uncompressed output format 0: 24bppBGR 1: BlackWhite 2: 8bppGray 3: 16bppGray 4: 16bppGrayFixedPoint 5: 16bppGrayHalf 7: 32bppGrayFixedPoint 8: 32bppGrayFloat 9: 24bppRGB 10: 48bppRGB 11: 48bppRGBFixedPoint 12: 48bppRGBHalf 14: 96bppRGBFixedPoint 15: 128bppRGBFloat 16: 32bppRGBE 17: 32bppCMYK 18: 64bppCMYK 22: 32bppBGRA 23: 64bppRGBA 24: 64bppRGBAFixedPoint 25: 64bppRGBAHalf 27: 128bppRGBAFixedPoint 28: 128bppRGBFloat 29: 16bppBGR555 30: 16bppBGR565 31: 32bppBGR101010 32: 40bppCMYKA 33: 80bppCMYKA 34: 32bppBGR
-r top left height width	Specifies the rectangle for region decode

-T m Reduced resolution (mipmap) decode
 0: Full resolution (default)
 1: 1/2 resolution (down-sampled)
 2: 1/4 resolution (native decode)
 3: 1/8 resolution (down-sampled)
 4: 1/16 resolution (native decode)
 >4: 1/(2^m) resolution (down-sampled)

-O orientation 0: No transformation (default)
 1: Flip vertically
 2: Flip horizontally
 3: Flip vertically & horizontally
 4: Rotate 90 degrees CW
 5: Rotate 90 degrees CW & vert flip
 6: Rotate 90 degrees CW & horz flip
 7: Rotate 90 degrees CW & horz/vert flip

-s skip subbands Used for compressed domain transcoding
 0: All subbands included (default)
 1: Skip flexbits
 2: Skip highpass
 3: Skip highpass & lowpass (DC only)

-a alpha decode 0: Decode without alpha channel
 1: Decode only alpha channel
 2: Decode image & alpha (default)

-p strength Post processing filter strength
 0: None (default)
 1: Light
 2: Medium
 3: Strong
 4: Very strong

-C Suppress overlapping macro blocks
 (Used for compressed domain tile extraction)

-t Display timing information

-v Display verbose decoder information

Eg: wmpdecapp -i input.wdp -o output.bmp -c 0

Chapter 3. Additional Utilities

3.1 ImageComp

This is a basic command line utility that provides measurement data comparing two uncompressed image files. ImageComp provides a variety of image measurements, and can optionally provide a difference image for certain formats.

```
imagecomp ImageFileName1 ImageFileName2 [-i InputFormat] [-M 0/1]
        [-o diff.bmp / -O diff.raw] -k [scalefactor]
```

```
-i InputFormat: 1 BMP (Default)
                2 TIF
                3 HDR
```

```
-M Mode          0 Only SSE, MSE and PSNR (Default)
                1 More outputs (MaxDiff, Error Image...)
```

```
-o filename.bmp Produces difference image in .bmp format
```

```
-O filename.raw Produces difference image in raw BGR888 format
```

```
-k Scale factor for difference image (default = 1)
```

```
Difference image = clip((input1 - input2) * k + 128)
```

```
Difference image options (-o, -O) only work with BMP images
```

3.2 HDR2HDR

This is a simple command line utility to convert a compressed .hdr file to an uncompressed .hdr file. WMPEncApp.exe can encode HD Photo files in the Radiance (RGBE) format from uncompressed .hdr files. However, Adobe Photoshop CS2 can only create compressed .hdr files.

Usage:

```
HDR2HDR compressed.hdr uncompressed.hdr
```

Chapter 4. Encoder and Decoder Internal Interfaces and Data Structures

This section provides implementation details on tables and structures used in DPK encoder and decoder reference source code. Some information defined here is reserved for future use.

4.1 ImageInfo Structure

cWidth cHeight	image size; must be between $1-2^{18}$; cWidth must be even for YUV_422 and YUV_420; cHeight must be even for YUV_420; for BAYER, if sensor image size is w & h, then cWidth = w / 2, cHeight = h / 2.	B
cfColorFormat	color format; Y_ONLY, YUV_420, YUV_422, YUV_444, CMYK, BAYER, N_CHANNEL, CF_RGB, CF_RGBE, CF_PALLETIZED; for planar alpha, use Y_ONLY; see table 3 for valid encoder/decoder cfColorformat combinations	O
bdBitDepth	color component bit depth; BD_1, BD_8, BD_16, BD_16S, BD_16F, BD_32, BD_32S, BD_32F, BD_5, BD_10; refer to the table 2 for supported cfColorFormat and bdBitDepth combinations	B
cBitsPerUnit	bits per pixel unit; a pixel unit is color components of a pixel with possible leading and/or trailing padding; if not BD_1, must be multiples of 8; for YUV_420, a 'pixel' means YYYYUV; for YUV_422, a pixel means UYVY; for BAYER, a unit is a sensor unit and no padding is allowed.	U
cLeadingPadding	Number of leading padding of a pixel unit	U
bRGB	RGB or BGR(valid only if BD_8 and CF_RGB); true for RGB order, false for BGR order	
bpBayerPattern	bayer pattern(valid only if BAYER); 0(GR/BG), 1(RG/BG), 2(BG/GR), 3(GB, RG)	B
cChromaCentering	relative location of Chroma w.r.t Luma	B
cChromaInterpretation	colorspace	B
bSkipFlexBits	decode (false) or not decode(true) flex bits	D
cThumbnailWidth cThumbnailHeight	size of a thumbnail; for s(= 2^t) : 1 thumbnail (s = 1 for full resolution), cThumbnailWidth = (cWidth + s - 1) / s, cThumbnailHeight = (cHeight + s - 1) / s; codec will reset invalid thumbnail to full resolution	D
cROILeftX cROIWidth cROITopY cROIHeight	region to be decoded in a thumbnail; codec will trim to fit in [0, cThumbnailWidth), [0, cThumbnailHeight) or reset it to full thumbnail if the region is null or completely detached from the thumbnail	D

oOrientation	decoded image orientation; CRW FlipH FlipV O_NONE = 0, // 0 0 0 O_FLIPV, // 0 0 1 O_FLIPH, // 0 1 0 O_FLIPVH, // 0 1 1 O_RCW, // 1 0 0 O_RCW_FLIPV, // 1 0 1 O_RCW_FLIPH, // 1 1 0 O_RCW_FLIPVH, // 1 1 1	D
cPostProcStrength	strength of post processing; 0(none) 1(light) 2(medium) 3(heavy) >3(very heavy)	D
fPaddedUserBuffer	boolean indicating whether the output buffer may use optimized code path	D

B: info carried in bitstream and not decoder over writable

O: info carried in bitstream and decoder over writable

D: decoder only

U: user input, could be different for encoder and decoder

4.2 cfColorFormat and bdDepth Combinations

	Y_ONLY	YUV_420	YUV_422	YUV_444	CMYK	BAYER	N_CHANNEL	CF_RGB	CF_RGBE	CF_PALLETIZED
BD_1	√	x	x	x	x	x	x	x	x	
BD_8	√	√	√	√	√	√	√	√	√	
BD_16	√	√	√	√	√	√	√	√	x	
BD_16S	√			√	√	√	√	√	x	
BD_16F	√			√			√	√	x	
BD_32	√			√			√	√	x	
BD_32S	√			√			√	√	x	
BD_32F	√			√			√	√	x	
BD_5	x	x	x	x	x	x	x	√	x	
BD_10	x	x	x	x	x	x	x	√	x	

√: supported x: not supported blank: might be supported

4.3 Encoder cfColorFormat and Decoder cfColorFormat Combinations

decoder\encoder	Y_ONLY	YUV_420	YUV_422	YUV_444	CMYK	BAYER	N_CHANNEL	CF_RGB	CF_RGBE	CF_PALLETIZED
Y_ONLY	√	√	√	√	√	√	√	√		
YUV_420		√	√	√						
YUV_422			√	√						
YUV_444				√						
CMYK					√					
BAYER						√				
N_CHANNEL							√			
CF_RGB						√		√		
CF_RGBE									√	
CF_PALLETIZED										

√: supported blank: not supported

4.4 CodecParam Structure

uiDefaultQPIndex	for macroblock quantization (DQUANT)	B
cfColorFormat	Internal color format; Y_ONLY, YUV_420, YUV_422, YUV_444, CMYK, BAYER, N_CHANNEL; for planar alpha, use Y_ONLY; see table 5 for valid external / internal color format combinations	B
bdBitDepth	internal bit depth; BD_SHORT, BD_LONG; Only support BD_LONG now.	B
olOverlap	type of overlap; OL_NONE, OL_ONE, OL_TWO	B
bfBitstreamFormat	bitstream format; SPATIAL, FREQUENCY	B
uAlphaMode	alpha channel info; encoder: 0: RGB only 1: A only 2: Planar RGBA 3: Interleaved RGBA this info is in the bitstream: 0(no alpha encoded), else(interleaved alpha encoded). decoder: 0: don't decode alpha or no alpha in the bitsream 1: only decode alpha if presents: x + alpha => alpha transcoding 2: decode alpha and other channels	O
cChannel	number of color channels including alpha if presents; user needs to set it properly only if N_CHANNEL; must be 1 to 16	B
cNumOfSliceMinus1V; uiTileX[256] cNumOfSliceMinus1H; uiTileY[256]	tiling info: number of horizontal/vertical slices and width/height; no more than 256 slices each direction; slice width/height must be no bigger than 65536	B
sbSubband	kept subbands	D
uiTrimFlexBits	trimmed flexbits during encode	U
nLenMantissaOrShift; nExpBias;	32f and 32s conversion parameters	B

pWStream; cbStream;	Bitsream pointer and number of bytes in the bitstream	U
bBlackWhite	the mode user would like to encode 1bpp black/white images. 0: 0 = black 1: 0 = white	B
fMeasurePerf	turn on/off perf measurement	U
bVerbose	turn on/off verbose mode	U

4.5 External cfColorFormat and Internal cfColorFormat Combinations

internal\external	Y_ONLY	YUV_420	YUV_422	YUV_444	CMYK	BAYER	N_CHANNEL	CF_RGB	CF_RGBE	CF_PALETIZED
Y_ONLY	B	I	I	I	I	I	I	I	I	
YUV_420	O	B	B	B		I		B	B	
YUV_422	O		B	B		I		B	B	
YUV_444	O			B		I		B	B	
CMYK	O				B					
BAYER	O					B		D		
N_CHANNEL	O						B			

I: input image O: output image B: both input and output blank: not supported

4.6 ImageBufInfo Structure

pv	pointer to input/output image buffer
cLine	number pixel rows in buffer; for YUV_420, a pixel row means 2 Y rows and 1 U/V row; for YUV_422, a pixel row means 2 sensor row; on encoder side, one MB row at a time and cLine >= 16 (8 if YUV_420) except for the last MB row; on decoder side, cLine >= cROIHeight (cROIWidth if oOrientation >= O_RCW)
cbStride	how many bytes each pixel row occupies in the buffer; must be enough to contain a pixel row

An example: possible ways to encode a RGBA image (BD_8, cBitsPerPixel = 24)

	cfColorFormat	cLeadingPadding	uAlphaMode
Encode RGBA	YUV_xxx or Y_ONLY	0	2 (Planar) or 3 (Interleaved)
Encode RGB only *	YUV_xxx or Y_ONLY	0	0
Encode Alpha only *	Y_ONLY	3	1
Encode G channel only *	Y_ONLY	1	1

* Not implemented.