DNG SDK 1.2

Generated by Doxygen 1.5.5

Thu Mar 13 04:26:55 2008

Contents

1 Adobe Digital Negative SDK 1.2 1

2 doc_dng_validate 2

3 Class Index 4

4 Class Index 6

5 File Index 9

6 Class Documentation 12

7 File Documentation 171

1 Adobe Digital Negative SDK 1.2

1.1 Introduction

Digital Negative (DNG) is a non-proprietary file format for camera raw image data and metadata. A wide variety of cameras and sensor types are supported by DNG, using the same documented file layout.

This SDK provides support for reading and writing DNG files as well as support for converting DNG data into a displayable or processible image. This SDK is intended to serve as a starting point for adding DNG support to existing applications that use and manipulate images.

1.2 Command line validation: dng_validate

A good place to start investigating the DNG SDK is the dng_validate command line tool, which can read, validate and convert an existing DNG file. The dng_validate.cpp file demonstrates a number of common uses of the SDK. Documentation for the tool can be found here.

1.3 Starting points

- dng_host Used to customize memory allocation, to communicate progress updates and test for cancellation.

- dng_negative Main container for metadata and image data in a DNG file.
- dng_image Class used to hold and manipualte image data.
- dng_render Class used to convert DNG RAW data to displayable image data.
- dng_image_writer Class used to write DNG files.

1.4 Related documentation

- The Adobe Digital Negative specification: http://www.adobe.com/products/dng/pdfs/dng_spec.pdf
- TIFF 6 specification: http://partners.adobe.com/public/developer/tiff/index.html
- TIFF/EP specification: http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetai
- EXIF specification: http://www.jeita.or.jp/english/standard/html/1_-4.htm
- IPTC specification: http://www.iptc.org/IPTC7901/

2 doc_dng_validate

dng_validate Version 1.2 12-Mar-08

"dng_validate" is a command-line tool that parses the tag structure of DNG (and other TIFF-EP based format) files, and reports any deviations from the DNG specification that it finds.

The usage syntax is:

dng_validate [-v] [-d <number>] [-f] [-b4] [-s < CFA index >] [-q <target-binned-width>] [-cs1|-cs2|-cs3|-cs4|-cs5|-cs6] [-16] { [-1 <stage1-out-filename>] [-2 <stage2-out-filename>] [-3 <stage3-out-filename>] [-tif <TIFF-out-filename>] [-dng <DNG-out-file>] {<list of files>}*

Any deviations from the DNG specification are written to the standard error stream.

The "-v" option turns on "verbose" mode, which writes the parsed tag structure to the standard output stream. Any tags that are not parsed by this tool are preceded by an asterisk.

The "-d <number>" option both implies verbose mode, and also specifies the maximum number of lines of data displayed per tag.

The "-f" option switches dng_validate to using floating-point math where possible, instead of the default 16-bit integer.

The "-b4" option causes the demosaic algorithm to produce a four-channel output rather than a three-channel one. (The input DNG must be a three-channel Bayer pattern image.) This option is only useful when used with the -3 switch. The extra channel is the result of doing two interpolations of the Bayer green channel such that the greens on the same row as the reds produces one channel and the greens on the same row as the blues produce another channel. The second green channel will be the highest numbered channel in the output. This option is used to gauge the difference between greens in each row to decide whether the DNG BayerGreenSplit tag should be used for a given source of image data (e.g. camera).

The "-s <CFA index>" option chooses which set of color filter arrays to use when there are multiple ones for an input image. Each CFA array is a separate channel in the DNG input. This applies to the Fuji SR cameras for example, where the first channel is from the S-sensing elements and the second channel is from the R-sensing elements. The S elements are more sensitive and the R elements are less so with the goal of using both to increase the dynamic range the sensor can capture in a single image. By default dng_validate generates an image from only the S-sensors. By using "-s 1" the R-sensing elementsí data can be used to construct the output image. (This index is 0-based. The default is 0.)

The "-q <target-binned-size>" option enables binning during the demosaic process. This is useful for creating previews or thumbnails. The binning factor is determined from the target-binned-size, which is the size in pixels of the larger dimension of the image that is desired. An integer binning factor will be computed to produce an image of that size or larger. For example, if the input image is 3008 x 2000 pixels and the target-binned-width is 700, factor of 4 binning will be used and the result output image (after demosaicing) will be 752 x 500 pixels.

The "-cs1" option generates the output image in sRGB color space.

The "-cs2" option generates the output image in AdobeRGB color space.

The "-cs3" option generates the output image in ProPhotoRGB color space.

The "-cs4" option generates the output image in ColorMatch color space.

The "-cs5" option generates the output image in grayscale gamma 1.8 color space.

The "-cs6" option generates the output image in a grayscale gamma 2.2 color space.

The "-16" option causes dng_validate to output 16-bit-per-component images rather than the default 8-bit.

The "-1" option causes the unprocessed raw image data to be written to the named output file. This applies only to the next input file after the switch.

The "-2" option causes the image data after linearization and black/white level mapping to be written to the named output file. This applies only to the next input file after the switch.

The "-3" option causes the image data after demosaic processing, but prior to color space conversion, noise reduction, sharpening, etc., to be written to the named output

file. This applies only to the next input file after the switch.

The "-tif" option causes the final rendered image to be written as TIFF to the named output file. This applies only to the next input file after the switch.

The "-dng" option causes the parsed DNG data to be reserialized and written to the named output file. This mostly serves to provide an example code path for the process of writing a DNG file, as the output may not differ significantly from the input DNG. (Parameters, such as whether the data is compressed or not, may vary between the input and output DNG files.) This applies only to the next input file after the switch.

3 Class Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AutoPtr< T > 12

dng_1d_function 16

- **dng_1d_concatenate** 14
- **dng_1d_identity** 18
- **dng_1d_inverse** 19
- **dng_function_exposure_ramp** 58
- **dng_function_exposure_tone** 59
- **dng_function_gamma_encode** 60
- **dng_function_GammaEncode_1_8** 61
- **dng_function_GammaEncode_2_2** 63
- **dng_function_GammaEncode_sRGB** 64
- **dng_tone_curve_acr3_default** 170

dng_1d_table 21

dng_abort_sniffer 22

dng_area_task 24

dng_filter_task 54

dng_camera_profile 30

dng_color_space 37

dng_space_AdobeRGB 146

dng_space_ColorMatch 147

dng_space_GrayGamma18 148

dng_space_GrayGamma22 149

dng_space_ProPhoto 150

dng_space_sRGB 150

dng_color_spec 39

dng_date_time 42

dng_date_time_info 45

dng_date_time_storage_info 45

dng_exception 48

dng_exif 49

dng_fingerprint 57

dng_host 66

dng_ifd 73

dng_image 76

dng_simple_image 143

dng_image_writer 82

dng_info 85

dng_iptc 87

dng_linearization_info 90

dng_memory_allocator 94

dng_memory_block 95

dng_memory_data 100

dng_mosaic_info 109

dng_negative 114

dng_pixel_buffer 125

dng_tile_buffer 168

dng_const_tile_buffer 41

dng_dirty_tile_buffer 47

dng_render 139

dng_sniffer_task 144

dng_stream 151

dng_file_stream 53

dng_memory_stream 107

dng_time_zone 170

4 Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AutoPtr< T > (A class intended to be used in stack scope to hold a pointer from new. The held pointer will be deleted automatically if the scope is left without calling Release on the AutoPtr first) 12

dng_1d_concatenate (A dng_1d_function that represents the composition (curry) of two other dng_1d_functions) 14

dng_1d_function (A 1D floating-point function) 16

dng_1d_identity (An identity (x -> y such that x == y for all x) mapping function) 18

dng_1d_inverse (A dng_1d_function that represents the inverse of another dng_1d_function) 19

dng_1d_table (A 1D floating-point lookup table using linear interpolation) 21

dng_abort_sniffer (Class for signaling user cancellation and receiving progress updates) 22

dng_area_task (Abstract class for rectangular processing operations with support for partitioning across multiple processing resources and observing memory constraints) 24

dng_camera_profile (Container for DNG camera color profile and calibration data) 30

dng_color_space (An abstract color space) 37

dng_color_spec 39

dng_const_tile_buffer (Class to get resource acquisition is instantiation behavior for constant (read-only) tile buffers) 41

dng_date_time (Class for holding a date/time and converting to and from relevant date/time formats) 42

dng_date_time_info (Class for holding complete data/time/zone information) 45

dng_date_time_storage_info (Store file offset from which date was read) 45

dng_dirty_tile_buffer (Class to get resource acquisition is instantiation behavior for dirty (writable) tile buffers) 47

dng_exception (All exceptions thrown by the DNG SDK use this exception class) 48

dng_exif (Container class for parsing and holding EXIF tags) 49

dng_file_stream (A stream to/from a disk file. See dng_stream for read/write interface) 53

dng_filter_task (Represents a task which filters an area of a source dng_image to an area of a destination dng_image) 54

dng_fingerprint (Container fingerprint (MD5 only at present)) 57

dng_function_exposure_ramp (Curve for pre-exposure-compensation adjustment based on noise floor, shadows, and highlight level) 58

dng_function_exposure_tone (Exposure compensation curve for a given compensation amount in stops using quadric for roll-off) 59

dng_function_gamma_encode (Encoding gamma curve for a given color space) 60

dng_function_GammaEncode_1_8 (A dng_1d_function for gamma encoding with 1.8 gamma) 61

dng_function_GammaEncode_2_2 (A dng_1d_function for gamma encoding with 2.2 gamma) 63

dng_function_GammaEncode_sRGB (A dng_1d_function for gamma encoding in sRGB color space) 64

dng_host (The main class for communication between the application and the DNG SDK. Used to customize memory allocation and other behaviors) 66

dng_ifd (Container for a single image file directory of a digital negative) 73

dng_image (Base class for holding image data in DNG SDK. See dng_simple_image for derived class most often used in DNG SDK) 76

dng_image_writer (Support for writing dng_image or dng_negative instances to a dng_stream in TIFF or DNG format) 82

dng_info (Top-level structure of DNG file with access to metadata) 85

dng_iptc (Class for reading and holding IPTC metadata associated with a DNG file) 87

dng_linearization_info (Class for managing data values related to DNG linearization) 90

dng_memory_allocator (Interface for dng_memory_block allocator) 94

dng_memory_block (Class to provide resource acquisition is instantiation discipline for image buffers and other larger memory allocations) 95

dng_memory_data (Class to provide resource acquisition is instantiation discipline for small memory allocations) 100

dng_memory_stream (A dng_stream which can be read from or written to memory) 107

dng_mosaic_info (Support for describing color filter array patterns and manipulating mosaic sample data) 109

dng_negative (Main class for holding DNG image data and associated metadata) 114

dng_pixel_buffer (Holds a buffer of pixel data with "pixel geometry" metadata) 125

dng_render (Class used to render digital negative to displayable image) 139

dng_simple_image (Dng_image derived class with simple Trim and Rotate functionality) 143

dng_sniffer_task (Class to establish scope of a named subtask in DNG processing) 144

dng_space_AdobeRGB (Singleton class for AdobeRGB color space) 146

dng_space_ColorMatch (Singleton class for ColorMatch color space) 147

dng_space_GrayGamma18 (Singleton class for gamma 1.8 grayscale color space) 148

dng_space_GrayGamma22 (Singleton class for gamma 2.2 grayscale color space) 149

dng_space_ProPhoto (Singleton class for ProPhoto RGB color space) 150

dng_space_sRGB (Singleton class for sRGB color space) 150

dng_stream 151

dng_tile_buffer (Class to get resource acquisition is instantiation behavior for tile buffers. Can be dirty or constant tile access) 168

dng_time_zone (Class for holding a time zone) 170

dng_tone_curve_acr3_default (Default ACR3 tone curve) 170

5 File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

dng_1d_function.h 171

dng_1d_table.h 172

dng_abort_sniffer.h 172

dng_area_task.h 172

dng_assertions.h 173

dng_auto_ptr.h 173

dng_bottlenecks.h 174

dng_camera_profile.h 179

dng_classes.h ??

dng_color_space.h 180

dng_color_spec.h 181

dng_date_time.h 182

dng_errors.h 183

dng_exceptions.h 184

dng_exif.h 186

dng_fast_module.h 186

dng_file_stream.h 186

dng_filter_task.h 187

dng_fingerprint.h 187

dng_flags.h 187

dng_globals.h 188

dng_host.h 188

dng_hue_sat_map.h ??

dng_ifd.h 188

dng_image.h 189

dng_image_writer.h 189

dng_info.h 190

dng_iptc.h 190

dng_linearization_info.h 191

dng_lossless_jpeg.h ??

dng_matrix.h 191

dng_memory.h ??

dng_memory_stream.h 192

dng_mosaic_info.h 192

dng_mutex.h ??

dng_negative.h 192

dng_orientation.h ??

dng_parse_utils.h ??

dng_pixel_buffer.h 193

dng_point.h ??

dng_preview.h ??

dng_pthread.h ??

dng_rational.h ??

dng_read_image.h 193

dng_rect.h ??

dng_reference.h ??

dng_render.h 194

dng_resample.h ??

dng_sdk_limits.h 194

dng_shared.h ??

dng_simple_image.h ??

dng_spline.h ??

dng_stream.h ??

dng_string.h ??

dng_string_list.h ??

dng_tag_codes.h ??

dng_tag_types.h ??

dng_tag_values.h ??

dng_temperature.h ??

dng_tile_iterator.h ??

dng_tone_curve.h ??

dng_types.h ??

dng_utils.h ??

dng_xmp.h ??

dng_xmp_sdk.h ??

dng_xy_coord.h ??

6 Class Documentation

6.1 AutoPtr< T > Class Template Reference

A class intended to be used in stack scope to hold a pointer from new. The held pointer will be deleted automatically if the scope is left without calling Release on the AutoPtr first.

`#include <dng_auto_ptr.h>`

Public Member Functions

- AutoPtr ()

  Construct an AutoPtr with no referent.

- AutoPtr (T *p)
- ~AutoPtr ()

  Reset() is called on destruction.

- void Alloc ()

  Call Reset with a pointer from new. Uses T's default constructor.

- T * Get () const

  Return the owned pointer of this AutoPtr, NULL if none. No change in ownership or other effects occur.

- T * Release ()

  Return the owned pointer of this AutoPtr, NULL if none. The AutoPtr gives up ownership and takes NULL as its value.

- void Reset (T *p)

  If a pointer is owned, it is deleted. Ownership is taken of passed in pointer.

- void Reset ()

  If a pointer is owned, it is deleted and the AutoPtr takes NULL as its value.

- T * operator → () const

  Allows members of the owned pointer to be accessed directly. It is an error to call this if the AutoPtr has NULL as its value.

- T & operator* () const

  Returns a reference to the object that the owned pointer points to. It is an error to call this if the AutoPtr has NULL as its value.

6.1.1 Detailed Description

template<class T> class AutoPtr< T >

A class intended to be used in stack scope to hold a pointer from new. The held pointer will be deleted automatically if the scope is left without calling Release on the AutoPtr first.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 template<class T> AutoPtr< T >::AutoPtr (T * *p*) [inline, explicit]

Construct an AutoPtr which owns the argument pointer.

Parameters:

p pointer which constructed AutoPtr takes ownership of. p will be deleted on destruction or Reset unless Release is called first.

The documentation for this class was generated from the following file:

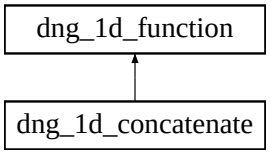
- dng_auto_ptr.h

6.2 dng_1d_concatenate Class Reference

A dng_1d_function that represents the composition (curry) of two other dng_1d_-functions.

`#include <dng_1d_function.h>`

Inheritance diagram for dng_1d_concatenate::



Public Member Functions

- dng_1d_concatenate (const dng_1d_function &function1, const dng_1d_-function &function2)
- virtual bool IsIdentity () const

  Only true if both function1 and function2 have IsIdentity equal to true.

- virtual real64 Evaluate (real64 x) const
- virtual real64 EvaluateInverse (real64 y) const

Protected Attributes

- const dng_1d_function & **fFunction1**
- const dng_1d_function & **fFunction2**

6.2.1 Detailed Description

A dng_1d_function that represents the composition (curry) of two other dng_1d_-functions.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 dng_1d_concatenate::dng_1d_concatenate (const dng_1d_function & *function1*, const dng_1d_function & *function2*)

Create a dng_1d_function which computes y = function2.Evaluate(function1.Evaluate(x)). Compose function1 and function2 to compute y = function2.Evaluate(function1.Evaluate(x)). The range of function1.Evaluate must be a subset of 0.0 to 1.0 inclusive, otherwise the result of function1(x) will be pinned (clipped) to 0.0 if <0.0 and to 1.0 if > 1.0 .

Parameters:

function1 Inner function of composition.

function2 Outer function of composition.

6.2.3 Member Function Documentation

6.2.3.1 real64 dng_1d_concatenate::Evaluate (real64 *x*) const [virtual]

Return the composed mapping for value x.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

function2.Evaluate(function1.Evaluate(x)).

Implements dng_1d_function.

References dng_1d_function::Evaluate().

6.2.3.2 real64 dng_1d_concatenate::EvaluateInverse (real64 *y*) const [virtual]

Return the reverse mapped value for y. Be careful using this method with compositions where the inner function does not have a range 0.0 to 1.0 . (Or better yet, do not use such functions.)

Parameters:

y A value to reverse map. Should be within the range of function2.Evaluate.

Return values:

A value x such that function2.Evaluate(function1.Evaluate(x)) == y (to very close approximation).

Reimplemented from dng_1d_function.

References dng_1d_function::EvaluateInverse().

The documentation for this class was generated from the following files:

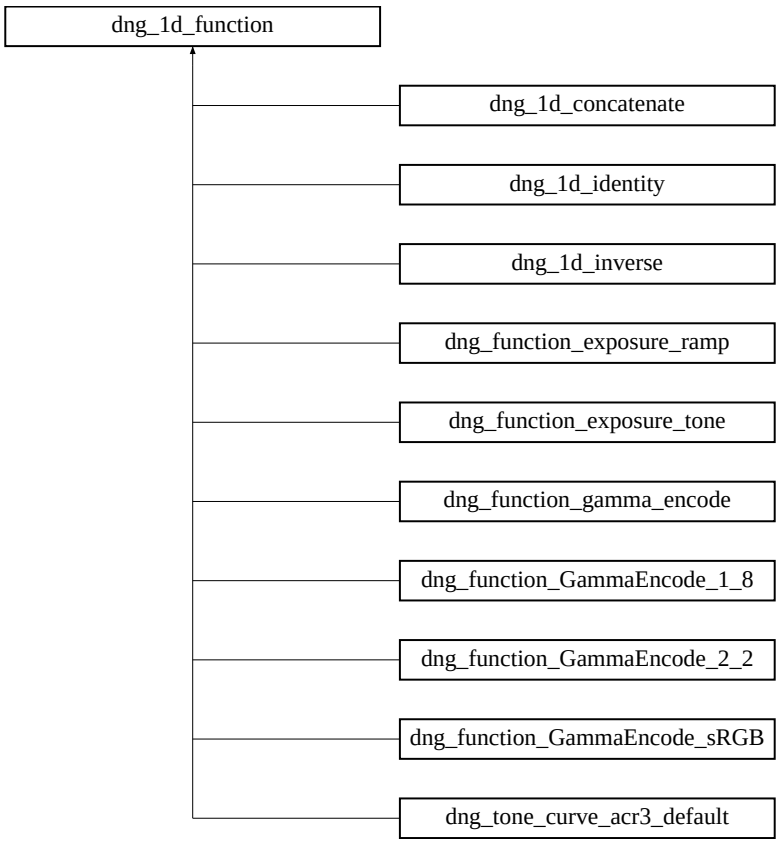
- dng_1d_function.h
- dng_1d_function.cpp

6.3 dng_1d_function Class Reference

A 1D floating-point function.

`#include <dng_1d_function.h>`

Inheritance diagram for dng_1d_function::



Public Member Functions

- virtual bool IsIdentity () const

  Returns true if this function is the map x -> y such that x == y for all x . That is if Evaluate(x) == x for all x.

- virtual real64 Evaluate (real64 x) const =0
- virtual real64 EvaluateInverse (real64 y) const

6.3.1 Detailed Description

A 1D floating-point function.

The domain (input) is always from 0.0 to 1.0, while the range (output) can be an arbitrary interval.

6.3.2 Member Function Documentation

6.3.2.1 virtual real64 dng_1d_function::Evaluate (real64 *x*) const [pure virtual]

Return the mapping for value x. This method must be implemented by a derived class of dng_1d_function and the derived class determines the lookup method and function used.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

Mapped value for x

Implemented in dng_1d_identity, dng_1d_concatenate, dng_1d_inverse, dng_function_GammaEncode_sRGB, dng_function_GammaEncode_1_8, dng_function_GammaEncode_2_2, dng_function_exposure_ramp, dng_function_exposure_tone, dng_tone_curve_acr3_default, and dng_function_gamma_encode.

Referenced by dng_1d_concatenate::Evaluate(), dng_1d_inverse::EvaluateInverse(), EvaluateInverse(), dng_color_space::GammaEncode(), and dng_1d_table::Initialize().

6.3.2.2 real64 dng_1d_function::EvaluateInverse (real64 *y*) const [virtual]

Return the reverse mapped value for y. This method can be implemented by derived classes. The default implementation uses Newton's method to solve for x such that Evaluate(x) == y.

Parameters:

y A value to reverse map. Should be within the range of the function implemented by this dng_1d_function .

Return values:

A value x such that Evaluate(x) == y (to very close approximation).

Reimplemented in dng_1d_identity, dng_1d_concatenate, dng_1d_inverse, dng_function_GammaEncode_sRGB, dng_function_GammaEncode_1_8, dng_function_GammaEncode_2_2, and dng_tone_curve_acr3_default.

References Evaluate().

Referenced by dng_1d_inverse::Evaluate(), dng_function_GammaEncode_2_2::EvaluateInverse(), dng_function_GammaEncode_1_8::EvaluateInverse(), dng_1d_concatenate::EvaluateInverse(), and dng_color_space::GammaDecode().

The documentation for this class was generated from the following files:

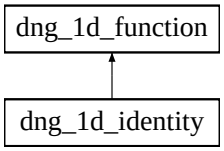
- dng_1d_function.h
- dng_1d_function.cpp

6.4 dng_1d_identity Class Reference

An identity (x -> y such that x == y for all x) mapping function.

`#include <dng_1d_function.h>`

Inheritance diagram for dng_1d_identity::



Public Member Functions

- virtual bool IsIdentity () const

  Always returns true for this class.

- virtual real64 Evaluate (real64 x) const

  Always returns x for this class.

- virtual real64 EvaluateInverse (real64 y) const

  Always returns y for this class.

Static Public Member Functions

- static const dng_1d_function & Get ()

  This class is a singleton, and is entirely threadsafe. Use this method to get an instance of the class.

6.4.1 Detailed Description

An identity (x -> y such that x == y for all x) mapping function.

The documentation for this class was generated from the following files:

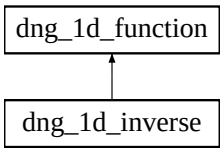
- dng_1d_function.h
- dng_1d_function.cpp

6.5 dng_1d_inverse Class Reference

A dng_1d_function that represents the inverse of another dng_1d_function.

`#include <dng_1d_function.h>`

Inheritance diagram for dng_1d_inverse::



Public Member Functions

- **dng_1d_inverse** (const dng_1d_function &f)
- virtual bool IsIdentity () const

  Returns true if this function is the map x -> y such that x == y for all x . That is if Evaluate(x) == x for all x.

- virtual real64 Evaluate (real64 x) const
- virtual real64 EvaluateInverse (real64 y) const

Protected Attributes

- const dng_1d_function & **fFunction**

6.5.1 Detailed Description

A dng_1d_function that represents the inverse of another dng_1d_function.

6.5.2 Member Function Documentation

6.5.2.1 real64 dng_1d_inverse::Evaluate (real64 *x*) const [virtual]

Return the mapping for value x. This method must be implemented by a derived class of dng_1d_function and the derived class determines the lookup method and function used.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

Mapped value for x

Implements dng_1d_function.

References dng_1d_function::EvaluateInverse().

6.5.2.2 real64 dng_1d_inverse::EvaluateInverse (real64 *y*) const [virtual]

Return the reverse mapped value for y. This method can be implemented by derived classes. The default implementation uses Newton's method to solve for x such that Evaluate(x) == y.

Parameters:

y A value to reverse map. Should be within the range of the function implemented by this dng_1d_function .

Return values:

A value x such that Evaluate(x) == y (to very close approximation).

Reimplemented from dng_1d_function.

References dng_1d_function::Evaluate().

The documentation for this class was generated from the following files:

- dng_1d_function.h
- dng_1d_function.cpp

6.6 dng_1d_table Class Reference

A 1D floating-point lookup table using linear interpolation.

`#include <dng_1d_table.h>`

Public Types

- enum { **kTableBits** = 12, **kTableSize** = (1 << kTableBits) }

  Constants denoting size of table.

Public Member Functions

- void Initialize (dng_memory_allocator &allocator, const dng_1d_function &function, bool subSample=false)
- real32 Interpolate (real32 x) const
- const real32 * Table () const

  Direct access function for table data.

- void Expand16 (uint16 *table16) const

  Expand the table to a 16-bit to 16-bit table.

Protected Attributes

- AutoPtr< dng_memory_block > **fBuffer**
- real32 * **fTable**

6.6.1 Detailed Description

A 1D floating-point lookup table using linear interpolation.

6.6.2 Member Function Documentation

6.6.2.1 void dng_1d_table::Initialize (dng_memory_allocator & *allocator*, const dng_1d_function & *function*, bool *subSample* = `false`)

Set up table, initialize entries using functiion. This method can throw an exception, e.g. if there is not enough memory.

Parameters:

allocator Memory allocator from which table memory is allocated.

function Table is initialized with values of finction.Evalluate(0.0) to function.Evaluate(1.0).

subSample If true, only sample the function a limited number of times (257) and interpolate.

References dng_memory_allocator::Allocate(), dng_1d_function::Evaluate(), and AutoPtr< T >::Reset().

6.6.2.2 real32 dng_1d_table::Interpolate (real32 *x*) const [inline]

Lookup and interpolate mapping for an input.

Parameters:

x value from 0.0 to 1.0 used as input for mapping

Return values:

Approximation of function.Evaluate(x)

References DNG_ASSERT.

The documentation for this class was generated from the following files:

- dng_1d_table.h
- dng_1d_table.cpp

6.7 dng_abort_sniffer Class Reference

Class for signaling user cancellation and receiving progress updates.

`#include <dng_abort_sniffer.h>`

Static Public Member Functions

- static void SniffForAbort (dng_abort_sniffer *sniffer)

Protected Member Functions

- virtual void Sniff ()=0

  Should be implemented by derived classes to check for an user cancellation.

- virtual void StartTask (const char *name, real64 fract)
- virtual void EndTask ()

  Signals the end of the innermost task that has been started.

- virtual void UpdateProgress (real64 fract)

Friends

- class dng_sniffer_task

6.7.1 Detailed Description

Class for signaling user cancellation and receiving progress updates.

DNG SDK clients should derive a host application specific implementation from this class.

6.7.2 Member Function Documentation

6.7.2.1 static void dng_abort_sniffer::SniffForAbort (dng_abort_sniffer * *sniffer*) [inline, static]

Check for pending user cancellation or other abort. ThrowUserCanceled will be called if one is pending. This static method is provided as a convenience for quickly testing for an abort and throwing an exception if one is pending.

Parameters:

sniffer The dng_sniffer to test for a pending abort. Can be NULL, in which case there an abort is never signalled.

References Sniff().

Referenced by dng_stream::Flush(), dng_stream::Get(), dng_area_task::ProcessOnThread(), dng_stream::Put(), and dng_host::SniffForAbort().

6.7.2.2 void dng_abort_sniffer::StartTask (const char * *name*, real64 *fract*) [protected, virtual]

Signals the start of a named task withn processing in the DNG SDK. Tasks may be nested.

Parameters:

name of the task

fract Percentage of total processing this task is expected to take. From 0.0 to 1.0 .

Referenced by dng_sniffer_task::dng_sniffer_task().

6.7.2.3 void dng_abort_sniffer::UpdateProgress (real64 *fract*) [protected, virtual]

Signals progress made on current task.

Parameters:

fract percentage of processing completed on current task. From 0.0 to 1.0 .

Referenced by dng_sniffer_task::UpdateProgress().

The documentation for this class was generated from the following files:

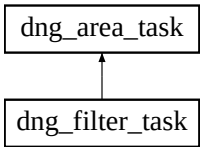
- dng_abort_sniffer.h
- dng_abort_sniffer.cpp

6.8 dng_area_task Class Reference

Abstract class for rectangular processing operations with support for partitioning across multiple processing resources and observing memory constraints.

`#include <dng_area_task.h>`

Inheritance diagram for dng_area_task::



Public Member Functions

- virtual uint32 MaxThreads () const
- virtual uint32 MinTaskArea () const
- virtual dng_point UnitCell () const
- virtual dng_point MaxTileSize () const
- virtual dng_rect RepeatingTile1 () const
- virtual dng_rect RepeatingTile2 () const
- virtual dng_rect RepeatingTile3 () const
- virtual void Start (uint32 threadCount, const dng_point &tileSize, dng_memory_allocator *allocator, dng_abort_sniffer *sniffer)

- virtual void Process (uint32 threadIndex, const dng_rect &tile, dng_abort_sniffer *sniffer)=0
- virtual void Finish (uint32 threadCount)
- dng_point FindTileSize (const dng_rect &area) const
- void ProcessOnThread (uint32 threadIndex, const dng_rect &area, const dng_point &tileSize, dng_abort_sniffer *sniffer)

Static Public Member Functions

- static void Perform (dng_area_task &task, const dng_rect &area, dng_memory_allocator *allocator, dng_abort_sniffer *sniffer)

Protected Attributes

- uint32 **fMaxThreads**
- uint32 **fMinTaskArea**
- dng_point **fUnitCell**
- dng_point **fMaxTileSize**

6.8.1 Detailed Description

Abstract class for rectangular processing operations with support for partitioning across multiple processing resources and observing memory constraints.

6.8.2 Member Function Documentation

6.8.2.1 virtual uint32 dng_area_task::MaxThreads () const [inline, virtual]

Getter for the maximum number of threads (resources) that can be used for processing

Return values:

Number of threads, minimum of 1, that can be used for this task.

6.8.2.2 virtual uint32 dng_area_task::MinTaskArea () const [inline, virtual]

Getter for minimum area of a partitioned rectangle. Often it is not profitable to use more resources if it requires partitioning the input into chunks that are too small, as the overhead increases more than the speedup. This method can be ovreridden for a specific task to indicate the smallest area for partitioning. Default is 256x256 pixels.

Return values:

Minimum area for a partitoned tile in order to give performant operation. (Partitions can be smaller due to small inputs and edge cases.)

6.8.2.3 virtual dng_point dng_area_task::UnitCell () const [inline, virtual]

Getter for dimensions of which partitioned tiles should be a multiple. Various methods of processing prefer certain alignments. The partitioning attempts to construct tiles such that the sizes are a multiple of the dimensions of this point.

Return values:

a point giving preferred alignment in x and y

Referenced by FindTileSize().

6.8.2.4 virtual dng_point dng_area_task::MaxTileSize () const [inline, virtual]

Getter for maximum size of a tile for processing. Often processing will need to allocate temporary buffers or use other resources that are either fixed or in limited supply. The maximum tile size forces further partitioning if the tile is bigger than this size.

Return values:

Maximum tile size allowed for this area task.

Referenced by FindTileSize().

6.8.2.5 dng_rect dng_area_task::RepeatingTile1 () const [virtual]

Getter for RepeatingTile1. RepeatingTile1, RepeatingTile2, and RepeatingTile3 are used to establish a set of 0 to 3 tile patterns for which the resulting partitions that the final Process method is called on will not cross tile boundaries in any of the tile patterns. This can be used for a processing routine that needs to read from two tiles and write to a third such that all the tiles are aligned and sized in a certain way. A RepeatingTile value is valid if it is non-empty. Higher numbered RepeatingTile patterns are only used if all lower ones are non-empty. A RepeatingTile pattern must be a multiple of UnitCell in size for all constraints of the partitionerr to be met.

Referenced by FindTileSize(), and ProcessOnThread().

6.8.2.6 dng_rect dng_area_task::RepeatingTile2 () const [virtual]

Getter for RepeatingTile2. RepeatingTile1, RepeatingTile2, and RepeatingTile3 are used to establish a set of 0 to 3 tile patterns for which the resulting partitions that the final Process method is called on will not cross tile boundaries in any of the tile patterns. This can be used for a processing routine that needs to read from two tiles and write to a third such that all the tiles are aligned and sized in a certain way. A RepeatingTile value is valid if it is non-empty. Higher numbered RepeatingTile patterns are only used if all lower ones are non-empty. A RepeatingTile pattern must be a multiple of UnitCell in size for all constraints of the partitionerr to be met.

Referenced by FindTileSize(), and ProcessOnThread().

6.8.2.7 dng_rect dng_area_task::RepeatingTile3 () const [virtual]

Getter for RepeatingTile3. RepeatingTile1, RepeatingTile2, and RepeatingTile3 are used to establish a set of 0 to 3 tile patterns for which the resulting partitions that the final Process method is called on will not cross tile boundaries in any of the tile patterns. This can be used for a processing routine that needs to read from two tiles and write to a third such that all the tiles are aligned and sized in a certain way. A RepeatingTile value is valid if it is non-empty. Higher numbered RepeatingTile patterns are only used if all lower ones are non-empty. A RepeatingTile pattern must be a multiple of UnitCell in size for all constraints of the partitionerr to be met.

Referenced by FindTileSize(), and ProcessOnThread().

6.8.2.8 void dng_area_task::Start (uint32 *threadCount*, const dng_point & *tileSize*, dng_memory_allocator * *allocator*, dng_abort_sniffer * *sniffer*) [virtual]

Task startup method called before any processing is done on partitions. The Start method is called before any processing is done and can be overridden to allocate temporary buffers, etc.

Parameters:

threadCount Total number of threads that will be used for processing. Less than or equal to MaxThreads.

tileSize Size of source tiles which will be processed. (Not all tiles will be this size due to edge conditions.)

allocator dng_memory_allocator to use for allocating temporary buffers, etc.

sniffer Sniffer to test for user cancellation and to set up progress.

Reimplemented in dng_filter_task.

Referenced by Perform().

6.8.2.9 virtual void dng_area_task::Process (uint32 *threadIndex*, const dng_rect & *tile*, dng_abort_sniffer * *sniffer*) [pure virtual]

Process one tile or fully partitioned area. This method is overridden by derived classes to implement the actual image processing. Note that the sniffer can be ignored if it is certain that a processing task will complete very quickly. This method should never be called directly but rather accessed via Process. There is no allocator parameter as all allocation should be done in Start.

Parameters:

threadIndex 0 to threadCount - 1 index indicating which thread this is. (Can be used to get a thread-specific buffer allocated in the Start method.)

tile Area to process.

sniffer dng_abort_sniffer to use to check for user cancellation and progress updates.

Implemented in dng_filter_task.

Referenced by ProcessOnThread().

6.8.2.10 void dng_area_task::Finish (uint32 *threadCount*) [virtual]

Task computation finalization and teardown method. Called after all resources have completed processing. Can be overridden to accumulate results and free resources allocated in Start.

Parameters:

threadCount Number of threads used for processing. Same as value passed to Start.

Referenced by Perform().

6.8.2.11 dng_point dng_area_task::FindTileSize (const dng_rect & *area*) const

Find tile size taking into account repeating tiles, unit cell, and maximum tile size.

Parameters:

area Computation area for which to find tile size.

Return values:

Tile size as height and width in point.

References MaxTileSize(), RepeatingTile1(), RepeatingTile2(), RepeatingTile3(), and UnitCell().

Referenced by Perform().

6.8.2.12 void dng_area_task::ProcessOnThread (uint32 *threadIndex*, const dng_rect & *area*, const dng_point & *tileSize*, dng_abort_sniffer * *sniffer*)

Handle one resource's worth of partitioned tiles. Called after thread partitioning has already been done. Area may be further subdivided to handle maximum tile size, etc. It will be rare to override this method.

Parameters:

threadIndex 0 to threadCount - 1 index indicating which thread this is.

area Tile area partitioned to this resource.

tileSize

sniffer dng_abort_sniffer to use to check for user cancellation and progress updates.

References Process(), RepeatingTile1(), RepeatingTile2(), RepeatingTile3(), and dng_abort_sniffer::SniffForAbort().

Referenced by Perform().

6.8.2.13 void dng_area_task::Perform (dng_area_task & *task*, const dng_rect & *area*, dng_memory_allocator * *allocator*, dng_abort_sniffer * *sniffer*) [static]

Default resource partitioner that assumes a single resource to be used for processing. Implementations that are aware of multiple processing resources should override (replace) this method. This is usually done in dng_host::PerformAreaTask .

Parameters:

task The task to perform.

area The area on which mage processing should be performed.

allocator dng_memory_allocator to use for allocating temporary buffers, etc.

sniffer dng_abort_sniffer to use to check for user cancellation and progress updates.

References FindTileSize(), Finish(), ProcessOnThread(), and Start().

Referenced by dng_host::PerformAreaTask().

The documentation for this class was generated from the following files:

- dng_area_task.h
- dng_area_task.cpp

6.9 dng_camera_profile Class Reference

Container for DNG camera color profile and calibration data.

`#include <dng_camera_profile.h>`

Public Member Functions

- void SetName (const char *name)
- const dng_string & Name () const
- bool NameIsEmbedded () const
- void SetCalibrationIlluminant1 (uint32 light)
- void SetCalibrationIlluminant2 (uint32 light)
- uint32 CalibrationIlluminant1 () const
- uint32 CalibrationIlluminant2 () const
- real64 CalibrationTemperature1 () const
- real64 CalibrationTemperature2 () const
- void SetColorMatrix1 (const dng_matrix &m)
- void SetColorMatrix2 (const dng_matrix &m)
- bool HasColorMatrix1 () const

  Predicate to test if first camera matrix is set.

- bool HasColorMatrix2 () const

  Predicate to test if second camera matrix is set.

- const dng_matrix & ColorMatrix1 () const

  Getter for first of up to two color matrices used for calibrations.

- const dng_matrix & ColorMatrix2 () const

  Getter for second of up to two color matrices used for calibrations.

- void SetForwardMatrix1 (const dng_matrix &m)

  Setter for first of up to two forward matrices used for calibrations.

- void SetForwardMatrix2 (const dng_matrix &m)

  Setter for second of up to two forward matrices used for calibrations.

- const dng_matrix & ForwardMatrix1 () const

  Getter for first of up to two forward matrices used for calibrations.

- const dng_matrix & ForwardMatrix2 () const

  Getter for second of up to two forward matrices used for calibrations.

- void SetReductionMatrix1 (const dng_matrix &m)
- void SetReductionMatrix2 (const dng_matrix &m)
- const dng_matrix & ReductionMatrix1 () const

  Getter for first of up to two dimensionality reduction hints for four color cameras.

- const dng_matrix & ReductionMatrix2 () const

  Getter for second of up to two dimensionality reduction hints for four color cameras.

- const dng_fingerprint & Fingerprint () const

  Getter function from profile fingerprint.

- dng_camera_profile_id ProfileID () const
- void SetCopyright (const char *copyright)
- const dng_string & Copyright () const
- void **SetEmbedPolicy** (uint32 policy)
- uint32 **EmbedPolicy** () const
- bool **IsLegalToEmbed** () const
- bool **HasHueSatDeltas** () const
- const dng_hue_sat_map & **HueSatDeltas1** () const
- void **SetHueSatDeltas1** (dng_hue_sat_map &deltas1)
- const dng_hue_sat_map & **HueSatDeltas2** () const
- void **SetHueSatDeltas2** (dng_hue_sat_map &deltas2)
- const dng_tone_curve & **ToneCurve** () const
- void **SetToneCurve** (const dng_tone_curve &curve)
- void **SetProfileCalibrationSignature** (const char *signature)
- const dng_string & **ProfileCalibrationSignature** () const
- void SetUniqueCameraModelRestriction (const char *camera)
- const dng_string & UniqueCameraModelRestriction () const
- void **SetWasReadFromDNG** (bool state=true)
- bool **WasReadFromDNG** () const
- bool IsValid (uint32 channels) const
- bool EqualData (const dng_camera_profile &profile) const
- void Parse (dng_stream &stream, dng_camera_profile_info &profileInfo)

  Parse profile from dng_camera_profile_info data.

- bool ParseExtended (dng_stream &stream)
- virtual void SetFourColorBayer ()

  Convert from a three-color to a four-color Bayer profile.

- dng_hue_sat_map * HueSatMapForWhite (const dng_xy_coord &white) const

Static Public Member Functions

- static void NormalizeColorMatrix (dng_matrix &m)

  Utility function to normalize the scale of the color matrix.

- static void NormalizeForwardMatrix (dng_matrix &m)

  Utility function to normalize the scale of the forward matrix.

Protected Member Functions

- void **ClearFingerprint** ()
- void **CalculateFingerprint** () const

Static Protected Member Functions

- static real64 **IlluminantToTemperature** (uint32 light)
- static bool **ValidForwardMatrix** (const dng_matrix &m)
- static void **ReadHueSatMap** (dng_stream &stream, dng_hue_sat_map &hueSatMap, uint32 hues, uint32 sats, uint32 vals, bool skipSat0)

Protected Attributes

- dng_string **fName**
- uint32 **fCalibrationIlluminant1**
- uint32 **fCalibrationIlluminant2**
- dng_matrix **fColorMatrix1**
- dng_matrix **fColorMatrix2**
- dng_matrix **fForwardMatrix1**
- dng_matrix **fForwardMatrix2**
- dng_matrix **fReductionMatrix1**
- dng_matrix **fReductionMatrix2**
- dng_fingerprint **fFingerprint**
- dng_string **fCopyright**
- uint32 **fEmbedPolicy**
- dng_hue_sat_map **fHueSatDeltas1**
- dng_hue_sat_map **fHueSatDeltas2**
- dng_tone_curve **fToneCurve**
- dng_string **fProfileCalibrationSignature**
- dng_string **fUniqueCameraModelRestriction**
- bool **fWasReadFromDNG**

6.9.1 Detailed Description

Container for DNG camera color profile and calibration data.

6.9.2 Member Function Documentation

6.9.2.1 void dng_camera_profile::SetName (const char * *name*) [inline]

Setter for camera profile name.

Parameters:

name Name to use for this camera profile.

Referenced by Parse().

6.9.2.2 const dng_string& dng_camera_profile::Name () const [inline]

Getter for camera profile name.

Return values:

Name of profile.

Referenced by ProfileID().

6.9.2.3 bool dng_camera_profile::NameIsEmbedded () const [inline]

Test if this name is embedded.

Return values:

true if the name matches the name of the embedded camera profile.

6.9.2.4 void dng_camera_profile::SetCalibrationIlluminant1 (uint32 *light*) [inline]

Setter for first of up to two light sources used for calibration. Uses the EXIF encodings for illuminant and is used to distinguish which matrix to use. Corresponds to the DNG CalibrationIlluminant1 tag.

Referenced by Parse().

6.9.2.5 void dng_camera_profile::SetCalibrationIlluminant2 (uint32 *light*) [inline]

Setter for second of up to two light sources used for calibration. Uses the EXIF encodings for illuminant and is used to distinguish which matrix to use. Corresponds to the DNG CalibrationIlluminant2 tag.

Referenced by Parse().

6.9.2.6 uint32 dng_camera_profile::CalibrationIlluminant1 () const [inline]

Getter for first of up to two light sources used for calibration. Uses the EXIF encodings for illuminant and is used to distinguish which matrix to use. Corresponds to the DNG CalibrationIlluminant1 tag.

Referenced by CalibrationTemperature1().

6.9.2.7 uint32 dng_camera_profile::CalibrationIlluminant2 () const [inline]

Getter for second of up to two light sources used for calibration. Uses the EXIF encodings for illuminant and is used to distinguish which matrix to use. Corresponds to the DNG CalibrationIlluminant2 tag.

Referenced by CalibrationTemperature2().

6.9.2.8 real64 dng_camera_profile::CalibrationTemperature1 () const [inline]

Getter for first of up to two light sources used for calibration, returning result as color temperature.

References CalibrationIlluminant1().

Referenced by dng_color_spec::dng_color_spec(), and HueSatMapForWhite().

6.9.2.9 real64 dng_camera_profile::CalibrationTemperature2 () const [inline]

Getter for second of up to two light sources used for calibration, returning result as color temperature.

References CalibrationIlluminant2().

Referenced by dng_color_spec::dng_color_spec(), and HueSatMapForWhite().

6.9.2.10 void dng_camera_profile::SetColorMatrix1 (const dng_matrix & *m*)

Setter for first of up to two color matrices used for reference camera calibrations. These matrices map XYZ values to camera values. The DNG SDK needs to map colors that direction in order to determine the clipping points for highlight recovery logic based on

the white point. If cameras were all three-color, the matrix could be stored as a forward matrix. The inverse matrix is requried to support four-color cameras.

References NormalizeColorMatrix().

Referenced by Parse().

6.9.2.11 void dng_camera_profile::SetColorMatrix2 (const dng_matrix & *m*)

Setter for second of up to two color matrices used for reference camera calibrations. These matrices map XYZ values to camera values. The DNG SDK needs to map colors that direction in order to determine the clipping points for highlight recovery logic based on the white point. If cameras were all three-color, the matrix could be stored as a forward matrix. The inverse matrix is requried to support four-color cameras.

References NormalizeColorMatrix().

Referenced by Parse().

6.9.2.12 void dng_camera_profile::SetReductionMatrix1 (const dng_matrix & *m*)

Setter for first of up to two dimensionality reduction hints for four-color cameras. This is an optional matrix that maps four components to three. See Appendix 6 of the DNG 1.1.0 specification.

Referenced by Parse().

6.9.2.13 void dng_camera_profile::SetReductionMatrix2 (const dng_matrix & *m*)

Setter for second of up to two dimensionality reduction hints for four-color cameras. This is an optional matrix that maps four components to three. See Appendix 6 of the DNG 1.1.0 specification.

Referenced by Parse().

6.9.2.14 dng_camera_profile_id dng_camera_profile::ProfileID () const [inline]

Getter for camera profile id.

Return values:

ID of profile.

References Fingerprint(), and Name().

6.9.2.15 void dng_camera_profile::SetCopyright (const char * *copyright*) [inline]

Setter for camera profile copyright.

Parameters:

copyright Copyright string to use for this camera profile.

Referenced by Parse().

6.9.2.16 const dng_string& dng_camera_profile::Copyright () const [inline]

Getter for camera profile copyright.

Return values:

Copyright string for profile.

6.9.2.17 void dng_camera_profile::SetUniqueCameraModelRestriction (const char * *camera*) [inline]

Setter for camera unique model name to restrict use of this profile.

Parameters:

camera Camera unique model name designating only camera this profile can be used with. (Empty string for no restriction.)

Referenced by Parse().

6.9.2.18 const dng_string& dng_camera_profile::UniqueCameraModelRestriction () const [inline]

Getter for camera unique model name to restrict use of this profile.

Return values:

Unique model name of only camera this profile can be used with or empty if no restriction.

6.9.2.19 bool dng_camera_profile::IsValid (uint32 *channels*) const

Determines if this a valid profile for this number of color channels?

Return values:

true if the profile is valid.

Referenced by dng_color_spec::dng_color_spec(), dng_info::Parse(), and SetFourColorBayer().

6.9.2.20 bool dng_camera_profile::EqualData (const dng_camera_profile & *profile*) const

Predicate to check if two camera profiles are colorwise equal, thus ignores the profile name.

Parameters:

profile Camera profile to compare to.

References fCalibrationIlluminant1, fCalibrationIlluminant2, fColorMatrix1, fColorMatrix2, fForwardMatrix1, fForwardMatrix2, fHueSatDeltas1, fHueSatDeltas2, fProfileCalibrationSignature, fReductionMatrix1, fReductionMatrix2, and fToneCurve.

6.9.2.21 bool dng_camera_profile::ParseExtended (dng_stream & *stream*)

Parse from an extended profile stream, which is similar to stand alone TIFF file.

References Parse().

6.9.2.22 dng_hue_sat_map * dng_camera_profile::HueSatMapForWhite (const dng_xy_coord & *white*) const

Find the hue/sat table to use for a given white point, if any. The calling routine owns the resulting table.

References CalibrationTemperature1(), and CalibrationTemperature2().

The documentation for this class was generated from the following files:

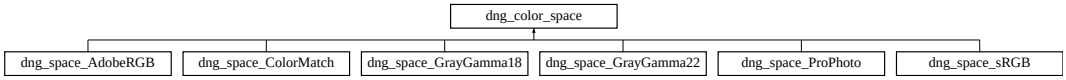
- dng_camera_profile.h
- dng_camera_profile.cpp

6.10 dng_color_space Class Reference

An abstract color space.

`#include <dng_color_space.h>`

Inheritance diagram for dng_color_space::



Public Member Functions

- const dng_matrix & MatrixToPCS () const

  Return a matrix which transforms source data in this color space into the Profile Connection Space.

- const dng_matrix & MatrixFromPCS () const

  Return a matrix which transforms Profile Connection Space data into this color space.

- bool IsMonochrome () const

  Predicate which is true if this color space is monochrome (has only a single column).

- virtual const dng_1d_function & GammaFunction () const

  Getter for the gamma function for this color space.

- bool IsLinear () const

  Returns true if this color space is linear. (I.e. has gamma 1.0.).

- real64 GammaEncode (real64 x) const

  Map an input value through this color space's encoding gamma.

- real64 GammaDecode (real64 y) const

  Map an input value through this color space's decoding gamma (inverse of the encoding gamma).

- virtual bool ICCProfile (uint32 &size, const uint8 *&data) const

Protected Member Functions

- void **SetMonochrome** ()
- void **SetMatrixToPCS** (const dng_matrix_3by3 &M)

Protected Attributes

- dng_matrix **fMatrixToPCS**
- dng_matrix **fMatrixFromPCS**

6.10.1 Detailed Description

An abstract color space.

6.10.2 Member Function Documentation

6.10.2.1 bool dng_color_space::ICCProfile (uint32 & *size*, const uint8 *& *data*) const [virtual]

Getter for ICC profile, if this color space has one.

Parameters:

size Out parameter which receives size on return.

data Receives bytes of profile.

Return values:

Returns true if this color space has an ICC profile, false otherwise.

Reimplemented in dng_space_sRGB, dng_space_AdobeRGB, dng_space_ColorMatch, dng_space_ProPhoto, dng_space_GrayGamma18, and dng_space_GrayGamma22.

The documentation for this class was generated from the following files:

- dng_color_space.h
- dng_color_space.cpp

6.11 dng_color_spec Class Reference

`#include <dng_color_spec.h>`

Public Member Functions

- dng_color_spec (const dng_negative &negative, const dng_camera_profile *profile)
- uint32 Channels () const
- void SetWhiteXY (const dng_xy_coord &white)
- const dng_xy_coord & WhiteXY () const
- const dng_vector & CameraWhite () const
- const dng_matrix & CameraToPCS () const
- dng_xy_coord NeutralToXY (const dng_vector &neutral)

6.11.1 Detailed Description

Color transform taking into account white point and camera calibration and individual calibration from DNG negative.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 dng_color_spec::dng_color_spec (const dng_negative & *negative*, const dng_camera_profile * *profile*)

Read calibration info from DNG negative and construct a dng_color_spec.

References dng_negative::AnalogBalance(), dng_camera_profile::CalibrationTemperature1(), dng_camera_profile::CalibrationTemperature2(), dng_negative::CameraCalibration1(), dng_negative::CameraCalibration2(), dng_camera_profile::ColorMatrix1(), dng_camera_profile::ColorMatrix2(), dng_camera_profile::ForwardMatrix1(), dng_camera_profile::ForwardMatrix2(), dng_camera_profile::HasColorMatrix2(), dng_camera_profile::IsValid(), dng_camera_profile::NormalizeForwardMatrix(), dng_camera_profile::ProfileCalibrationSignature(), dng_camera_profile::ReductionMatrix1(), dng_camera_profile::ReductionMatrix2(), and ThrowBadFormat().

6.11.3 Member Function Documentation

6.11.3.1 uint32 dng_color_spec::Channels () const [inline]

Number of channels used for this color transform. Three for most cameras.

6.11.3.2 void dng_color_spec::SetWhiteXY (const dng_xy_coord & *white*)

Setter for white point. Value is as XY colorspace coordinate.

Parameters:

white White point to set as an XY value.

6.11.3.3 const dng_xy_coord & dng_color_spec::WhiteXY () const

Getter for white point. Value is as XY colorspace coordinate.

Return values:

XY value of white point.

References DNG_ASSERT.

6.11.3.4 const dng_vector & dng_color_spec::CameraWhite () const

Return white point in camera native color coordinates.

Return values:

A dng_vector with components ranging from 0.0 to 1.0 that is normalized such that one component is equal to 1.0 .

References DNG_ASSERT.

6.11.3.5 const dng_matrix & dng_color_spec::CameraToPCS () const

Getter for camera to Profile Connection Space color transform.

Return values:

A transform that takes into account all camera calibration transforms and white point.

References DNG_ASSERT.

6.11.3.6 dng_xy_coord dng_color_spec::NeutralToXY (const dng_vector & *neutral*)

Return the XY value to use for SetWhiteXY for a given camera color space coordinate as the white point.

Parameters:

neutral A camera color space value to use for white point. Components range from 0.0 to 1.0 and should be normalized such that the largest value is 1.0 .

Return values:

White point in XY space that makes neutral map to this XY value as closely as possible.

The documentation for this class was generated from the following files:

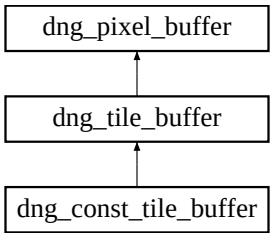
- dng_color_spec.h
- dng_color_spec.cpp

6.12 dng_const_tile_buffer Class Reference

Class to get resource acquisition is instantiation behavior for constant (read-only) tile buffers.

`#include <dng_image.h>`

Inheritance diagram for dng_const_tile_buffer::



Public Member Functions

- dng_const_tile_buffer (const dng_image &image, const dng_rect &tile)

6.12.1 Detailed Description

Class to get resource acquisition is instantiation behavior for constant (read-only) tile buffers.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 dng_const_tile_buffer::dng_const_tile_buffer (const dng_image & *image*, const dng_rect & *tile*)

Obtain a read-only tile from an image.

Parameters:

image Image tile will come from.

tile Rectangle denoting extent of tile.

The documentation for this class was generated from the following files:

- dng_image.h
- dng_image.cpp

6.13 dng_date_time Class Reference

Class for holding a date/time and converting to and from relevant date/time formats.

`#include <dng_date_time.h>`

Public Member Functions

- dng_date_time ()

  Construct an invalid date/time.

- dng_date_time (uint32 year, uint32 month, uint32 day, uint32 hour, uint32 minute, uint32 second)
- bool IsValid () const
- bool NotValid () const
- bool operator== (const dng_date_time &dt) const

  Equal operator.

- bool **operator!=** (const dng_date_time &dt) const
- void Clear ()

  Set date to an invalid value.

- bool Parse (const char *s)

Public Attributes

- uint32 **fYear**
- uint32 **fMonth**
- uint32 **fDay**
- uint32 **fHour**
- uint32 **fMinute**
- uint32 **fSecond**

6.13.1 Detailed Description

Class for holding a date/time and converting to and from relevant date/time formats.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 dng_date_time::dng_date_time (uint32 *year*, uint32 *month*, uint32 *day*, uint32 *hour*, uint32 *minute*, uint32 *second*)

Construct a date/time with specific values.

Parameters:

year Year to use as actual integer value, such as 2006.

month Month to use from 1 - 12, where 1 is January.

day Day of month to use from 1 -31, where 1 is the first.

hour Hour of day to use from 0 - 23, where 0 is midnight.

minute Minute of hour to use from 0 - 59.

second Second of minute to use from 0 - 59.

6.13.3 Member Function Documentation

6.13.3.1 bool dng_date_time::IsValid () const

Predicate to determine if a date is valid.

Return values:

true if all fields are within range.

Referenced by LocalTimeZone(), NotValid(), and Parse().

6.13.3.2 bool dng_date_time::NotValid () const [inline]

Predicate to determine if a date is invalid.

Return values:

true if any field is out of range.

References IsValid().

6.13.3.3 bool dng_date_time::Parse (const char * *s*)

Parse an EXIF format date string.

Parameters:

s Input date string to parse.

Return values:

true if date was parsed successfully and date is valid.

References IsValid().

The documentation for this class was generated from the following files:

- dng_date_time.h
- dng_date_time.cpp

6.14 dng_date_time_info Class Reference

Class for holding complete data/time/zone information.

`#include <dng_date_time.h>`

Public Member Functions

- bool **IsValid** () const
- bool **NotValid** () const
- void **Clear** ()
- const dng_date_time & **DateTime** () const
- void **SetDateTime** (const dng_date_time &dt)
- const dng_string & **Subseconds** () const
- void **SetSubseconds** (const dng_string &s)
- const dng_time_zone & **TimeZone** () const
- void **SetZone** (const dng_time_zone &zone)
- void **Decode_ISO_8601** (const char *s)
- dng_string **Encode_ISO_8601** () const
- void **Decode_IPTC_Date** (const char *s)
- dng_string **Encode_IPTC_Date** () const
- void **Decode_IPTC_Time** (const char *s)
- dng_string **Encode_IPTC_Time** () const

6.14.1 Detailed Description

Class for holding complete data/time/zone information.

The documentation for this class was generated from the following files:

- dng_date_time.h
- dng_date_time.cpp

6.15 dng_date_time_storage_info Class Reference

Store file offset from which date was read.

`#include <dng_date_time.h>`

Public Member Functions

- dng_date_time_storage_info ()

  The default constructor initializes to an invalid state.

- dng_date_time_storage_info (uint64 offset, dng_date_time_format format)

  Construct with file offset and date format.

- bool IsValid () const
- uint64 Offset () const
- dng_date_time_format Format () const

6.15.1 Detailed Description

Store file offset from which date was read.

Used internally by Adobe to update date in original file.

Warning:

Use at your own risk.

6.15.2 Member Function Documentation

6.15.2.1 bool dng_date_time_storage_info::IsValid () const

Predicate to determine if an offset is valid.

Return values:

true if offset is valid.

Referenced by Format(), and Offset().

6.15.2.2 uint64 dng_date_time_storage_info::Offset () const

Getter for offset in file.

Exceptions:

dng_exception with fErrorCode equal to dng_error_unknown if offset is not valid.

References IsValid(), and ThrowProgramError().

6.15.2.3 dng_date_time_format dng_date_time_storage_info::Format () const

Get for format date was originally stored in file. Throws a dng_error_unknown exception if offset is invalid.

Exceptions:

dng_exception with fErrorCode equal to dng_error_unknown if offset is not valid.

References IsValid(), and ThrowProgramError().

The documentation for this class was generated from the following files:

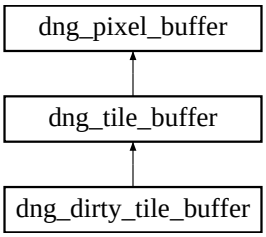
- dng_date_time.h
- dng_date_time.cpp

6.16 dng_dirty_tile_buffer Class Reference

Class to get resource acquisition is instantiation behavior for dirty (writable) tile buffers.

`#include <dng_image.h>`

Inheritance diagram for dng_dirty_tile_buffer::



Public Member Functions

- dng_dirty_tile_buffer (dng_image &image, const dng_rect &tile)

6.16.1 Detailed Description

Class to get resource acquisition is instantiation behavior for dirty (writable) tile buffers.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 dng_dirty_tile_buffer::dng_dirty_tile_buffer (dng_image & *image*, const dng_rect & *tile*)

Obtain a writable tile from an image.

Parameters:

image Image tile will come from.

tile Rectangle denoting extent of tile.

The documentation for this class was generated from the following files:

- dng_image.h
- dng_image.cpp

6.17 dng_exception Class Reference

All exceptions thrown by the DNG SDK use this exception class.

`#include <dng_exceptions.h>`

Public Member Functions

- dng_exception (dng_error_code code)
- dng_error_code ErrorCode () const

6.17.1 Detailed Description

All exceptions thrown by the DNG SDK use this exception class.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 dng_exception::dng_exception (dng_error_code *code*) [inline]

Construct an exception representing the given error code.

Parameters:

code Error code this exception is for.

6.17.3 Member Function Documentation

6.17.3.1 dng_error_code dng_exception::ErrorCode () const [inline]

Getter for error code of this exception

Return values:

The error code of this exception.

The documentation for this class was generated from the following file:

- dng_exceptions.h

6.18 dng_exif Class Reference

Container class for parsing and holding EXIF tags.

`#include <dng_exif.h>`

Public Member Functions

- virtual dng_exif * **Clone** () const
- void **SetExposureTime** (real64 et, bool snap=true)
- void **SetShutterSpeedValue** (real64 ss)
- void **SetFNumber** (real64 fs)
- void **SetApertureValue** (real64 av)
- void **UpdateDateTime** (const dng_date_time_info &dt)
- virtual bool **ParseTag** (dng_stream &stream, dng_shared &shared, uint32 parentCode, bool isMainIFD, uint32 tagCode, uint32 tagType, uint32 tagCount, uint64 tagOffset)
- virtual void **PostParse** (dng_host &host, dng_shared &shared)

Static Public Member Functions

- static real64 **SnapExposureTime** (real64 et)
- static dng_urational **EncodeFNumber** (real64 fs)

Public Attributes

- dng_string **fImageDescription**
- dng_string **fMake**
- dng_string **fModel**
- dng_string **fSoftware**
- dng_string **fArtist**
- dng_string **fCopyright**
- dng_string **fCopyright2**
- dng_string **fUserComment**
- dng_date_time_info **fDateTime**
- dng_date_time_storage_info **fDateTimeStorageInfo**
- dng_date_time_info **fDateTimeOriginal**
- dng_date_time_storage_info **fDateTimeOriginalStorageInfo**

- dng_date_time_info **fDateTimeDigitized**
- dng_date_time_storage_info **fDateTimeDigitizedStorageInfo**
- uint32 **fTIFF_EP_StandardID**
- uint32 **fExifVersion**
- uint32 **fFlashPixVersion**
- dng_urational **fExposureTime**
- dng_urational **fFNumber**
- dng_srational **fShutterSpeedValue**
- dng_urational **fApertureValue**
- dng_srational **fBrightnessValue**
- dng_srational **fExposureBiasValue**
- dng_urational **fMaxApertureValue**
- dng_urational **fFocalLength**
- dng_urational **fDigitalZoomRatio**
- dng_urational **fExposureIndex**
- dng_urational **fSubjectDistance**
- dng_urational **fGamma**
- dng_urational **fBatteryLevelR**
- dng_string **fBatteryLevelA**
- uint32 **fExposureProgram**
- uint32 **fMeteringMode**
- uint32 **fLightSource**
- uint32 **fFlash**
- uint32 **fFlashMask**
- uint32 **fSensingMethod**
- uint32 **fColorSpace**
- uint32 **fFileSource**
- uint32 **fSceneType**
- uint32 **fCustomRendered**
- uint32 **fExposureMode**
- uint32 **fWhiteBalance**
- uint32 **fSceneCaptureType**
- uint32 **fGainControl**
- uint32 **fContrast**
- uint32 **fSaturation**
- uint32 **fSharpness**
- uint32 **fSubjectDistanceRange**
- uint32 **fSelfTimerMode**
- uint32 **fImageNumber**
- uint32 **fFocalLengthIn35mmFilm**
- uint32 **fISOSpeedRatings** [3]
- uint32 **fSubjectAreaCount**
- uint32 **fSubjectArea** [4]

- uint32 **fComponentsConfiguration**
- dng_urational **fCompresssedBitsPerPixel**
- uint32 **fPixelXDimension**
- uint32 **fPixelYDimension**
- dng_urational **fFocalPlaneXResolution**
- dng_urational **fFocalPlaneYResolution**
- uint32 **fFocalPlaneResolutionUnit**
- uint32 **fCFARepeatPatternRows**
- uint32 **fCFARepeatPatternCols**
- uint8 **fCFAPattern** [kMaxCFAPattern][kMaxCFAPattern]
- dng_fingerprint **fImageUniqueID**
- uint32 **fGPSVersionID**
- dng_string **fGPSLatitudeRef**
- dng_urational **fGPSLatitude** [3]
- dng_string **fGPSLongitudeRef**
- dng_urational **fGPSLongitude** [3]
- uint32 **fGPSAltitudeRef**
- dng_urational **fGPSAltitude**
- dng_urational **fGPSTimeStamp** [3]
- dng_string **fGPSSatellites**
- dng_string **fGPSStatus**
- dng_string **fGPSMeasureMode**
- dng_urational **fGPSDOP**
- dng_string **fGPSSpeedRef**
- dng_urational **fGPSSpeed**
- dng_string **fGPSTrackRef**
- dng_urational **fGPSTrack**
- dng_string **fGPSImgDirectionRef**
- dng_urational **fGPSImgDirection**
- dng_string **fGPSMapDatum**
- dng_string **fGPSDestLatitudeRef**
- dng_urational **fGPSDestLatitude** [3]
- dng_string **fGPSDestLongitudeRef**
- dng_urational **fGPSDestLongitude** [3]
- dng_string **fGPSDestBearingRef**
- dng_urational **fGPSDestBearing**
- dng_string **fGPSDestDistanceRef**
- dng_urational **fGPSDestDistance**
- dng_string **fGPSProcessingMethod**
- dng_string **fGPSAreaInformation**
- dng_string **fGPSDateStamp**
- uint32 **fGPSDifferential**
- dng_string **fInteroperabilityIndex**

- uint32 **fInteroperabilityVersion**
- dng_string **fRelatedImageFileFormat**
- uint32 **fRelatedImageWidth**
- uint32 **fRelatedImageLength**
- dng_string **fCameraSerialNumber**
- dng_urational **fLensInfo** [4]
- dng_string **fLensID**
- dng_string **fLensName**
- dng_string **fLensSerialNumber**
- dng_srational **fFlashCompensation**
- dng_string **fOwnerName**
- dng_string **fFirmware**

Protected Member Functions

- virtual bool **Parse_ifd0** (dng_stream &stream, dng_shared &shared, uint32 parentCode, uint32 tagCode, uint32 tagType, uint32 tagCount, uint64 tagOffset)
- virtual bool **Parse_ifd0_main** (dng_stream &stream, dng_shared &shared, uint32 parentCode, uint32 tagCode, uint32 tagType, uint32 tagCount, uint64 tagOffset)
- virtual bool **Parse_ifd0_exif** (dng_stream &stream, dng_shared &shared, uint32 parentCode, uint32 tagCode, uint32 tagType, uint32 tagCount, uint64 tagOffset)

- virtual bool **Parse_gps** (dng_stream &stream, dng_shared &shared, uint32 parentCode, uint32 tagCode, uint32 tagType, uint32 tagCount, uint64 tagOffset)
- virtual bool **Parse_interoperability** (dng_stream &stream, dng_shared &shared, uint32 parentCode, uint32 tagCode, uint32 tagType, uint32 tagCount, uint64 tagOffset)

6.18.1 Detailed Description

Container class for parsing and holding EXIF tags.

Public member fields are documented in EXIF specification.

The documentation for this class was generated from the following files:

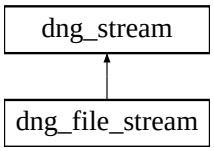
- dng_exif.h
- dng_exif.cpp

6.19 dng_file_stream Class Reference

A stream to/from a disk file. See dng_stream for read/write interface.

`#include <dng_file_stream.h>`

Inheritance diagram for dng_file_stream::



Public Member Functions

- dng_file_stream (const char *filename, bool output=false, uint32 bufferSize=kDefaultBufferSize)

Protected Member Functions

- virtual uint64 **DoGetLength** ()
- virtual void **DoRead** (void *data, uint32 count, uint64 offset)
- virtual void **DoWrite** (const void *data, uint32 count, uint64 offset)

6.19.1 Detailed Description

A stream to/from a disk file. See dng_stream for read/write interface.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 dng_file_stream::dng_file_stream (const char * *filename*, bool *output* = `false`, uint32 *bufferSize* = `kDefaultBufferSize`)

Open a stream on a file.

Parameters:

filename Pathname in platform synax.

output Set to true if writing, false otherwise.

References ThrowOpenFile(), and ThrowSilentError().

The documentation for this class was generated from the following files:

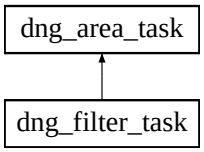
- dng_file_stream.h
- dng_file_stream.cpp

6.20 dng_filter_task Class Reference

Represents a task which filters an area of a source dng_image to an area of a destination dng_image.

`#include <dng_filter_task.h>`

Inheritance diagram for dng_filter_task::



Public Member Functions

- dng_filter_task (const dng_image &srcImage, dng_image &dstImage)
- virtual dng_rect SrcArea (const dng_rect &dstArea)
- virtual dng_point SrcTileSize (const dng_point &dstTileSize)
- virtual void ProcessArea (uint32 threadIndex, dng_pixel_buffer &srcBuffer, dng_pixel_buffer &dstBuffer)=0
- virtual void Start (uint32 threadCount, const dng_point &tileSize, dng_memory_allocator *allocator, dng_abort_sniffer *sniffer)
- virtual void Process (uint32 threadIndex, const dng_rect &area, dng_abort_sniffer *sniffer)

Protected Attributes

- const dng_image & **fSrcImage**
- dng_image & **fDstImage**
- uint32 **fSrcPlane**
- uint32 **fSrcPlanes**
- uint32 **fSrcPixelType**
- uint32 **fDstPlane**
- uint32 **fDstPlanes**
- uint32 **fDstPixelType**
- dng_point **fSrcRepeat**
- AutoPtr< dng_memory_block > **fSrcBuffer** [kMaxMPThreads]
- AutoPtr< dng_memory_block > **fDstBuffer** [kMaxMPThreads]

6.20.1 Detailed Description

Represents a task which filters an area of a source dng_image to an area of a destination dng_image.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 dng_filter_task::dng_filter_task (const dng_image & *srcImage*, dng_image & *dstImage*)

Construct a filter task given a source and destination images.

Parameters:

srcImage Image from which source pixels are read.

dstImage Image to which result pixels are written.

6.20.3 Member Function Documentation

6.20.3.1 virtual dng_rect dng_filter_task::SrcArea (const dng_rect & *dstArea*) [inline, virtual]

Compute the source area needed for a given destination area. Default implementation assumes desitnation area is equal to source area for all cases.

Parameters:

dstArea Area to for which pixels will be computed.

Return values:

The source area needed as input to calculate the requested destination area.

Referenced by Process(), and SrcTileSize().

6.20.3.2 virtual dng_point dng_filter_task::SrcTileSize (const dng_point & *dstTileSize*) [inline, virtual]

Given a destination tile size, calculate input tile size. Simlar to SrcArea, and should seldom be overridden.

Parameters:

dstTileSize The destination tile size that is targeted for output.

Return values:

The source tile size needed to compute a tile of the destination size.

References SrcArea().

Referenced by Start().

6.20.3.3 virtual void dng_filter_task::ProcessArea (uint32 *threadIndex*, dng_pixel_buffer & *srcBuffer*, dng_pixel_buffer & *dstBuffer*) [pure virtual]

Implements filtering operation from one buffer to another. Source and destination pixels are set up in member fields of this class. Ideally, no allocation should be done in this routine.

Parameters:

threadIndex The thread on which this routine is being called, between 0 and threadCount - 1 for the threadCount passed to Start method.

srcBuffer Input area and source pixels.

dstBuffer Output area and destination pixels.

Referenced by Process().

6.20.3.4 void dng_filter_task::Start (uint32 *threadCount*, const dng_point & *tileSize*, dng_memory_allocator * *allocator*, dng_abort_sniffer * *sniffer*) [virtual]

Called prior to processing on specific threads. Can be used to allocate per-thread memory buffers, etc.

Parameters:

threadCount Total number of threads that will be used for processing. Less than or equal to MaxThreads of dng_area_task.

tileSize Size of source tiles which will be processed. (Not all tiles will be this size due to edge conditions.)

allocator dng_memory_allocator to use for allocating temporary buffers, etc.

sniffer Sniffer to test for user cancellation and to set up progress.

Reimplemented from dng_area_task.

References dng_memory_allocator::Allocate(), and SrcTileSize().

6.20.3.5 void dng_filter_task::Process (uint32 *threadIndex*, const dng_rect & *area*, dng_abort_sniffer * *sniffer*) [virtual]

Process one tile or partitioned area. Should not be overridden. Instead, override ProcessArea, which is where to implement filter processing for a specific type of dng_filter_task. There is no allocator parameter as all allocation should be done in Start.

Parameters:

threadIndex 0 to threadCount - 1 index indicating which thread this is. (Can be used to get a thread-specific buffer allocated in the Start method.)

area Size of tiles to be used for sizing buffers, etc. (Edges of processing can be smaller.)

sniffer dng_abort_sniffer to use to check for user cancellation and progress updates.

Implements dng_area_task.

References dng_image::edge_repeat, dng_pixel_buffer::fArea, dng_pixel_buffer::fData, dng_pixel_buffer::fPixelRange, dng_pixel_buffer::fPixelSize, dng_pixel_buffer::fPixelType, dng_pixel_buffer::fPlane, dng_pixel_buffer::fPlanes, dng_pixel_buffer::fPlaneStep, dng_pixel_buffer::fRowStep, dng_image::Get(), dng_image::PixelRange(), dng_image::PixelType(), ProcessArea(), dng_image::Put(), SrcArea(), and ThrowProgramError().

The documentation for this class was generated from the following files:

- dng_filter_task.h
- dng_filter_task.cpp

6.21 dng_fingerprint Class Reference

Container fingerprint (MD5 only at present).

`#include <dng_fingerprint.h>`

Public Member Functions

- bool IsNull () const

  Check if fingerprint is all zeros.

- bool IsValid () const

  Same as IsNull but expresses intention of testing validity.

- void Clear ()

  Set to all zeros, a value used to indicate an invalid fingerprint.

- bool operator== (const dng_fingerprint &print) const

  Test if two fingerprints are equal.

- bool operator!= (const dng_fingerprint &print) const

Test if two fingerprints are not equal.

- uint32 Collapse32 () const

  Produce a 32-bit hash value from fingerprint used for faster hashing of fingerprints.

Public Attributes

- uint8 **data** [16]

6.21.1 Detailed Description

Container fingerprint (MD5 only at present).

The documentation for this class was generated from the following files:

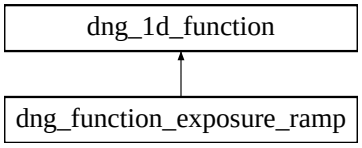
- dng_fingerprint.h
- dng_fingerprint.cpp

6.22 dng_function_exposure_ramp Class Reference

Curve for pre-exposure-compensation adjustment based on noise floor, shadows, and highlight level.

`#include <dng_render.h>`

Inheritance diagram for dng_function_exposure_ramp::



Public Member Functions

- **dng_function_exposure_ramp** (real64 white, real64 black, real64 minBlack)
- virtual real64 Evaluate (real64 x) const

Public Attributes

- real64 **fSlope**
- real64 **fBlack**

- real64 **fRadius**
- real64 **fQScale**

6.22.1 Detailed Description

Curve for pre-exposure-compensation adjustment based on noise floor, shadows, and highlight level.

6.22.2 Member Function Documentation

6.22.2.1 real64 dng_function_exposure_ramp::Evaluate (real64 *x*) const [virtual]

Return the mapping for value x. This method must be implemented by a derived class of dng_1d_function and the derived class determines the lookup method and function used.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

Mapped value for x

Implements dng_1d_function.

The documentation for this class was generated from the following files:

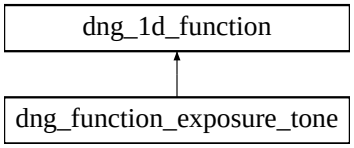
- dng_render.h
- dng_render.cpp

6.23 dng_function_exposure_tone Class Reference

Exposure compensation curve for a given compensation amount in stops using quadric for roll-off.

`#include <dng_render.h>`

Inheritance diagram for dng_function_exposure_tone::



Public Member Functions

- **dng_function_exposure_tone** (real64 exposure)
- virtual real64 Evaluate (real64 x) const

  Returns output value for a given input tone.

Protected Attributes

- bool **fIsNOP**
- real64 **fSlope**
- real64 **a**
- real64 **b**
- real64 **c**

6.23.1 Detailed Description

Exposure compensation curve for a given compensation amount in stops using quadric for roll-off.

The documentation for this class was generated from the following files:

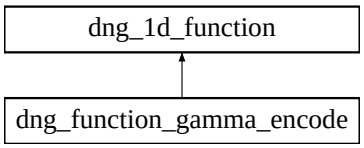
- dng_render.h
- dng_render.cpp

6.24 dng_function_gamma_encode Class Reference

Encoding gamma curve for a given color space.

`#include <dng_render.h>`

Inheritance diagram for dng_function_gamma_encode::



Public Member Functions

- **dng_function_gamma_encode** (const dng_color_space &space)
- virtual real64 Evaluate (real64 x) const

Protected Attributes

- const dng_color_space & **fSpace**

6.24.1 Detailed Description

Encoding gamma curve for a given color space.

6.24.2 Member Function Documentation

6.24.2.1 virtual real64 dng_function_gamma_encode::Evaluate (real64 *x*) const [virtual]

Return the mapping for value x. This method must be implemented by a derived class of dng_1d_function and the derived class determines the lookup method and function used.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

Mapped value for x

Implements dng_1d_function.

The documentation for this class was generated from the following file:

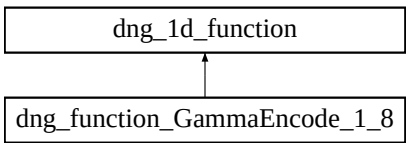
- dng_render.h

6.25 dng_function_GammaEncode_1_8 Class Reference

A dng_1d_function for gamma encoding with 1.8 gamma.

`#include <dng_color_space.h>`

Inheritance diagram for dng_function_GammaEncode_1_8::



Public Member Functions

- virtual real64 Evaluate (real64 x) const
- virtual real64 EvaluateInverse (real64 y) const

Static Public Member Functions

- static const dng_1d_function & **Get** ()

6.25.1 Detailed Description

A dng_1d_function for gamma encoding with 1.8 gamma.

6.25.2 Member Function Documentation

6.25.2.1 real64 dng_function_GammaEncode_1_8::Evaluate (real64 *x*) const [virtual]

Return the mapping for value x. This method must be implemented by a derived class of dng_1d_function and the derived class determines the lookup method and function used.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

Mapped value for x

Implements dng_1d_function.

6.25.2.2 real64 dng_function_GammaEncode_1_8::EvaluateInverse (real64 *y*) const [virtual]

Return the reverse mapped value for y. This method can be implemented by derived classes. The default implementation uses Newton's method to solve for x such that Evaluate(x) == y.

Parameters:

y A value to reverse map. Should be within the range of the function implemented by this dng_1d_function .

Return values:

A value x such that Evaluate(x) == y (to very close approximation).

Reimplemented from dng_1d_function.

References dng_1d_function::EvaluateInverse().

The documentation for this class was generated from the following files:

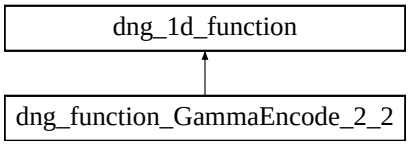
- dng_color_space.h
- dng_color_space.cpp

6.26 dng_function_GammaEncode_2_2 Class Reference

A dng_1d_function for gamma encoding with 2.2 gamma.

`#include <dng_color_space.h>`

Inheritance diagram for dng_function_GammaEncode_2_2::



Public Member Functions

- virtual real64 Evaluate (real64 x) const
- virtual real64 EvaluateInverse (real64 y) const

Static Public Member Functions

- static const dng_1d_function & **Get** ()

6.26.1 Detailed Description

A dng_1d_function for gamma encoding with 2.2 gamma.

6.26.2 Member Function Documentation

6.26.2.1 real64 dng_function_GammaEncode_2_2::Evaluate (real64 *x*) const `[virtual]`

Return the mapping for value x. This method must be implemented by a derived class of dng_1d_function and the derived class determines the lookup method and function used.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

Mapped value for x

Implements dng_1d_function.

6.26.2.2 real64 dng_function_GammaEncode_2_2::EvaluateInverse (real64 *y*) const [virtual]

Return the reverse mapped value for y. This method can be implemented by derived classes. The default implementation uses Newton's method to solve for x such that Evaluate(x) == y.

Parameters:

y A value to reverse map. Should be within the range of the function implemented by this dng_1d_function .

Return values:

A value x such that Evaluate(x) == y (to very close approximation).

Reimplemented from dng_1d_function.

References dng_1d_function::EvaluateInverse().

The documentation for this class was generated from the following files:

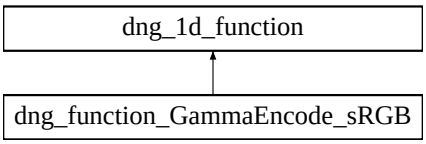
- dng_color_space.h
- dng_color_space.cpp

6.27 dng_function_GammaEncode_sRGB Class Reference

A dng_1d_function for gamma encoding in sRGB color space.

`#include <dng_color_space.h>`

Inheritance diagram for dng_function_GammaEncode_sRGB::



Public Member Functions

- virtual real64 Evaluate (real64 x) const
- virtual real64 EvaluateInverse (real64 y) const

Static Public Member Functions

- static const dng_1d_function & **Get** ()

6.27.1 Detailed Description

A dng_1d_function for gamma encoding in sRGB color space.

6.27.2 Member Function Documentation

6.27.2.1 real64 dng_function_GammaEncode_sRGB::Evaluate (real64 *x*) const [virtual]

Return the mapping for value x. This method must be implemented by a derived class of dng_1d_function and the derived class determines the lookup method and function used.

Parameters:

x A value between 0.0 and 1.0 (inclusive).

Return values:

Mapped value for x

Implements dng_1d_function.

6.27.2.2 real64 dng_function_GammaEncode_sRGB::EvaluateInverse (real64 *y*) const [virtual]

Return the reverse mapped value for y. This method can be implemented by derived classes. The default implementation uses Newton's method to solve for x such that Evaluate(x) == y.

Parameters:

y A value to reverse map. Should be within the range of the function implemented by this dng_1d_function .

Return values:

A value x such that Evaluate(x) == y (to very close approximation).

Reimplemented from dng_1d_function.

The documentation for this class was generated from the following files:

- dng_color_space.h
- dng_color_space.cpp

6.28 dng_host Class Reference

The main class for communication between the application and the DNG SDK. Used to customize memory allocation and other behaviors.

`#include <dng_host.h>`

Public Member Functions

- dng_host (dng_memory_allocator *allocator=NULL, dng_abort_sniffer *sniffer=NULL)
- virtual ~dng_host ()
- dng_memory_allocator & Allocator ()

  Getter for host's memory allocator.

- virtual dng_memory_block * Allocate (uint32 logicalSize)
- void SetSniffer (dng_abort_sniffer *sniffer)

  Setter for host's abort sniffer.

- dng_abort_sniffer * Sniffer ()

  Getter for host's abort sniffer.

- virtual void SniffForAbort ()
- void SetNeedsMeta (bool needs)
- bool NeedsMeta () const

  Getter for flag determining whether all XMP metadata should be parsed.

- void SetNeedsImage (bool needs)
- bool NeedsImage () const

  Setter for flag determining whether DNG image data is needed.

- void SetForPreview (bool preview)
- bool ForPreview () const
- void SetMinimumSize (uint32 size)
- uint32 MinimumSize () const

  Getter for the minimum preview size.

- void SetPreferredSize (uint32 size)
- uint32 PreferredSize () const

  Getter for the preferred preview size.

- void SetMaximumSize (uint32 size)
- uint32 MaximumSize () const

  Getter for the maximum preview size.

- void SetCropFactor (real64 cropFactor)
- real64 CropFactor () const

  Getter for the cropping factor.

- void ValidateSizes ()

  Makes sures minimum, preferred, and maximum sizes are reasonable.

- void SetKeepStage1 (bool keep)
- bool KeepStage1 () const
- void SetKeepStage2 (bool keep)
- bool KeepStage2 () const
- void SetKeepDNGPrivate (bool keep)
- bool KeepDNGPrivate ()

  Getter for flag determining whether all DNG private data will be kept.

- void SetKeepOriginalFile (bool keep)
- bool KeepOriginalFile ()

  Getter for flag determining whether to keep original RAW file data.

- virtual bool IsTransientError (dng_error_code code)
- virtual void PerformAreaTask (dng_area_task &task, const dng_rect &area)
- virtual dng_exif * Make_dng_exif ()
- virtual dng_shared * Make_dng_shared ()
- virtual dng_ifd * Make_dng_ifd ()
- virtual dng_negative * Make_dng_negative ()
- virtual dng_image * Make_dng_image (const dng_rect &bounds, uint32 planes, uint32 pixelType)

6.28.1 Detailed Description

The main class for communication between the application and the DNG SDK. Used to customize memory allocation and other behaviors.

dng_host allows setting parameters for the DNG conversion, mediates callback style interactions between the host application and the DNG SDK, and allows controlling certain internal behavior of the SDK such as memory allocation. Many applications will be able to use the default implementation of dng_host by just setting the dng_memory_allocator and dng_abort_sniffer in the constructor. More complex interactions will require deriving a class from dng_host.

Multiple dng_host objects can be allocated in a single process. This may be useful for DNG processing on separate threads. (Distinct dng_host objects are completely thread-safe for read/write. The application is responsible for establishing mutual exclusion for read/write access to a single dng_host object if it is used in multiple threads.)

6.28.2 Constructor & Destructor Documentation

6.28.2.1 dng_host::dng_host (dng_memory_allocator * *allocator* = NULL, dng_abort_sniffer * *sniffer* = NULL)

Allocate a dng_host object, possiblly with custom allocator and sniffer.

Parameters:

allocator Allows controlling all memory allocation done via this dng_host. Defaults to singleton global dng_memory_allocator, which calls new/delete dng_malloc_block for appropriate size.

sniffer Used to periodically check if pending DNG conversions should be aborted and to communicate progress updates. Defaults to singleton global dng_abort_sniffer, which never aborts and ignores progress updated.

6.28.2.2 dng_host::~dng_host () [virtual]

Clean up direct memory for dng_host. Memory allocator and abort sniffer are not deleted. Objects such as dng_image and others returned from host can still be used after host is deleted.

6.28.3 Member Function Documentation

6.28.3.1 dng_memory_block * dng_host::Allocate (uint32 *logicalSize*) [virtual]

Alocate a new dng_memory_block using the host's memory allocator. Uses the Allocator() property of host to allocate a new block of memory. Will call ThrowMemoryFull if block cannot be allocated.

Parameters:

logicalSize Number of usable bytes returned dng_memory_block must contain.

References dng_memory_allocator::Allocate(), and Allocator().

Referenced by dng_mosaic_info::InterpolateGeneric().

6.28.3.2 void dng_host::SniffForAbort () [virtual]

Check for pending abort. Should call ThrowUserCanceled if an abort is pending.

References Sniffer(), and dng_abort_sniffer::SniffForAbort().

Referenced by dng_mosaic_info::InterpolateGeneric().

6.28.3.3 void dng_host::SetNeedsMeta (bool *needs*) [inline]

Setter for flag determining whether all XMP metadata should be parsed. Defaults to true. One might not want metadata when doing a quick check to see if a file is readable.

Parameters:

needs If true, metadata is needed.

6.28.3.4 void dng_host::SetNeedsImage (bool *needs*) [inline]

Setter for flag determining whether DNG image data is needed. Defaults to true. Image data might not be needed for applications which only manipulate metadata.

Parameters:

needs If true, image data is needed.

6.28.3.5 void dng_host::SetForPreview (bool *preview*) [inline]

Setter for flag determining whether image should be preview quality, or full quality.

Parameters:

preview If true, rendered images are for preview.

6.28.3.6 bool dng_host::ForPreview () const [inline]

Getter for flag determining whether image should be preview quality. Preview quality images may be rendered more quickly. Current DNG SDK does not change rendering behavior based on this flag, but derived versions may use this getter to choose between a slower more accurate path and a faster "good enough for preview" one. Data produce with ForPreview set to true should not be written back to a DNG file, except as a preview image.

6.28.3.7 void dng_host::SetMinimumSize (uint32 *size*) [inline]

Setter for the minimum preview size.

Parameters:

size Minimum pixel size (long side of image).

Referenced by ValidateSizes().

6.28.3.8 void dng_host::SetPreferredSize (uint32 *size*) [inline]

Setter for the preferred preview size.

Parameters:

size Preferred pixel size (long side of image).

Referenced by ValidateSizes().

6.28.3.9 void dng_host::SetMaximumSize (uint32 *size*) [inline]

Setter for the maximum preview size.

Parameters:

size Maximum pixel size (long side of image).

6.28.3.10 void dng_host::SetCropFactor (real64 *cropFactor*) [inline]

Setter for the cropping factor.

Parameters:

cropFactor Fraction of image to be used after crop.

6.28.3.11 void dng_host::SetKeepStage1 (bool *keep*) [inline]

Setter for flag determining whether to keep stage 1, unprocessed raw data, after processing all stages.

Parameters:

keep If true, stage 1 data will be kept.

6.28.3.12 bool dng_host::KeepStage1 () const [inline]

Getter for flag determining whether to keep stage 1, unprocessed raw data, after processing all stages.

6.28.3.13 void dng_host::SetKeepStage2 (bool *keep*) [inline]

Setter for flag determining whether to keep stage 2, linearized, tone curve processed data, after processing all stages.

Parameters:

keep If true, stage 2 data will be kept.

6.28.3.14 bool dng_host::KeepStage2 () const [inline]

Getter for flag determining whether to keep stage 2, linearized, tone curve processed data, after processing all stages.

6.28.3.15 void dng_host::SetKeepDNGPrivate (bool *keep*) [inline]

Setter for flag determining whether DNG private data will be kept.

Parameters:

keep If true, DNG private data will be kept.

6.28.3.16 void dng_host::SetKeepOriginalFile (bool *keep*) [inline]

Setter for flag determining whether to keep original RAW file data.

Parameters:

keep If true, origianl RAW data will be kept.

6.28.3.17 bool dng_host::IsTransientError (dng_error_code *code*) [virtual]

Determine if an error is the result of a temporary, but planned-for occurence such as user cancellation or memory exhaustion. This method is sometimes used to determine whether to try and continue processing a DNG file despite errors in the file format, etc. In such cases, processing will be continued if IsTransientError returns false. This is so that user cancellation and memory exhaustion always terminate processing.

Parameters:

code Error to test for transience.

6.28.3.18 void dng_host::PerformAreaTask (dng_area_task & *task*, const dng_rect & *area*) [virtual]

General top-level botttleneck for image processing tasks. Default implementation calls dng_area_task::PerformAreaTask method on task. Can be overridden in derived classes to support multiprocessing, for example.

Parameters:

task Image processing task to perform on area.

area Rectangle over which to perform image processing task.

References Allocator(), dng_area_task::Perform(), and Sniffer().

Referenced by dng_mosaic_info::InterpolateFast(), dng_linearization_info::Linearize(), and dng_render::Render().

6.28.3.19 dng_exif * dng_host::Make_dng_exif () [virtual]

Factory method for dng_exif class. Can be used to customize allocation or to ensure a derived class is used instead of dng_exif.

References ThrowMemoryFull().

Referenced by dng_info::Parse().

6.28.3.20 dng_shared * dng_host::Make_dng_shared () [virtual]

Factory method for dng_shared class. Can be used to customize allocation or to ensure a derived class is used instead of dng_shared.

References ThrowMemoryFull().

Referenced by dng_info::Parse().

6.28.3.21 dng_ifd * dng_host::Make_dng_ifd () [virtual]

Factory method for dng_ifd class. Can be used to customize allocation or to ensure a derived class is used instead of dng_ifd.

References ThrowMemoryFull().

Referenced by dng_info::Parse().

6.28.3.22 dng_negative * dng_host::Make_dng_negative () [virtual]

Factory method for dng_negative class. Can be used to customize allocation or to ensure a derived class is used instead of dng_negative.

References Allocator().

6.28.3.23 dng_image * dng_host::Make_dng_image (const dng_rect & *bounds*, uint32 *planes*, uint32 *pixelType*) [virtual]

Factor method for dng_image class. Can be used to customize allocation or to ensure a derived class is used instead of dng_simple_image.

References Allocator(), and ThrowMemoryFull().

Referenced by dng_render::Render().

The documentation for this class was generated from the following files:

- dng_host.h
- dng_host.cpp

6.29 dng_ifd Class Reference

Container for a single image file directory of a digital negative.

`#include <dng_ifd.h>`

Public Types

- enum { **kMaxTileInfo** = 32 }

Public Member Functions

- virtual bool **ParseTag** (dng_stream &stream, uint32 parentCode, uint32 tagCode, uint32 tagType, uint32 tagCount, uint64 tagOffset)
- virtual void **PostParse** ()
- virtual bool **IsValidDNG** (dng_shared &shared, uint32 parentCode)
- dng_rect **Bounds** () const
- uint32 **TilesAcross** () const
- uint32 **TilesDown** () const
- uint32 **TilesPerImage** () const
- dng_rect **TileArea** (uint32 rowIndex, uint32 colIndex) const
- virtual uint32 **TileByteCount** (const dng_rect &tile) const
- void **SetSingleStrip** ()
- void **FindTileSize** (uint32 bytesPerTile=128 *1024, uint32 cellH=16, uint32 cellV=16)
- void **FindStripSize** (uint32 bytesPerStrip=128 *1024, uint32 cellV=16)
- virtual uint32 **PixelType** () const
- virtual bool **IsBaselineJPEG** () const
- virtual bool **CanRead** () const
- virtual void **ReadImage** (dng_host &host, dng_stream &stream, dng_image &image) const

Public Attributes

- bool **fUsesNewSubFileType**
- uint32 **fNewSubFileType**
- uint32 **fImageWidth**
- uint32 **fImageLength**
- uint32 **fBitsPerSample** [kMaxSamplesPerPixel]
- uint32 **fCompression**
- uint32 **fPredictor**
- uint32 **fPhotometricInterpretation**
- uint32 **fFillOrder**
- uint32 **fOrientation**
- uint32 **fOrientationType**
- uint64 **fOrientationOffset**
- bool **fOrientationBigEndian**
- uint32 **fSamplesPerPixel**
- uint32 **fPlanarConfiguration**
- real64 **fXResolution**
- real64 **fYResolution**
- uint32 **fResolutionUnit**
- bool **fUsesStrips**
- bool **fUsesTiles**
- uint32 **fTileWidth**
- uint32 **fTileLength**
- uint32 **fTileOffsetsType**
- uint32 **fTileOffsetsCount**
- uint64 **fTileOffsetsOffset**
- uint64 **fTileOffset** [kMaxTileInfo]
- uint32 **fTileByteCountsType**
- uint32 **fTileByteCountsCount**
- uint64 **fTileByteCountsOffset**
- uint32 **fTileByteCount** [kMaxTileInfo]
- uint32 **fSubIFDsCount**
- uint64 **fSubIFDsOffset**
- uint32 **fExtraSamplesCount**
- uint32 **fExtraSamples** [kMaxSamplesPerPixel]
- uint32 **fSampleFormat** [kMaxSamplesPerPixel]
- uint32 **fJPEGTablesCount**
- uint64 **fJPEGTablesOffset**
- uint64 **fJPEGInterchangeFormat**
- uint32 **fJPEGInterchangeFormatLength**
- real64 **fYCbCrCoefficientR**
- real64 **fYCbCrCoefficientG**

- real64 **fYCbCrCoefficientB**
- uint32 **fYCbCrSubSampleH**
- uint32 **fYCbCrSubSampleV**
- uint32 **fYCbCrPositioning**
- real64 **fReferenceBlackWhite** [6]
- uint32 **fCFARepeatPatternRows**
- uint32 **fCFARepeatPatternCols**
- uint8 **fCFAPattern** [kMaxCFAPattern][kMaxCFAPattern]
- uint8 **fCFAPlaneColor** [kMaxColorPlanes]
- uint32 **fCFALayout**
- uint32 **fLinearizationTableType**
- uint32 **fLinearizationTableCount**
- uint64 **fLinearizationTableOffset**
- uint32 **fBlackLevelRepeatRows**
- uint32 **fBlackLevelRepeatCols**
- real64 **fBlackLevel** [kMaxBlackPattern][kMaxBlackPattern][kMaxSamplesPerPixel]

- uint32 **fBlackLevelDeltaHType**
- uint32 **fBlackLevelDeltaHCount**
- uint64 **fBlackLevelDeltaHOffset**
- uint32 **fBlackLevelDeltaVType**
- uint32 **fBlackLevelDeltaVCount**
- uint64 **fBlackLevelDeltaVOffset**
- real64 **fWhiteLevel** [kMaxSamplesPerPixel]
- dng_urational **fDefaultScaleH**
- dng_urational **fDefaultScaleV**
- dng_urational **fBestQualityScale**
- dng_urational **fDefaultCropOriginH**
- dng_urational **fDefaultCropOriginV**
- dng_urational **fDefaultCropSizeH**
- dng_urational **fDefaultCropSizeV**
- uint32 **fBayerGreenSplit**
- dng_urational **fChromaBlurRadius**
- dng_urational **fAntiAliasStrength**
- dng_rect **fActiveArea**
- uint32 **fMaskedAreaCount**
- dng_rect **fMaskedArea** [kMaxMaskedAreas]
- uint32 **fRowInterleaveFactor**
- uint32 **fSubTileBlockRows**
- uint32 **fSubTileBlockCols**
- dng_preview_info **fPreviewInfo**
- bool **fLosslessJPEGBug16**
- uint32 **fSampleBitShift**
- uint64 **fThisIFD**
- uint64 **fNextIFD**

Protected Member Functions

- virtual bool **IsValidCFA** (dng_shared &shared, uint32 parentCode)

6.29.1 Detailed Description

Container for a single image file directory of a digital negative.

See DNG 1.1.0 specification for documentation of specific tags.

The documentation for this class was generated from the following files:

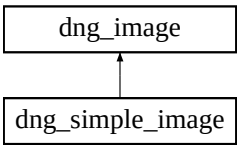
- dng_ifd.h
- dng_ifd.cpp

6.30 dng_image Class Reference

Base class for holding image data in DNG SDK. See dng_simple_image for derived class most often used in DNG SDK.

`#include <dng_image.h>`

Inheritance diagram for dng_image::



Public Types

- enum edge_option { edge_none, edge_zero, edge_repeat, edge_repeat_zero_last }

  How to handle requests to get image areas outside the image bounds.

Public Member Functions

- const dng_rect & Bounds () const

  Getter method for bounds of an image.

- dng_point Size () const

  Getter method for size of an image.

- uint32 Width () const

  Getter method for width of an image.

- uint32 Height () const

  Getter method for height of an image.

- uint32 Planes () const

  Getter method for number of planes in an image.

- uint32 PixelType () const
- virtual void SetPixelType (uint32 pixelType)
- uint32 PixelSize () const
- uint32 PixelRange () const
- virtual void SetPixelRange (uint32 pixelRange)
- virtual dng_rect RepeatingTile () const

  Getter for best "tile stride" for accessing image.

- void Get (dng_pixel_buffer &buffer, edge_option edgeOption=edge_none, uint32 repeatV=1, uint32 repeatH=1) const
- void Put (const dng_pixel_buffer &buffer)
- virtual void Trim (const dng_rect &r)
- virtual void Rotate (const dng_orientation &orientation)
- void CopyArea (const dng_image &src, const dng_rect &area, uint32 srcPlane, uint32 dstPlane, uint32 planes)
- void CopyArea (const dng_image &src, const dng_rect &area, uint32 plane, uint32 planes)
- bool EqualArea (const dng_image &rhs, const dng_rect &area, uint32 plane, uint32 planes) const
- void **SetConstant_uint8** (uint8 value)
- void **SetConstant_uint16** (uint16 value)
- void **SetConstant_int16** (int16 value)
- void **SetConstant_uint32** (uint32 value)
- void **SetConstant_real32** (real32 value)
- virtual void **GetRepeat** (dng_pixel_buffer &buffer, const dng_rect &srcArea, const dng_rect &dstArea) const

Protected Member Functions

- **dng_image** (const dng_rect &bounds, uint32 planes, uint32 pixelType, uint32 pixelRange)
- virtual void **AcquireTileBuffer** (dng_tile_buffer &buffer, const dng_rect &area, bool dirty) const

- virtual void **ReleaseTileBuffer** (dng_tile_buffer &buffer) const
- virtual void **DoGet** (dng_pixel_buffer &buffer) const
- virtual void **DoPut** (const dng_pixel_buffer &buffer)
- void **GetEdge** (dng_pixel_buffer &buffer, edge_option edgeOption, const dng_rect &srcArea, const dng_rect &dstArea) const
- virtual void **SetConstant** (uint32 value)

Protected Attributes

- dng_rect **fBounds**
- uint32 **fPlanes**
- uint32 **fPixelType**
- uint32 **fPixelRange**

Friends

- class dng_tile_buffer

6.30.1 Detailed Description

Base class for holding image data in DNG SDK. See dng_simple_image for derived class most often used in DNG SDK.

6.30.2 Member Enumeration Documentation

6.30.2.1 enum dng_image::edge_option

How to handle requests to get image areas outside the image bounds.

Enumerator:

edge_none Leave edge pixels unchanged.

edge_zero Pad with zeros.

edge_repeat Repeat edge pixels.

edge_repeat_zero_last Repeat edge pixels, except for last plane which is zero padded.

6.30.3 Member Function Documentation

6.30.3.1 uint32 dng_image::PixelType () const [inline]

Getter for pixel type.

Return values:

See dng_tagtypes.h . Valid values are ttByte, ttShort, ttSShort, ttLong, ttFloat .

Referenced by dng_mosaic_info::InterpolateGeneric(), PixelSize(), dng_filter_task::Process(), dng_render::Render(), and dng_image_writer::WriteTIFF().

6.30.3.2 void dng_image::SetPixelType (uint32 *pixelType*) [virtual]

Setter for pixel type.

Parameters:

pixelType The new pixel type .

Reimplemented in dng_simple_image.

References PixelSize(), and ThrowProgramError().

Referenced by dng_simple_image::SetPixelType().

6.30.3.3 uint32 dng_image::PixelSize () const

Getter for pixel size.

Return values:

Size,in bytes, of pixel type for this image .

References PixelType().

Referenced by dng_mosaic_info::InterpolateGeneric(), and SetPixelType().

6.30.3.4 uint32 dng_image::PixelRange () const

Getter for pixel range. For unsigned types, range is 0 to return value. For signed types, range is return value - 0x8000U. For ttFloat type, pixel range is 0.0 to 1.0 and this routine returns 1.

Referenced by dng_mosaic_info::InterpolateGeneric(), and dng_filter_task::Process().

6.30.3.5 void dng_image::SetPixelRange (uint32 *pixelRange*) [virtual]

Setter for pixel range.

Parameters:

pixelType The new pixel range .

Reimplemented in dng_simple_image.

Referenced by dng_simple_image::SetPixelRange().

6.30.3.6 void dng_image::Get (dng_pixel_buffer & *buffer*, edge_option *edgeOption* = `edge_none`, uint32 *repeatV* = `1`, uint32 *repeatH* = `1`) const

Get a pixel buffer of data on image with proper edge padding.

Parameters:

buffer Receives resulting pixel buffer.

edgeOption edge_option describing how to pad edges.

repeatV Amount of repeated padding needed in vertical for edge_repeat and edge_repeat_zero_last edgeOption cases.

repeatH Amount of repeated padding needed in horizontal for edge_repeat and edge_repeat_zero_last edgeOption cases.

References dng_pixel_buffer::DirtyPixel(), edge_none, dng_pixel_buffer::fArea, dng_pixel_buffer::fData, and dng_pixel_buffer::fPlane.

Referenced by dng_mosaic_info::InterpolateGeneric(), and dng_filter_task::Process().

6.30.3.7 void dng_image::Put (const dng_pixel_buffer & *buffer*)

Put a pixel buffer into image.

Parameters:

buffer Pixel buffer to copy from.

References dng_pixel_buffer::ConstPixel(), dng_pixel_buffer::fArea, dng_pixel_buffer::fData, dng_pixel_buffer::fPlane, dng_pixel_buffer::fPlanes, and Planes().

Referenced by dng_mosaic_info::InterpolateGeneric(), and dng_filter_task::Process().

6.30.3.8 void dng_image::Trim (const dng_rect & *r*) `[virtual]`

Shrink bounds of image to given rectangle.

Parameters:

r Rectangle to crop to.

Reimplemented in dng_simple_image.

References Bounds(), and ThrowProgramError().

6.30.3.9 void dng_image::Rotate (const dng_orientation & *orientation*) `[virtual]`

Rotate image to reflect given orientation change.

Parameters:

orientation Directive to rotate image in a certain way.

Reimplemented in dng_simple_image.

References ThrowProgramError().

6.30.3.10 void dng_image::CopyArea (const dng_image & *src*, const dng_rect & *area*, uint32 *srcPlane*, uint32 *dstPlane*, uint32 *planes*)

Copy image data from an area of one image to same area of another.

Parameters:

src Image to copy from.

area Rectangle of images to copy.

srcPlane Plane to start copying in src.

dstPlane Plane to start copying in this.

planes Number of planes to copy.

References dng_pixel_buffer::CopyArea().

Referenced by CopyArea().

6.30.3.11 void dng_image::CopyArea (const dng_image & *src*, const dng_rect & *area*, uint32 *plane*, uint32 *planes*) [inline]

Copy image data from an area of one image to same area of another.

Parameters:

src Image to copy from.

area Rectangle of images to copy.

plane Plane to start copying in src and this.

planes Number of planes to copy.

References CopyArea().

6.30.3.12 bool dng_image::EqualArea (const dng_image & *rhs*, const dng_rect & *area*, uint32 *plane*, uint32 *planes*) const

Return true if the contents of an area of the image are the same as those of another.

Parameters:

rhs Image to compare against.

area Rectangle of image to test.

plane Plane to start comparing.

planes Number of planes to compare.

References dng_pixel_buffer::EqualArea().

The documentation for this class was generated from the following files:

- dng_image.h
- dng_image.cpp

6.31 dng_image_writer Class Reference

Support for writing dng_image or dng_negative instances to a dng_stream in TIFF or DNG format.

`#include <dng_image_writer.h>`

Public Member Functions

- virtual void **WriteImage** (dng_host &host, const dng_ifd &ifd, dng_basic_tag_set &basic, dng_stream &stream, const dng_image &image, bool mapRange=false, uint32 fakeChannels=1)
- virtual void WriteTIFF (dng_host &host, dng_stream &stream, const dng_image &image, uint32 photometricInterpretation=piBlackIsZero, uint32 compression=ccUncompressed, const dng_negative *negative=NULL, const dng_color_space *space=NULL, const dng_resolution *resolution=NULL, const dng_jpeg_preview *thumbnail=NULL, const dng_memory_block *imageResources=NULL)
- virtual void WriteDNG (dng_host &host, dng_stream &stream, const dng_negative &negative, const dng_image_preview &thumbnail, uint32 compression=ccJPEG, const dng_preview_list *previewList=NULL)

Protected Types

- enum { **kImageBufferSize** = 128 * 1024 }

Protected Member Functions

- virtual uint32 **CompressedBufferSize** (const dng_ifd &ifd, uint32 uncompressedSize)
- virtual void **EncodePredictor** (dng_host &host, const dng_ifd &ifd, dng_pixel_buffer &buffer)

- virtual void **ByteSwapBuffer** (dng_host &host, dng_pixel_buffer &buffer)
- void **ReorderSubTileBlocks** (const dng_ifd &ifd, dng_pixel_buffer &buffer)
- virtual void **WriteData** (dng_host &host, const dng_ifd &ifd, dng_stream &stream, dng_pixel_buffer &buffer)
- virtual void **WriteTile** (dng_host &host, const dng_ifd &ifd, dng_stream &stream, const dng_image &image, const dng_rect &tileArea, bool mapRange, uint32 fakeChannels)

Protected Attributes

- AutoPtr< dng_memory_block > **fCompressedBuffer**
- AutoPtr< dng_memory_block > **fUncompressedBuffer**
- AutoPtr< dng_memory_block > **fSubTileBlockBuffer**

6.31.1 Detailed Description

Support for writing dng_image or dng_negative instances to a dng_stream in TIFF or DNG format.

6.31.2 Member Function Documentation

6.31.2.1 void dng_image_writer::WriteTIFF (dng_host & *host*, dng_stream & *stream*, const dng_image & *image*, uint32 *photometricInterpretation* = piBlackIsZero, uint32 *compression* = ccUncompressed, const dng_negative * *negative* = NULL, const dng_color_space * *space* = NULL, const dng_resolution * *resolution* = NULL, const dng_jpeg_preview * *thumbnail* = NULL, const dng_memory_block * *imageResources* = NULL) [virtual]

Write a dng_image to a dng_stream in TIFF format.

Parameters:

host Host interface used for progress updates, abort testing, buffer allocation, etc.

stream The dng_stream on which to write the TIFF.

image The actual image data to be written.

photometricInterpretation Either piBlackIsZero for monochrome or piRGB for RGB images.

compression Must be ccUncompressed .

negative If non-NULL, EXIF, IPTC, and XMP metadata from this negative is written to TIFF.

space If non-null and color space has an ICC profile, TIFF will be tagged with this profile. No color space conversion of image data occurs.

resolution If non-NULL, TIFF will be tagged with this resolution.

thumbnail If non-NULL, will be stored in TIFF as preview image.

imageResources If non-NULL, will image resources be stored in TIFF as well.

References dng_stream::BigEndian(), dng_image::Bounds(), dng_ifd::fBitsPerSample, dng_ifd::fCompression, dng_ifd::fExtraSamplesCount, dng_ifd::fImageLength, dng_ifd::fImageWidth, dng_ifd::FindStripSize(), dng_stream::Flush(), dng_ifd::fNewSubFileType, dng_ifd::fPhotometricInterpretation, dng_ifd::fPredictor, dng_ifd::fSampleFormat, dng_ifd::fSamplesPerPixel, AutoPtr< T >::Get(), dng_negative::GetExif(), dng_negative::GetXMP(), dng_negative::IPTCData(), dng_negative::IPTCDigest(), dng_negative::IPTCLength(), dng_negative::IsMakerNoteSafe(), dng_negative::MakerNoteData(), dng_negative::MakerNoteLength(), dng_image::PixelType(), dng_image::Planes(), dng_stream::Position(), dng_stream::Put_uint16(), dng_stream::Put_uint32(), AutoPtr< T >::Reset(), dng_stream::SetLength(), dng_ifd::SetSingleStrip(), dng_stream::SetWritePosition(), and dng_image::Size().

6.31.2.2 void dng_image_writer::WriteDNG (dng_host & *host*, dng_stream & *stream*, const dng_negative & *negative*, const dng_image_preview & *thumbnail*, uint32 *compression* = ccJPEG, const dng_preview_list * *previewList* = NULL) [virtual]

Write a dng_image to a dng_stream in DNG format.

Parameters:

host Host interface used for progress updates, abort testing, buffer allocation, etc.

stream The dng_stream on which to write the TIFF.

negative The image data and metadata (EXIF, IPTC, XMP) to be written.

thumbnail Thumbanil image. Must be provided.

compression Either ccUncompressed or ccJPEG for lossless JPEG.

previewCount The number of previews (not counting thumbnail).

preview Array of previewCount dng_preview pointers.

References dng_negative::AntiAliasStrength(), dng_negative::BaselineExposureR(), dng_negative::BaselineNoiseR(), dng_negative::BaselineSharpnessR(), dng_negative::BestQualityScale(), dng_stream::BigEndian(), dng_memory_data::Buffer_uint32(), dng_negative::CameraProfileToEmbed(), dng_negative::ChromaBlurRadius(), dng_negative::ColorimetricReference(), dng_fingerprint::data, dng_negative::DefaultCropOriginH(), dng_negative::DefaultCropOriginV(), dng_negative::DefaultCropSizeH(), dng_negative::DefaultCropSizeV(), dng_negative::DefaultScaleH(), dng_negative::DefaultScaleV(), dng_ifd::fBitsPerSample, dng_mosaic_info::fCFAPatternSize, dng_ifd::fCompression, dng_ifd::fImageLength,

dng_ifd::fImageWidth, dng_negative::FindOriginalRawFileDigest(), dng_negative::FindRawDataUniqueID(), dng_negative::FindRawImageDigest(), dng_ifd::FindTileSize(), dng_linearization_info::fLinearizationTable, dng_stream::Flush(), dng_ifd::fPhotometricInterpretation, dng_ifd::fRowInterleaveFactor, dng_ifd::fSamplesPerPixel, dng_ifd::fSubTileBlockCols, dng_ifd::fSubTileBlockRows, AutoPtr< T >::Get(), dng_negative::GetExif(), dng_negative::GetLinearizationInfo(), dng_negative::GetMosaicInfo(), dng_negative::GetXMP(), dng_negative::HasOriginalRawFileName(), dng_mosaic_info::IsColorFilterArray(), dng_negative::IsMakerNoteSafe(), dng_negative::IsMonochrome(), dng_fingerprint::IsValid(), kMaxDNGPreviews, dng_negative::LocalName(), dng_negative::MakerNoteData(), dng_negative::MakerNoteLength(), dng_negative::ModelName(), dng_negative::NoiseReductionApplied(), dng_negative::Orientation(), dng_negative::OriginalRawFileData(), dng_negative::OriginalRawFileDataLength(), dng_negative::OriginalRawFileDigest(), dng_negative::OriginalRawFileName(), dng_stream::Position(), dng_negative::PrivateData(), dng_negative::PrivateLength(), dng_negative::ProfileByIndex(), dng_negative::ProfileCount(), dng_stream::Put_uint16(), dng_stream::Put_uint32(), dng_negative::RawDataUniqueID(), dng_negative::RawImage(), dng_negative::RawImageDigest(), AutoPtr< T >::Reset(), dng_stream::SetLength(), dng_ifd::SetSingleStrip(), dng_stream::SetWritePosition(), dng_negative::ShadowScaleR(), ThrowProgramError(), and dng_camera_profile::WasReadFromDNG().

The documentation for this class was generated from the following files:

- dng_image_writer.h
- dng_image_writer.cpp

6.32 dng_info Class Reference

Top-level structure of DNG file with access to metadata.

`#include <dng_info.h>`

Public Member Functions

- virtual void Parse (dng_host &host, dng_stream &stream)
- virtual void PostParse (dng_host &host)

  Must be called immediately after a successful Parse operation.

- virtual bool IsValidDNG ()

Public Attributes

- uint64 **fTIFFBlockOffset**

- uint64 **fTIFFBlockOriginalOffset**
- bool **fBigEndian**
- uint32 **fMagic**
- AutoPtr< dng_exif > **fExif**
- AutoPtr< dng_shared > **fShared**
- int32 **fMainIndex**
- uint32 **fIFDCount**
- AutoPtr< dng_ifd > **fIFD** [kMaxSubIFDs+1]
- uint32 **fChainedIFDCount**
- AutoPtr< dng_ifd > **fChainedIFD** [kMaxChainedIFDs]

Protected Member Functions

- virtual void **ValidateMagic** ()
- virtual void **ParseTag** (dng_host &host, dng_stream &stream, dng_exif *exif, dng_shared *shared, dng_ifd *ifd, uint32 parentCode, uint32 tagCode, uint32 tagType, uint32 tagCount, uint64 tagOffset, int64 offsetDelta)
- virtual bool **ValidateIFD** (dng_stream &stream, uint64 ifdOffset, int64 offsetDelta)
- virtual void **ParseIFD** (dng_host &host, dng_stream &stream, dng_exif *exif, dng_shared *shared, dng_ifd *ifd, uint64 ifdOffset, int64 offsetDelta, uint32 parentCode)
- virtual bool **ParseMakerNoteIFD** (dng_host &host, dng_stream &stream, uint64 ifdSize, uint64 ifdOffset, int64 offsetDelta, uint64 minOffset, uint64 maxOffset, uint32 parentCode)
- virtual void **ParseMakerNote** (dng_host &host, dng_stream &stream, uint32 makerNoteCount, uint64 makerNoteOffset, int64 offsetDelta, uint64 minOffset, uint64 maxOffset)
- virtual void **ParseSonyPrivateData** (dng_host &host, dng_stream &stream, uint32 count, uint64 oldOffset, uint64 newOffset)
- virtual void **ParseDNGPrivateData** (dng_host &host, dng_stream &stream)

Protected Attributes

- uint32 **fMakerNoteNextIFD**

6.32.1 Detailed Description

Top-level structure of DNG file with access to metadata.

See DNG 1.1.0 specification for information on member fields of this class.

6.32.2 Member Function Documentation

6.32.2.1 void dng_info::Parse (dng_host & *host*, dng_stream & *stream*) [virtual]

Read dng_info from a dng_stream

Parameters:

host DNG host used for progress updating, abort testing, buffer allocation, etc.

stream Stream to read DNG data from.

References AutoPtr< T >::Get(), dng_stream::Get_uint16(), dng_stream::Get_uint32(), dng_camera_profile::IsValid(), kMaxChainedIFDs, kMaxSubIFDs, dng_stream::Length(), dng_host::Make_dng_exif(), dng_host::Make_dng_ifd(), dng_host::Make_dng_shared(), dng_camera_profile::Parse(), dng_stream::Position(), dng_stream::PositionInOriginalFile(), AutoPtr< T >::Reset(), dng_stream::SetBigEndian(), dng_stream::SetLittleEndian(), dng_stream::SetReadPosition(), and ThrowBadFormat().

6.32.2.2 bool dng_info::IsValidDNG () [virtual]

Test validity of DNG data.

Return values:

true if stream provided a valid DNG.

References AutoPtr< T >::Get().

The documentation for this class was generated from the following files:

- dng_info.h
- dng_info.cpp

6.33 dng_iptc Class Reference

Class for reading and holding IPTC metadata associated with a DNG file.

`#include <dng_iptc.h>`

Public Member Functions

- bool IsEmpty () const
- bool NotEmpty () const
- void Parse (const void *blockData, uint32 blockSize, uint64 offsetInOriginalFile)
- dng_memory_block * Spool (dng_memory_allocator &allocator)

Public Attributes

- dng_string **fTitle**
- int32 **fUrgency**
- dng_string **fCategory**
- dng_string_list **fSupplementalCategories**
- dng_string_list **fKeywords**
- dng_string **fInstructions**
- dng_date_time_info **fDateTimeCreated**
- dng_string **fAuthor**
- dng_string **fAuthorsPosition**
- dng_string **fCity**
- dng_string **fState**
- dng_string **fCountry**
- dng_string **fCountryCode**
- dng_string **fLocation**
- dng_string **fTransmissionReference**
- dng_string **fHeadline**
- dng_string **fCredit**
- dng_string **fSource**
- dng_string **fCopyrightNotice**
- dng_string **fDescription**
- dng_string **fDescriptionWriter**

Protected Types

- enum **DataSet** {

  kRecordVersionSet = 0, **kObjectNameSet** = 5, **kUrgencySet** = 10, **kCategorySet** = 15,

  kSupplementalCategoriesSet = 20, **kKeywordsSet** = 25, **kSpecialInstructionsSet** = 40, **kDateCreatedSet** = 55,

  kTimeCreatedSet = 60, **kBylineSet** = 80, **kBylineTitleSet** = 85, **kCitySet** = 90,

  kSublocationSet = 92, **kProvinceStateSet** = 95, **kCountryCodeSet** = 100, **kCountryNameSet** = 101,

  kOriginalTransmissionReferenceSet = 103, **kHeadlineSet** = 105, **kCreditSet** = 110, **kSourceSet** = 115,

  kCopyrightNoticeSet = 116, **kCaptionSet** = 120, **kCaptionWriterSet** = 122 }
- enum **CharSet** { **kCharSetUnknown** = 0, **kCharSetUTF8** = 1 }

Protected Member Functions

- void **ParseString** (dng_stream &stream, dng_string &s, CharSet charSet)
- void **SpoolString** (dng_stream &stream, const dng_string &s, uint8 dataSet, uint32 maxChars, CharSet charSet)
- bool **SafeForSystemEncoding** () const

Static Protected Member Functions

- static bool **SafeForSystemEncoding** (const dng_string &s)
- static bool **SafeForSystemEncoding** (const dng_string_list &list)

6.33.1 Detailed Description

Class for reading and holding IPTC metadata associated with a DNG file.

See the IPTC specification for information on member fields of this class.

6.33.2 Member Function Documentation

6.33.2.1 bool dng_iptc::IsEmpty () const

Test if IPTC metadata exists.

Return values:

true if no IPTC metadata exists for this DNG.

References dng_date_time_info::IsValid(), and NotEmpty().

Referenced by NotEmpty().

6.33.2.2 bool dng_iptc::NotEmpty () const [inline]

Test if IPTC metadata exists.

Return values:

true if IPTC metadata exists for this DNG.

References IsEmpty().

Referenced by IsEmpty().

6.33.2.3 void dng_iptc::Parse (const void * *blockData*, uint32 *blockSize*, uint64 *offsetInOriginalFile*)

Parse a complete block of IPTC data.

Parameters:

blockData The block of IPTC data.

blockSize Size in bytes of data block.

offsetInOriginalFile Used to enable certain file patching operations such as updating date/time in place.

References dng_date_time_info::Decode_IPTC_Date(), dng_date_time_info::Decode_IPTC_Time(), dng_stream::Get(), dng_stream::Get_int8(), dng_stream::Get_uint16(), dng_stream::Get_uint8(), dng_stream::Length(), dng_stream::Position(), dng_stream::SetBigEndian(), and dng_stream::SetReadPosition().

6.33.2.4 dng_memory_block * dng_iptc::Spool (dng_memory_allocator & *allocator*)

Serialize IPTC data to a memory block.

Parameters:

allocator Memory allocator used to acquire memory block.

Return values:

Memory block

References dng_stream::AsMemoryBlock(), DNG_ASSERT, dng_date_time_info::Encode_IPTC_Date(), dng_date_time_info::Encode_IPTC_Time(), dng_stream::Flush(), dng_date_time_info::IsValid(), dng_stream::Put(), dng_stream::Put_uint16(), dng_stream::Put_uint8(), and dng_stream::SetBigEndian().

The documentation for this class was generated from the following files:

- dng_iptc.h
- dng_iptc.cpp

6.34 dng_linearization_info Class Reference

Class for managing data values related to DNG linearization.

`#include <dng_linearization_info.h>`

Public Member Functions

- void **RoundBlacks** ()
- virtual void **Parse** (dng_host &host, dng_stream &stream, dng_info &info)
- virtual void **PostParse** (dng_host &host, dng_negative &negative)
- real64 MaxBlackLevel (uint32 plane) const

  Compute the maximum black level for a given sample plane taking into account base black level, repeated black level patter, and row/column delta maps.

- virtual void Linearize (dng_host &host, const dng_image &srcImage, dng_image &dstImage)
- dng_urational BlackLevel (uint32 row, uint32 col, uint32 plane) const
- uint32 RowBlackCount () const

  Number of per-row black level deltas in fBlackDeltaV.

- dng_srational RowBlack (uint32 row) const
- uint32 ColumnBlackCount () const

  Number of per-column black level deltas in fBlackDeltaV.

- dng_srational ColumnBlack (uint32 col) const

Public Attributes

- dng_rect fActiveArea
- uint32 fMaskedAreaCount

  Number of rectangles in fMaskedArea.

- dng_rect fMaskedArea [kMaxMaskedAreas]
- AutoPtr< dng_memory_block > fLinearizationTable
- uint32 fBlackLevelRepeatRows

  Actual number of rows in fBlackLevel pattern.

- uint32 fBlackLevelRepeatCols

  Actual number of columns in fBlackLevel pattern.

- real64 fBlackLevel [kMaxBlackPattern][kMaxBlackPattern][kMaxSamplesPerPixel]

  Repeating pattern of black level deltas fBlackLevelRepeatRows by fBlackLevelRepeatCols in size.

- AutoPtr< dng_memory_block > fBlackDeltaH

  Memory block of double-precision floating point deltas between baseline black level and a given column's black level.

- AutoPtr< dng_memory_block > fBlackDeltaV

  Memory block of double-precision floating point deltas between baseline black level and a given row's black level.

- real64 fWhiteLevel [kMaxSamplesPerPixel]

  Single white level (maximum sensor value) for each sample plane.

Protected Attributes

- int32 **fBlackDenom**

6.34.1 Detailed Description

Class for managing data values related to DNG linearization.

See LinearizationTable, BlackLevel, BlackLevelRepeatDim, BlackLevelDeltaH, BlackLevelDeltaV and WhiteLevel tags in the DNG 1.1.0 specification.

6.34.2 Member Function Documentation

6.34.2.1 void dng_linearization_info::Linearize (dng_host & *host*, const dng_image & *srcImage*, dng_image & *dstImage*) [virtual]

Convert raw data from in-file format to a true linear image using linearization data from DNG.

Parameters:

host Used to allocate buffers, check for aborts, and post progress updates.

srcImage Input pre-linearization RAW samples.

dstImage Output linearized image.

References fActiveArea, and dng_host::PerformAreaTask().

6.34.2.2 dng_urational dng_linearization_info::BlackLevel (uint32 *row*, uint32 *col*, uint32 *plane*) const

Compute black level for one coordinate and sample plane in the image.

Parameters:

row Row to compute black level for.

col Column to compute black level for.

plane Sample plane to compute black level for.

References fBlackLevel.

6.34.2.3 dng_srational dng_linearization_info::RowBlack (uint32 *row*) const

Lookup black level delta for a given row.

Parameters:

row Row to get black level for.

Return values:

black level for indicated row.

References fBlackDeltaV, and AutoPtr< T >::Get().

6.34.2.4 dng_srational dng_linearization_info::ColumnBlack (uint32 *col*) const

Lookup black level delta for a given column.

Parameters:

col Column to get black level for.

Return values:

black level for indicated column.

References fBlackDeltaH, and AutoPtr< T >::Get().

6.34.3 Member Data Documentation

6.34.3.1 dng_rect dng_linearization_info::fActiveArea

This rectangle defines the active (non-masked) pixels of the sensor. The order of the rectangle coordinates is: top, left, bottom, right.

Referenced by Linearize().

6.34.3.2 dng_rect dng_linearization_info::fMaskedArea[kMaxMaskedAreas]

List of non-overlapping rectangle coordinates of fully masked pixels. Can be optionally used by DNG readers to measure the black encoding level. The order of each

rectangle's coordinates is: top, left, bottom, right. If the raw image data has already had its black encoding level subtracted, then this tag should not be used, since the masked pixels are no longer useful. Note that DNG writers are still required to include an estimate and store the black encoding level using the black level DNG tags. Support for the MaskedAreas tag is not required of DNG readers.

6.34.3.3 AutoPtr<dng_memory_block> dng_linearization_info::fLinearizationTable

A lookup table that maps stored values into linear values. This tag is typically used to increase compression ratios by storing the raw data in a non-linear, more visually uniform space with fewer total encoding levels. If SamplesPerPixel is not equal to one, e.g. Fuji S3 type sensor, this single table applies to all the samples for each pixel.

Referenced by dng_image_writer::WriteDNG().

The documentation for this class was generated from the following files:

- dng_linearization_info.h
- dng_linearization_info.cpp

6.35 dng_memory_allocator Class Reference

Interface for dng_memory_block allocator.

`#include <dng_memory.h>`

Public Member Functions

- virtual dng_memory_block * Allocate (uint32 size)

6.35.1 Detailed Description

Interface for dng_memory_block allocator.

6.35.2 Member Function Documentation

6.35.2.1 dng_memory_block * dng_memory_allocator::Allocate (uint32 *size*) [virtual]

Allocate a dng_memory block.

Parameters:

size Number of bytes in memory block.

Return values:

A dng_memory_block with at least size bytes of valid storage.

Exceptions:

dng_exception with fErrorCode equal to dng_error_memory.

References ThrowMemoryFull().

Referenced by dng_host::Allocate(), dng_stream::AsMemoryBlock(), dng_1d_table::Initialize(), and dng_filter_task::Start().

The documentation for this class was generated from the following files:

- dng_memory.h
- dng_memory.cpp

6.36 dng_memory_block Class Reference

Class to provide resource acquisition is instantiation discipline for image buffers and other larger memory allocations.

`#include <dng_memory.h>`

Inherited by dng_malloc_block.

Public Member Functions

- uint32 LogicalSize () const
- void * Buffer ()
- const void * Buffer () const
- char * Buffer_char ()
- const char * Buffer_char () const
- uint8 * Buffer_uint8 ()
- const uint8 * Buffer_uint8 () const
- uint16 * Buffer_uint16 ()
- const uint16 * Buffer_uint16 () const
- int16 * Buffer_int16 ()
- const int16 * Buffer_int16 () const
- uint32 * Buffer_uint32 ()
- const uint32 * Buffer_uint32 () const
- int32 * Buffer_int32 ()
- const int32 * Buffer_int32 () const
- real32 * Buffer_real32 ()
- const real32 * Buffer_real32 () const
- real64 * Buffer_real64 ()
- const real64 * Buffer_real64 () const

Protected Member Functions

- **dng_memory_block** (uint32 logicalSize)
- uint32 **PhysicalSize** ()
- void **SetBuffer** (void *p)

6.36.1 Detailed Description

Class to provide resource acquisition is instantiation discipline for image buffers and other larger memory allocations.

This class requires a dng_memory_allocator for allocation.

6.36.2 Member Function Documentation

6.36.2.1 uint32 dng_memory_block::LogicalSize () const [inline]

Getter for available size, in bytes, of memory block.

Return values:

size in bytes of available memory in memory block.

6.36.2.2 void* dng_memory_block::Buffer () [inline]

Return pointer to allocated memory as a void *..

Return values:

void * valid for as many bytes as were allocated.

Referenced by Buffer_char(), Buffer_int16(), Buffer_int32(), Buffer_real32(), Buffer_real64(), Buffer_uint16(), Buffer_uint32(), and Buffer_uint8().

6.36.2.3 const void* dng_memory_block::Buffer () const [inline]

Return pointer to allocated memory as a const void *.

Return values:

const void * valid for as many bytes as were allocated.

6.36.2.4 char* dng_memory_block::Buffer_char () [inline]

Return pointer to allocated memory as a char *.

Return values:

char * valid for as many bytes as were allocated.

References Buffer().

6.36.2.5 const char* dng_memory_block::Buffer_char () const [inline]

Return pointer to allocated memory as a const char *.

Return values:

const char * valid for as many bytes as were allocated.

References Buffer().

6.36.2.6 uint8* dng_memory_block::Buffer_uint8 () [inline]

Return pointer to allocated memory as a uint8 *.

Return values:

uint8 * valid for as many bytes as were allocated.

References Buffer().

Referenced by dng_memory_stream::CopyToStream().

6.36.2.7 const uint8* dng_memory_block::Buffer_uint8 () const [inline]

Return pointer to allocated memory as a const uint8 *.

Return values:

const uint8 * valid for as many bytes as were allocated.

References Buffer().

6.36.2.8 uint16* dng_memory_block::Buffer_uint16 () [inline]

Return pointer to allocated memory as a uint16 *.

Return values:

uint16 * valid for as many bytes as were allocated.

References Buffer().

6.36.2.9 const uint16* dng_memory_block::Buffer_uint16 () const [inline]

Return pointer to allocated memory as a const uint16 *.

Return values:

const uint16 * valid for as many bytes as were allocated.

References Buffer().

6.36.2.10 int16* dng_memory_block::Buffer_int16 () [inline]

Return pointer to allocated memory as a int16 *.

Return values:

int16 * valid for as many bytes as were allocated.

References Buffer().

6.36.2.11 const int16* dng_memory_block::Buffer_int16 () const [inline]

Return pointer to allocated memory as a const int16 *.

Return values:

const int16 * valid for as many bytes as were allocated.

References Buffer().

6.36.2.12 uint32* dng_memory_block::Buffer_uint32 () [inline]

Return pointer to allocated memory as a uint32 *.

Return values:

uint32 * valid for as many bytes as were allocated.

References Buffer().

6.36.2.13 const uint32* dng_memory_block::Buffer_uint32 () const [inline]

Return pointer to allocated memory as a const uint32 *.

Return values:

const uint32 * valid for as many bytes as were allocated.

References Buffer().

6.36.2.14 int32* dng_memory_block::Buffer_int32 () [inline]

Return pointer to allocated memory as a int32 *.

Return values:

int32 * valid for as many bytes as were allocated.

References Buffer().

6.36.2.15 const int32* dng_memory_block::Buffer_int32 () const [inline]

Return pointer to allocated memory as a const int32 *.

Return values:

const int32 * valid for as many bytes as were allocated.

References Buffer().

6.36.2.16 real32* dng_memory_block::Buffer_real32 () [inline]

Return pointer to allocated memory as a real32 *.

Return values:

real32 * valid for as many bytes as were allocated.

References Buffer().

6.36.2.17 const real32* dng_memory_block::Buffer_real32 () const [inline]

Return pointer to allocated memory as a const real32 *.

Return values:

const real32 * valid for as many bytes as were allocated.

References Buffer().

6.36.2.18 real64* dng_memory_block::Buffer_real64 () [inline]

Return pointer to allocated memory as a real64 *.

Return values:

real64 * valid for as many bytes as were allocated.

References Buffer().

6.36.2.19 const real64* dng_memory_block::Buffer_real64 () const [inline]

Return pointer to allocated memory as a const real64 *.

Return values:

const real64 * valid for as many bytes as were allocated.

References Buffer().

The documentation for this class was generated from the following file:

- dng_memory.h

6.37 dng_memory_data Class Reference

Class to provide resource acquisition is instantiation discipline for small memory allocations.

`#include <dng_memory.h>`

Public Member Functions

- dng_memory_data ()
- dng_memory_data (uint32 size)
- ~dng_memory_data ()

  Release memory buffer using free.

- void Allocate (uint32 size)
- void Clear ()
- void * Buffer ()
- const void * Buffer () const
- char * Buffer_char ()
- const char * Buffer_char () const
- uint8 * Buffer_uint8 ()

- const uint8 * Buffer_uint8 () const
- uint16 * Buffer_uint16 ()
- const uint16 * Buffer_uint16 () const
- int16 * Buffer_int16 ()
- const int16 * Buffer_int16 () const
- uint32 * Buffer_uint32 ()
- const uint32 * Buffer_uint32 () const
- int32 * Buffer_int32 ()
- const int32 * Buffer_int32 () const
- uint64 * Buffer_uint64 ()
- const uint64 * Buffer_uint64 () const
- int64 * Buffer_int64 ()
- const int64 * Buffer_int64 () const
- real32 * Buffer_real32 ()
- const real32 * Buffer_real32 () const
- real64 * Buffer_real64 ()
- const real64 * Buffer_real64 () const

6.37.1 Detailed Description

Class to provide resource acquisition is instantiation discipline for small memory allocations.

Support for memory allocation. This class does not use dng_memory_allocator for memory allocation.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 dng_memory_data::dng_memory_data ()

Construct an empty memory buffer using malloc.

Exceptions:

dng_memory_full with fErrorCode equal to dng_error_memory.

6.37.2.2 dng_memory_data::dng_memory_data (uint32 *size*)

Construct memory buffer of size bytes using malloc.

Parameters:

size Number of bytes of memory needed.

Exceptions:

dng_memory_full with fErrorCode equal to dng_error_memory.

References Allocate().

6.37.3 Member Function Documentation

6.37.3.1 void dng_memory_data::Allocate (uint32 *size*)

Clear existing memory buffer and allocate new memory of size bytes.

Parameters:

size Number of bytes of memory needed.

Exceptions:

dng_memory_full with fErrorCode equal to dng_error_memory.

References Clear(), and ThrowMemoryFull().

Referenced by dng_memory_data().

6.37.3.2 void dng_memory_data::Clear ()

Release any allocated memory using free. Object is still valid and Allocate can be called again.

Referenced by Allocate(), and ~dng_memory_data().

6.37.3.3 void* dng_memory_data::Buffer () [inline]

Return pointer to allocated memory as a void *..

Return values:

void * valid for as many bytes as were allocated.

Referenced by Buffer_char(), Buffer_int16(), Buffer_int32(), Buffer_int64(), Buffer_real32(), Buffer_real64(), Buffer_uint16(), Buffer_uint32(), Buffer_uint64(), Buffer_uint8(), dng_stream::CopyToStream(), and dng_stream::PutZeros().

6.37.3.4 const void* dng_memory_data::Buffer () const [inline]

Return pointer to allocated memory as a const void *.

Return values:

const void * valid for as many bytes as were allocated.

6.37.3.5 char* dng_memory_data::Buffer_char () [inline]

Return pointer to allocated memory as a char *.

Return values:

char * valid for as many bytes as were allocated.

References Buffer().

6.37.3.6 const char* dng_memory_data::Buffer_char () const [inline]

Return pointer to allocated memory as a const char *.

Return values:

const char * valid for as many bytes as were allocated.

References Buffer().

6.37.3.7 uint8* dng_memory_data::Buffer_uint8 () [inline]

Return pointer to allocated memory as a uint8 *.

Return values:

uint8 * valid for as many bytes as were allocated.

References Buffer().

6.37.3.8 const uint8* dng_memory_data::Buffer_uint8 () const [inline]

Return pointer to allocated memory as a const uint8 *.

Return values:

const uint8 * valid for as many bytes as were allocated.

References Buffer().

6.37.3.9 uint16* dng_memory_data::Buffer_uint16 () [inline]

Return pointer to allocated memory as a uint16 *.

Return values:

uint16 * valid for as many bytes as were allocated.

References Buffer().

6.37.3.10 const uint16* dng_memory_data::Buffer_uint16 () const [inline]

Return pointer to allocated memory as a const uint16 *.

Return values:

const uint16 * valid for as many bytes as were allocated.

References Buffer().

6.37.3.11 int16* dng_memory_data::Buffer_int16 () [inline]

Return pointer to allocated memory as a int16 *.

Return values:

int16 * valid for as many bytes as were allocated.

References Buffer().

6.37.3.12 const int16* dng_memory_data::Buffer_int16 () const [inline]

Return pointer to allocated memory as a const int16 *.

Return values:

const int16 * valid for as many bytes as were allocated.

References Buffer().

6.37.3.13 uint32* dng_memory_data::Buffer_uint32 () [inline]

Return pointer to allocated memory as a uint32 *.

Return values:

uint32 * valid for as many bytes as were allocated.

References Buffer().

Referenced by dng_image_writer::WriteDNG().

6.37.3.14 const uint32* dng_memory_data::Buffer_uint32 () const [inline]

Return pointer to allocated memory as a uint32 *.

Return values:

uint32 ∗ valid for as many bytes as were allocated.

References Buffer().

6.37.3.15 int32∗ dng_memory_data::Buffer_int32 () [inline]

Return pointer to allocated memory as a const int32 ∗.

Return values:

const int32 ∗ valid for as many bytes as were allocated.

References Buffer().

6.37.3.16 const int32∗ dng_memory_data::Buffer_int32 () const [inline]

Return pointer to allocated memory as a const int32 ∗.

Return values:

const int32 ∗ valid for as many bytes as were allocated.

References Buffer().

6.37.3.17 uint64∗ dng_memory_data::Buffer_uint64 () [inline]

Return pointer to allocated memory as a uint64 ∗.

Return values:

uint64 ∗ valid for as many bytes as were allocated.

References Buffer().

6.37.3.18 const uint64∗ dng_memory_data::Buffer_uint64 () const [inline]

Return pointer to allocated memory as a uint64 ∗.

Return values:

uint64 ∗ valid for as many bytes as were allocated.

References Buffer().

6.37.3.19 int64* dng_memory_data::Buffer_int64 () [inline]

Return pointer to allocated memory as a const int64 *.

Return values:

const int64 * valid for as many bytes as were allocated.

References Buffer().

6.37.3.20 const int64* dng_memory_data::Buffer_int64 () const [inline]

Return pointer to allocated memory as a const int64 *.

Return values:

const int64 * valid for as many bytes as were allocated.

References Buffer().

6.37.3.21 real32* dng_memory_data::Buffer_real32 () [inline]

Return pointer to allocated memory as a real32 *.

Return values:

real32 * valid for as many bytes as were allocated.

References Buffer().

6.37.3.22 const real32* dng_memory_data::Buffer_real32 () const [inline]

Return pointer to allocated memory as a const real32 *.

Return values:

const real32 * valid for as many bytes as were allocated.

References Buffer().

6.37.3.23 real64* dng_memory_data::Buffer_real64 () [inline]

Return pointer to allocated memory as a real64 *.

Return values:

real64 * valid for as many bytes as were allocated.

References Buffer().

6.37.3.24 const real64* dng_memory_data::Buffer_real64 () const [inline]

Return pointer to allocated memory as a const real64 *.

Return values:

const real64 * valid for as many bytes as were allocated.

References Buffer().

The documentation for this class was generated from the following files:

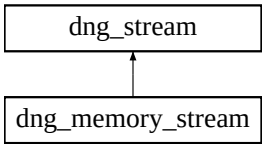
- dng_memory.h
- dng_memory.cpp

6.38 dng_memory_stream Class Reference

A dng_stream which can be read from or written to memory.

`#include <dng_memory_stream.h>`

Inheritance diagram for dng_memory_stream::



Public Member Functions

- dng_memory_stream (dng_memory_allocator &allocator, dng_abort_sniffer *sniffer=NULL, uint32 pageSize=64 *1024)
- virtual void CopyToStream (dng_stream &dstStream, uint64 count)

Protected Member Functions

- virtual uint64 **DoGetLength** ()
- virtual void **DoRead** (void *data, uint32 count, uint64 offset)
- virtual void **DoSetLength** (uint64 length)
- virtual void **DoWrite** (const void *data, uint32 count, uint64 offset)

Protected Attributes

- dng_memory_allocator & **fAllocator**
- uint32 **fPageSize**
- uint32 **fPageCount**
- uint32 **fPagesAllocated**
- dng_memory_block ** **fPageList**
- uint64 **fMemoryStreamLength**

6.38.1 Detailed Description

A dng_stream which can be read from or written to memory.

Stream is populated via writing and either read or accessed by asking for contents as a pointer.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 dng_memory_stream::dng_memory_stream (dng_memory_allocator & *allocator*, dng_abort_sniffer * *sniffer* = NULL, uint32 *pageSize* = 64 * 1024)

Construct a new memory-based stream.

Parameters:

allocator Allocator to use to allocate memory in stream as needed.

sniffer If non-NULL used to check for user cancellation.

pageSize Unit of allocation for data stored in stream.

6.38.3 Member Function Documentation

6.38.3.1 void dng_memory_stream::CopyToStream (dng_stream & *dstStream*, uint64 *count*) [virtual]

Copy a specified number of bytes to a target stream.

Parameters:

dstStream The target stream.

count The number of bytes to copy.

Reimplemented from dng_stream.

References dng_memory_block::Buffer_uint8(), dng_stream::CopyToStream(), dng_stream::Flush(), dng_stream::Length(), dng_stream::Position(), dng_stream::Put(), dng_stream::SetReadPosition(), and ThrowEndOfFile().

The documentation for this class was generated from the following files:

- dng_memory_stream.h
- dng_memory_stream.cpp

6.39 dng_mosaic_info Class Reference

Support for describing color filter array patterns and manipulating mosaic sample data.

`#include <dng_mosaic_info.h>`

Public Member Functions

- virtual void **Parse** (dng_host &host, dng_stream &stream, dng_info &info)
- virtual void **PostParse** (dng_host &host, dng_negative &negative)
- bool IsColorFilterArray () const
- virtual bool SetFourColorBayer ()
- virtual dng_point FullScale () const
- virtual dng_point DownScale (uint32 minSize, uint32 prefSize, real64 cropFactor) const
- virtual dng_point DstSize (const dng_point &downScale) const
- virtual void InterpolateGeneric (dng_host &host, dng_negative &negative, const dng_image &srcImage, dng_image &dstImage, uint32 srcPlane=0) const
- virtual void InterpolateFast (dng_host &host, dng_negative &negative, const dng_image &srcImage, dng_image &dstImage, const dng_point &downScale, uint32 srcPlane=0) const
- virtual void Interpolate (dng_host &host, dng_negative &negative, const dng_image &srcImage, dng_image &dstImage, const dng_point &downScale, uint32 srcPlane=0) const

Public Attributes

- dng_point fCFAPatternSize

  Size of fCFAPattern.

- uint8 fCFAPattern [kMaxCFAPattern][kMaxCFAPattern]

  CFA pattern from CFAPattern tag in the TIFF/EP specification..

- uint32 fColorPlanes

Number of color planes in DNG input.

- uint8 **fCFAPlaneColor** [kMaxColorPlanes]
- uint32 fCFALayout
- uint32 fBayerGreenSplit

Protected Member Functions

- virtual bool **IsSafeDownScale** (const dng_point &downScale) const
- uint32 **SizeForDownScale** (const dng_point &downScale) const
- virtual bool **ValidSizeDownScale** (const dng_point &downScale, uint32 minSize) const

Protected Attributes

- dng_point **fSrcSize**
- dng_point **fCroppedSize**
- real64 **fAspectRatio**

6.39.1 Detailed Description

Support for describing color filter array patterns and manipulating mosaic sample data.

See CFAPattern tag in TIFF/EP specification and CFAPlaneColor, CFALayout, and BayerGreenSplit tags in the DNG 1.1.0 specification.

6.39.2 Member Function Documentation

6.39.2.1 bool dng_mosaic_info::IsColorFilterArray () const [inline]

Returns whether the RAW data in this DNG file from a color filter array (mosaiced) source.

Return values:

true if this DNG file is from a color filter array (mosiaced) source.

References fCFAPatternSize.

Referenced by DownScale(), and dng_image_writer::WriteDNG().

6.39.2.2 bool dng_mosaic_info::SetFourColorBayer () [virtual]

Enable generating four-plane output from three-plane Bayer input. Extra plane is a second version of the green channel. First green is produced using green mosaic samples from one set of rows/columns (even/odd) and the second green channel is produced using the other set of rows/columns. One can compare the two versions to judge whether BayerGreenSplit needs to be set for a given input source.

References fCFALayout, fCFAPattern, fCFAPatternSize, and fColorPlanes.

6.39.2.3 dng_point dng_mosaic_info::FullScale () const [virtual]

Returns scaling factor relative to input size needed to capture output data. Staggered (or rotated) sensing arrays are produced to a larger output than the number of input samples. This method indicates how much larger.

Return values:

a point with integer scaling factors for the horizotal and vertical dimensions.

References fCFALayout.

Referenced by DstSize(), and InterpolateGeneric().

6.39.2.4 dng_point dng_mosaic_info::DownScale (uint32 *minSize*, uint32 *prefSize*, real64 *cropFactor*) const [virtual]

Returns integer factors by which mosaic data must be downsampled to produce an image which is as close to prefSize as possible in longer dimension, but no smaller than minSize.

Parameters:

minSize Number of pixels as minium for longer dimension of downsampled image.

prefSize Number of pixels as target for longer dimension of downsampled image.

cropFactor Faction of the image to be used after cropping.

Return values:

Point containing integer factors by which image must be downsampled.

References IsColorFilterArray().

6.39.2.5 dng_point dng_mosaic_info::DstSize (const dng_point & *downScale*) const [virtual]

Return size of demosaiced image for passed in downscaling factor.

Parameters:

downScale Integer downsampling factor obtained from DownScale method.

Return values:

Size of resulting demosaiced image.

References FullScale().

6.39.2.6 void dng_mosaic_info::InterpolateGeneric (dng_host & *host*, dng_negative & *negative*, const dng_image & *srcImage*, dng_image & *dstImage*, uint32 *srcPlane* = 0) const [virtual]

Demosaic interpolation of a single plane for non-downsampled case.

Parameters:

host dng_host to use for buffer allocation requests, user cancellation testing, and progress updates.

srcImage Source image for mosaiced data.

dstImage Destination image for resulting interpolated data.

srcPlane Which plane to interpolate.

References dng_host::Allocate(), dng_image::Bounds(), dng_image::edge_repeat, dng_pixel_buffer::fArea, fCFAPatternSize, dng_pixel_buffer::fColStep, dng_pixel_buffer::fData, dng_pixel_buffer::fPixelRange, dng_pixel_buffer::fPixelSize, dng_pixel_buffer::fPixelType, dng_pixel_buffer::fPlane, dng_pixel_buffer::fPlanes, dng_pixel_buffer::fPlaneStep, dng_pixel_buffer::fRowStep, FullScale(), dng_image::Get(), dng_image::PixelRange(), dng_image::PixelSize(), dng_image::PixelType(), dng_image::Put(), dng_image::RepeatingTile(), and dng_host::SniffForAbort().

Referenced by Interpolate().

6.39.2.7 void dng_mosaic_info::InterpolateFast (dng_host & *host*, dng_negative & *negative*, const dng_image & *srcImage*, dng_image & *dstImage*, const dng_point & *downScale*, uint32 *srcPlane* = 0) const [virtual]

Demosaic interpolation of a single plane for downsampled case.

Parameters:

host dng_host to use for buffer allocation requests, user cancellation testing, and progress updates.

srcImage Source image for mosaiced data.

dstImage Destination image for resulting interpolated data.

downScale Amount (in horizontal and vertical) by which to subsample image.

srcPlane Which plane to interpolate.

References dng_image::Bounds(), and dng_host::PerformAreaTask().

Referenced by Interpolate().

6.39.2.8 void dng_mosaic_info::Interpolate (dng_host & *host*, dng_negative & *negative*, const dng_image & *srcImage*, dng_image & *dstImage*, const dng_point & *downScale*, uint32 *srcPlane* = `0`) const `[virtual]`

Demosaic interpolation of a single plane. Chooses between generic and fast interpolators based on parameters.

Parameters:

host dng_host to use for buffer allocation requests, user cancellation testing, and progress updates.

srcImage Source image for mosaiced data.

dstImage Destination image for resulting interpolated data.

downScale Amount (in horizontal and vertical) by which to subsample image.

srcPlane Which plane to interpolate.

References InterpolateFast(), and InterpolateGeneric().

6.39.3 Member Data Documentation

6.39.3.1 uint32 dng_mosaic_info::fCFALayout

Value of CFALayout tag in the DNG 1.1.0 specification. CFALayout describes the spatial layout of the CFA. The currently defined values are:

- 1 = Rectangular (or square) layout.

2 = Staggered layout A: even columns are offset down by 1/2 row.

- 3 = Staggered layout B: even columns are offset up by 1/2 row.
- 4 = Staggered layout C: even rows are offset right by 1/2 column.
- 5 = Staggered layout D: even rows are offset left by 1/2 column.

Referenced by FullScale(), and SetFourColorBayer().

6.39.3.2 uint32 dng_mosaic_info::fBayerGreenSplit

Value of BayerGreeSplit tag in DNG file. BayerGreenSplit only applies to CFA images using a Bayer pattern filter array. This tag specifies, in arbitrary units, how closely the values of the green pixels in the blue/green rows track the values of the green pixels in the red/green rows.

A value of zero means the two kinds of green pixels track closely, while a non-zero value means they sometimes diverge. The useful range for this tag is from 0 (no divergence) to about 5000 (large divergence).

The documentation for this class was generated from the following files:

- dng_mosaic_info.h
- dng_mosaic_info.cpp

6.40 dng_negative Class Reference

Main class for holding DNG image data and associated metadata.

`#include <dng_negative.h>`

Public Member Functions

- dng_memory_allocator & Allocator () const

  Provide access to the memory allocator used for this object.

- void SetModelName (const char *name)

  Getter for ModelName.

- const dng_string & ModelName () const

  Setter for ModelName.

- void SetLocalName (const char *name)

  Setter for LocalName.

- const dng_string & LocalName () const

  Getter for LocalName.

- void SetBaseOrientation (const dng_orientation &orientation)

  Setter for BaseOrientation.

- bool HasBaseOrientation () const

  Has BaseOrientation been set?

- const dng_orientation & BaseOrientation () const

  Getter for BaseOrientation.

- virtual dng_orientation Orientation () const

  Hook to allow SDK host code to add additional rotations.

- void ApplyOrientation (const dng_orientation &orientation)
- void SetDefaultCropSize (const dng_urational &sizeH, const dng_urational &sizeV)

  Setter for DefaultCropSize.

- void SetDefaultCropSize (uint32 sizeH, uint32 sizeV)

  Setter for DefaultCropSize.

- const dng_urational & DefaultCropSizeH () const

  Getter for DefaultCropSize horizontal.

- const dng_urational & DefaultCropSizeV () const

  Getter for DefaultCropSize vertical.

- void SetDefaultCropOrigin (const dng_urational &originH, const dng_urational &originV)

  Setter for DefaultCropOrigin.

- void SetDefaultCropOrigin (uint32 originH, uint32 originV)

  Setter for DefaultCropOrigin.

- void SetDefaultCropCentered (const dng_point &rawSize)

  Set default crop around center of image.

- const dng_urational & DefaultCropOriginH () const

  Get default crop origin horizontal value.

- const dng_urational & DefaultCropOriginV () const

  Get default crop origin vertical value.

- void SetDefaultScale (const dng_urational &scaleH, const dng_urational &scaleV)

  Setter for DefaultScale.

- const dng_urational & DefaultScaleH () const

  Get default scale horizontal value.

- const dng_urational & DefaultScaleV () const

  Get default scale vertical value.

- void SetBestQualityScale (const dng_urational &scale)

  Setter for BestQualityScale.

- const dng_urational & BestQualityScale () const

  Getter for BestQualityScale.

- real64 RawToFullScaleH () const

  API for raw to full image scaling factors horizontal.

- real64 RawToFullScaleV () const

  API for raw to full image scaling factors vertical.

- real64 DefaultScale () const
- real64 SquareWidth () const

  Default cropped image size (at scale == 1.0) width.

- real64 SquareHeight () const

  Default cropped image size (at scale == 1.0) height.

- real64 AspectRatio () const

  Default cropped image aspect ratio.

- uint32 FinalWidth (real64 scale) const

  Default cropped image size at given scale factor width.

- uint32 FinalHeight (real64 scale) const

  Default cropped image size at given scale factor height.

- uint32 DefaultFinalWidth () const

  Default cropped image size at default scale factor width.

- uint32 DefaultFinalHeight () const

  Default cropped image size at default scale factor height.

- uint32 BestQualityFinalWidth () const
- uint32 BestQualityFinalHeight () const
- dng_rect DefaultCropArea (real64 scaleH=1.0, real64 scaleV=1.0) const
- void SetBaselineNoise (real64 noise)

  Setter for BaselineNoise.

- const dng_urational & BaselineNoiseR () const
  Getter for BaselineNoise as dng_urational.
- real64 BaselineNoise () const
  Getter for BaselineNoise as real64.
- void SetNoiseReductionApplied (const dng_urational &value)
  Setter for NoiseReductionApplied.
- const dng_urational & NoiseReductionApplied () const
  Getter for NosieReductionApplied.
- void SetBaselineExposure (real64 exposure)
  Setter for BaselineExposure.
- const dng_srational & BaselineExposureR () const
  Getter for BaselineExposure as dng_urational.
- real64 BaselineExposure () const
  Getter for BaselineExposure as real64.
- void SetBaselineSharpness (real64 sharpness)
  Setter for BaselineSharpness.
- const dng_urational & BaselineSharpnessR () const
  Getter for BaselineSharpness as dng_urational.
- real64 BaselineSharpness () const
  Getter for BaselineSharpness as real64.
- void SetChromaBlurRadius (const dng_urational &radius)
  Setter for ChromaBlurRadius.
- const dng_urational & ChromaBlurRadius () const
  Getter for ChromaBlurRadius as dng_urational.
- void SetAntiAliasStrength (const dng_urational &strength)
  Setter for AntiAliasStrength.
- const dng_urational & AntiAliasStrength () const
  Getter for AntiAliasStrength as dng_urational.

- void SetLinearResponseLimit (real64 limit)

  Setter for LinearResponseLimit.

- const dng_urational & LinearResponseLimitR () const

  Getter for LinearResponseLimit as dng_urational.

- real64 LinearResponseLimit () const

  Getter for LinearResponseLimit as real64.

- void SetShadowScale (const dng_urational &scale)

  Setter for ShadowScale.

- const dng_urational & ShadowScaleR () const

  Getter for ShadowScale as dng_urational.

- real64 ShadowScale () const

  Getter for ShadowScale as real64.

- void **SetColorimetricReference** (uint32 ref)
- uint32 **ColorimetricReference** () const
- void SetColorChannels (uint32 channels)

  Setter for ColorChannels.

- uint32 ColorChannels () const

  Getter for ColorChannels.

- void SetMonochrome ()

  Setter for Monochrome.

- bool IsMonochrome () const

  Getter for Monochrome.

- void SetAnalogBalance (const dng_vector &b)

  Setter for AnalogBalance.

- dng_urational AnalogBalanceR (uint32 channel) const

  Getter for AnalogBalance as dng_urational.

- real64 AnalogBalance (uint32 channel) const

  Getter for AnalogBalance as real64.

- void SetCameraNeutral (const dng_vector &n)

  Setter for CameraNeutral.

- void ClearCameraNeutral ()

  Clear CameraNeutral.

- bool HasCameraNeutral () const

  Determine if CameraNeutral has been set but not cleared.

- const dng_vector & CameraNeutral () const

  Getter for CameraNeutral.

- dng_urational **CameraNeutralR** (uint32 channel) const
- void SetCameraWhiteXY (const dng_xy_coord &coord)

  Setter for CameraWhiteXY.

- bool **HasCameraWhiteXY** () const
- const dng_xy_coord & **CameraWhiteXY** () const
- void **GetCameraWhiteXY** (dng_urational &x, dng_urational &y) const
- void SetCameraCalibration1 (const dng_matrix &m)
- void SetCameraCalibration2 (const dng_matrix &m)
- const dng_matrix & CameraCalibration1 () const

  Getter for first of up to two color matrices used for individual camera calibrations.

- const dng_matrix & CameraCalibration2 () const

  Getter for second of up to two color matrices used for individual camera calibrations.

- void **SetCameraCalibrationSignature** (const char *signature)
- const dng_string & **CameraCalibrationSignature** () const
- void **AddProfile** (AutoPtr< dng_camera_profile > &profile)
- void **ClearProfiles** ()
- uint32 **ProfileCount** () const
- const dng_camera_profile & **ProfileByIndex** (uint32 index) const
- const dng_camera_profile * **ProfileByID** (const dng_camera_profile_id &id, bool useDefaultIfNoMatch=true) const
- bool **HasProfileID** (const dng_camera_profile_id &id) const
- virtual const dng_camera_profile * **CameraProfileToEmbed** () const
- void **SetAsShotProfileName** (const char *name)
- const dng_string & **AsShotProfileName** () const
- virtual dng_color_spec * **MakeColorSpec** (const dng_camera_profile_id &id) const
- void **SetRawImageDigest** (const dng_fingerprint &digest)
- void **ClearRawImageDigest** ()

- const dng_fingerprint **RawImageDigest** () const
- void **FindRawImageDigest** (dng_host &host) const
- void **ValidateRawImageDigest** (dng_host &host) const
- void **SetRawDataUniqueID** (const dng_fingerprint &id)
- const dng_fingerprint & **RawDataUniqueID** () const
- void **FindRawDataUniqueID** (dng_host &host) const
- void **SetOriginalRawFileName** (const char *name)
- bool **HasOriginalRawFileName** () const
- const dng_string & **OriginalRawFileName** () const
- void **SetHasOriginalRawFileData** (bool hasData)
- bool **CanEmbedOriginalRaw** () const
- void **SetOriginalRawFileData** (AutoPtr< dng_memory_block > &data)
- const void * **OriginalRawFileData** () const
- uint32 **OriginalRawFileDataLength** () const
- void **SetOriginalRawFileDigest** (const dng_fingerprint &digest)
- const dng_fingerprint & **OriginalRawFileDigest** () const
- void **FindOriginalRawFileDigest** () const
- void **ValidateOriginalRawFileDigest** () const
- void **SetPrivateData** (AutoPtr< dng_memory_block > &block)
- void **ClearPrivateData** ()
- const uint8 * **PrivateData** () const
- uint32 **PrivateLength** () const
- void **SetMakerNoteSafety** (bool safe)
- bool **IsMakerNoteSafe** () const
- void **SetMakerNote** (AutoPtr< dng_memory_block > &block)
- void **ClearMakerNote** ()
- const void * **MakerNoteData** () const
- uint32 **MakerNoteLength** () const
- dng_exif * **GetExif** ()
- const dng_exif * **GetExif** () const
- virtual dng_memory_block * **BuildExifBlock** (const dng_resolution *resolution=NULL, bool includeIPTC=false) const
- dng_exif * **GetOriginalExif** ()
- const dng_exif * **GetOriginalExif** () const
- void **SetIPTC** (AutoPtr< dng_memory_block > &block, uint64 offset)
- void **SetIPTC** (AutoPtr< dng_memory_block > &block)
- void **ClearIPTC** ()
- const void * **IPTCData** () const
- uint32 **IPTCLength** () const
- uint64 **IPTCOffset** () const
- dng_fingerprint **IPTCDigest** () const
- void **RebuildIPTC** ()

- bool **SetXMP** (dng_host &host, const void *buffer, uint32 count, bool xmpInSidecar=false, bool xmpIsNewer=false)
- dng_xmp * **GetXMP** ()
- const dng_xmp * **GetXMP** () const
- bool **XMPinSidecar** () const
- const dng_linearization_info * **GetLinearizationInfo** () const
- void **ClearLinearizationInfo** ()
- void **SetLinearization** (AutoPtr< dng_memory_block > &curve)
- void **SetActiveArea** (const dng_rect &area)
- void **SetMaskedAreas** (uint32 count, const dng_rect *area)
- void **SetMaskedArea** (const dng_rect &area)
- void **SetBlackLevel** (real64 black, int32 plane=-1)
- void **SetQuadBlacks** (real64 black0, real64 black1, real64 black2, real64 black3)
- void **SetRowBlacks** (const real64 *blacks, uint32 count)
- void **SetColumnBlacks** (const real64 *blacks, uint32 count)
- uint32 **WhiteLevel** (uint32 plane=0) const
- void **SetWhiteLevel** (uint32 white, int32 plane=-1)
- const dng_mosaic_info * **GetMosaicInfo** () const
- void **ClearMosaicInfo** ()
- void **SetColorKeys** (ColorKeyCode color0, ColorKeyCode color1, ColorKeyCode color2, ColorKeyCode color3=colorKeyMaxEnum)
- void **SetRGB** ()
- void **SetCMY** ()
- void **SetGMCY** ()
- void **SetBayerMosaic** (uint32 phase)
- void **SetFujiMosaic** (uint32 phase)
- void **SetQuadMosaic** (uint32 pattern)
- void **SetGreenSplit** (uint32 split)
- virtual void **Parse** (dng_host &host, dng_stream &stream, dng_info &info)
- virtual void **PostParse** (dng_host &host, dng_stream &stream, dng_info &info)

- virtual void **SynchronizeMetadata** ()
- void **UpdateDateTime** (const dng_date_time_info &dt)
- void **UpdateDateTimeToNow** ()
- virtual bool **SetFourColorBayer** ()
- const dng_image * **Stage1Image** () const
- const dng_image * **Stage2Image** () const
- const dng_image * **Stage3Image** () const
- const dng_image & **RawImage** () const
- virtual void **ReadStage1Image** (dng_host &host, dng_stream &stream, dng_info &info)
- void **SetStage1Image** (AutoPtr< dng_image > &image)

- virtual void **BuildStage2Image** (dng_host &host, uint32 pixelType=ttShort)
- virtual void **ClearStage1** ()
- virtual void **BuildStage3Image** (dng_host &host, int32 srcPlane=-1)
- void **SetStage3Gain** (real64 gain)
- real64 **Stage3Gain** () const
- virtual void **ClearStage2** ()
- virtual void **ClearStage3** ()
- void **SetIsPreview** (bool preview)
- bool **IsPreview** () const

Static Public Member Functions

- static dng_negative * **Make** (dng_memory_allocator &allocator)

Protected Member Functions

- **dng_negative** (dng_memory_allocator &allocator)
- virtual void **Initialize** ()
- virtual dng_exif * **MakeExif** ()
- virtual dng_xmp * **MakeXMP** ()
- virtual dng_linearization_info * **MakeLinearizationInfo** ()
- void **NeedLinearizationInfo** ()
- virtual dng_mosaic_info * **MakeMosaicInfo** ()
- void **NeedMosaicInfo** ()
- virtual void **DoBuildStage2** (dng_host &host, uint32 pixelType)
- virtual void **DoBuildStage3** (dng_host &host, int32 srcPlane)
- virtual void **DoMergeStage3** (dng_host &host)

Protected Attributes

- dng_memory_allocator & **fAllocator**
- dng_string **fModelName**
- dng_string **fLocalName**
- bool **fHasBaseOrientation**
- dng_orientation **fBaseOrientation**
- dng_urational **fDefaultCropSizeH**
- dng_urational **fDefaultCropSizeV**
- dng_urational **fDefaultCropOriginH**
- dng_urational **fDefaultCropOriginV**
- dng_urational **fDefaultScaleH**
- dng_urational **fDefaultScaleV**
- dng_urational **fBestQualityScale**

- real64 **fRawToFullScaleH**
- real64 **fRawToFullScaleV**
- dng_urational **fBaselineNoise**
- dng_urational **fNoiseReductionApplied**
- dng_srational **fBaselineExposure**
- dng_urational **fBaselineSharpness**
- dng_urational **fChromaBlurRadius**
- dng_urational **fAntiAliasStrength**
- dng_urational **fLinearResponseLimit**
- dng_urational **fShadowScale**
- uint32 **fColorimetricReference**
- uint32 **fColorChannels**
- dng_vector **fAnalogBalance**
- dng_vector **fCameraNeutral**
- dng_xy_coord **fCameraWhiteXY**
- dng_matrix **fCameraCalibration1**
- dng_matrix **fCameraCalibration2**
- dng_string **fCameraCalibrationSignature**
- std::vector< dng_camera_profile * > **fCameraProfile**
- dng_string **fAsShotProfileName**
- dng_fingerprint **fRawImageDigest**
- dng_fingerprint **fRawDataUniqueID**
- dng_string **fOriginalRawFileName**
- bool **fHasOriginalRawFileData**
- AutoPtr< dng_memory_block > **fOriginalRawFileData**
- dng_fingerprint **fOriginalRawFileDigest**
- AutoPtr< dng_memory_block > **fDNGPrivateData**
- bool **fIsMakerNoteSafe**
- AutoPtr< dng_memory_block > **fMakerNote**
- AutoPtr< dng_exif > **fExif**
- AutoPtr< dng_exif > **fOriginalExif**
- AutoPtr< dng_memory_block > **fIPTCBlock**
- uint64 **fIPTCOffset**
- AutoPtr< dng_xmp > **fXMP**
- bool **fValidEmbeddedXMP**
- bool **fXMPinSidecar**
- bool **fXMPisNewer**
- AutoPtr< dng_linearization_info > **fLinearizationInfo**
- AutoPtr< dng_mosaic_info > **fMosaicInfo**
- AutoPtr< dng_image > **fStage1Image**
- AutoPtr< dng_image > **fStage2Image**
- AutoPtr< dng_image > **fStage3Image**
- real64 **fStage3Gain**
- bool **fIsPreview**

6.40.1 Detailed Description

Main class for holding DNG image data and associated metadata.

6.40.2 Member Function Documentation

6.40.2.1 void dng_negative::ApplyOrientation (const dng_orientation & *orientation*)

Logically rotates the image by changing the orientation values. This will also update the XMP data.

6.40.2.2 real64 dng_negative::DefaultScale () const [inline]

Get default scale factor. When specifing a single scale factor, we use the horizontal scale factor, and let the vertical scale factor be calculated based on the pixel aspect ratio.

References DefaultScaleH().

Referenced by BestQualityFinalHeight(), BestQualityFinalWidth(), DefaultFinalHeight(), and DefaultFinalWidth().

6.40.2.3 uint32 dng_negative::BestQualityFinalWidth () const [inline]

Get best quality width. For a naive conversion, one could use either the default size, or the best quality size.

References BestQualityScale(), DefaultScale(), and FinalWidth().

6.40.2.4 uint32 dng_negative::BestQualityFinalHeight () const [inline]

Get best quality height. For a naive conversion, one could use either the default size, or the best quality size.

References BestQualityScale(), DefaultScale(), and FinalHeight().

6.40.2.5 dng_rect dng_negative::DefaultCropArea (real64 *scaleH* = 1.0, real64 *scaleV* = 1.0) const

The default crop area after applying the specified horizontal and vertical scale factors to the stage 3 image.

References DNG_ASSERT, dng_image::Height(), and dng_image::Width().

Referenced by dng_render::Render().

6.40.2.6 void dng_negative::SetCameraCalibration1 (const dng_matrix & *m*)

Setter for first of up to two color matrices used for individual camera calibrations.

The sequence of matrix transforms is: Camera data –> camera calibration –> "inverse" of color matrix

This will be a 4x4 matrix for a four-color camera. The defaults are almost always the identity matrix, and for the cases where they aren't, they are diagonal matrices.

6.40.2.7 void dng_negative::SetCameraCalibration2 (const dng_matrix & *m*)

Setter for second of up to two color matrices used for individual camera calibrations.

The sequence of matrix transforms is: Camera data –> camera calibration –> "inverse" of color matrix

This will be a 4x4 matrix for a four-color camera. The defaults are almost always the identity matrix, and for the cases where they aren't, they are diagonal matrices.

The documentation for this class was generated from the following files:

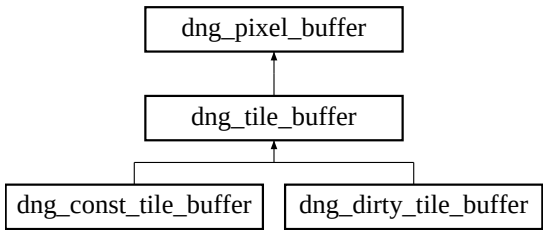
- dng_negative.h
- dng_negative.cpp

6.41 dng_pixel_buffer Class Reference

Holds a buffer of pixel data with "pixel geometry" metadata.

`#include <dng_pixel_buffer.h>`

Inheritance diagram for dng_pixel_buffer::



Public Member Functions

- **dng_pixel_buffer** (const dng_pixel_buffer &buffer)
- dng_pixel_buffer & **operator=** (const dng_pixel_buffer &buffer)
- uint32 PixelRange () const
- const dng_rect & Area () const

- uint32 Planes () const
- int32 RowStep () const
- int32 PlaneStep () const
- const void * ConstPixel (int32 row, int32 col, uint32 plane=0) const
- void * DirtyPixel (int32 row, int32 col, uint32 plane=0)
- const uint8 * ConstPixel_uint8 (int32 row, int32 col, uint32 plane=0) const
- uint8 * DirtyPixel_uint8 (int32 row, int32 col, uint32 plane=0)
- const int8 * ConstPixel_int8 (int32 row, int32 col, uint32 plane=0) const
- int8 * DirtyPixel_int8 (int32 row, int32 col, uint32 plane=0)
- const uint16 * ConstPixel_uint16 (int32 row, int32 col, uint32 plane=0) const
- uint16 * DirtyPixel_uint16 (int32 row, int32 col, uint32 plane=0)
- const int16 * ConstPixel_int16 (int32 row, int32 col, uint32 plane=0) const
- int16 * DirtyPixel_int16 (int32 row, int32 col, uint32 plane=0)
- const uint32 * ConstPixel_uint32 (int32 row, int32 col, uint32 plane=0) const
- uint32 * DirtyPixel_uint32 (int32 row, int32 col, uint32 plane=0)
- const int32 * ConstPixel_int32 (int32 row, int32 col, uint32 plane=0) const
- int32 * DirtyPixel_int32 (int32 row, int32 col, uint32 plane=0)
- const real32 * ConstPixel_real32 (int32 row, int32 col, uint32 plane=0) const
- real32 * DirtyPixel_real32 (int32 row, int32 col, uint32 plane=0)
- void SetConstant (const dng_rect &area, uint32 plane, uint32 planes, uint32 value)
- void SetConstant_uint8 (const dng_rect &area, uint32 plane, uint32 planes, uint8 value)
- void SetConstant_uint16 (const dng_rect &area, uint32 plane, uint32 planes, uint16 value)
- void SetConstant_int16 (const dng_rect &area, uint32 plane, uint32 planes, int16 value)
- void SetConstant_uint32 (const dng_rect &area, uint32 plane, uint32 planes, uint32 value)
- void SetConstant_real32 (const dng_rect &area, uint32 plane, uint32 planes, real32 value)
- void SetZero (const dng_rect &area, uint32 plane, uint32 planes)
- void CopyArea (const dng_pixel_buffer &src, const dng_rect &area, uint32 srcPlane, uint32 dstPlane, uint32 planes)
- void CopyArea (const dng_pixel_buffer &src, const dng_rect &area, uint32 plane, uint32 planes)
- void RepeatArea (const dng_rect &srcArea, const dng_rect &dstArea)
- void ShiftRight (uint32 shift)
- void FlipH ()
- void FlipV ()
- void FlipZ ()
- bool EqualArea (const dng_pixel_buffer &rhs, const dng_rect &area, uint32 plane, uint32 planes) const

Static Public Member Functions

- static dng_point RepeatPhase (const dng_rect &srcArea, const dng_rect &dstArea)

Public Attributes

- dng_rect **fArea**
- uint32 **fPlane**
- uint32 **fPlanes**
- int32 **fRowStep**
- int32 **fColStep**
- int32 **fPlaneStep**
- uint32 **fPixelType**
- uint32 **fPixelSize**
- uint32 **fPixelRange**
- void * **fData**
- bool **fDirty**

6.41.1 Detailed Description

Holds a buffer of pixel data with "pixel geometry" metadata.

The pixel geometry describes the layout in terms of how many planes, rows and columns plus the steps (in bytes) between each column, row and plane.

6.41.2 Member Function Documentation

6.41.2.1 uint32 dng_pixel_buffer::PixelRange () const

Get the range of pixel values.

Return values:

Range of value a pixel can take. (Meaning [0, max] for unsigned case. Signed case is biased so [-32768, max - 32768].)

Referenced by CopyArea().

6.41.2.2 const dng_rect& dng_pixel_buffer::Area () const [inline]

Get extent of pixels in buffer

Return values:

Rectangle giving valid extent of buffer.

6.41.2.3 uint32 dng_pixel_buffer::Planes () const [inline]

Number of planes of image data.

Return values:

Number of planes held in buffer.

6.41.2.4 int32 dng_pixel_buffer::RowStep () const [inline]

Step, in pixels not bytes, between rows of data in buffer.

Return values:

row step in pixels. May be negative.

6.41.2.5 int32 dng_pixel_buffer::PlaneStep () const [inline]

Step, in pixels not bytes, between planes of data in buffer.

Return values:

plane step in pixels. May be negative.

6.41.2.6 const void* dng_pixel_buffer::ConstPixel (int32 *row*, int32 *col*, uint32 *plane* = 0) const [inline]

Get read-only untyped (void *) pointer to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as void *.

Referenced by ConstPixel_int16(), ConstPixel_int32(), ConstPixel_int8(), ConstPixel_real32(), ConstPixel_uint16(), ConstPixel_uint32(), ConstPixel_uint8(), CopyArea(), EqualArea(), dng_image::Put(), and RepeatArea().

6.41.2.7 void* dng_pixel_buffer::DirtyPixel (int32 *row*, int32 *col*, uint32 *plane* = 0) [inline]

Get a writable untyped (void *) pointer to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as void *.

References DNG_ASSERT.

Referenced by CopyArea(), DirtyPixel_int16(), DirtyPixel_int32(), DirtyPixel_int8(), DirtyPixel_real32(), DirtyPixel_uint16(), DirtyPixel_uint32(), DirtyPixel_uint8(), dng_image::Get(), RepeatArea(), dng_simple_image::Rotate(), SetConstant(), ShiftRight(), and dng_simple_image::Trim().

6.41.2.8 const uint8* dng_pixel_buffer::ConstPixel_uint8 (int32 *row*, int32 *col*, uint32 *plane* = 0) const [inline]

Get read-only uint8 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as uint8 *.

References ConstPixel().

6.41.2.9 uint8* dng_pixel_buffer::DirtyPixel_uint8 (int32 *row*, int32 *col*, uint32 *plane* = 0) [inline]

Get a writable uint8 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as uint8 *.

References DirtyPixel().

6.41.2.10 const int8* dng_pixel_buffer::ConstPixel_int8 (int32 *row*, int32 *col*, uint32 *plane* = 0) const [inline]

Get read-only int8 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as int8 *.

References ConstPixel().

6.41.2.11 int8* dng_pixel_buffer::DirtyPixel_int8 (int32 *row*, int32 *col*, uint32 *plane* = 0) [inline]

Get a writable int8 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as int8 *.

References DirtyPixel().

6.41.2.12 const uint16* dng_pixel_buffer::ConstPixel_uint16 (int32 *row*, int32 *col*, uint32 *plane* = 0) const [inline]

Get read-only uint16 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as uint16 *.

References ConstPixel().

6.41.2.13 uint16* dng_pixel_buffer::DirtyPixel_uint16 (int32 *row*, int32 *col*, uint32 *plane* = 0) [inline]

Get a writable uint16 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as uint16 *.

References DirtyPixel().

6.41.2.14 const int16* dng_pixel_buffer::ConstPixel_int16 (int32 *row*, int32 *col*, uint32 *plane* = 0) const [inline]

Get read-only int16 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as int16 *.

References ConstPixel().

6.41.2.15 int16* dng_pixel_buffer::DirtyPixel_int16 (int32 *row*, int32 *col*, uint32 *plane* = 0) [inline]

Get a writable int16 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as int16 *.

References DirtyPixel().

6.41.2.16 const uint32* dng_pixel_buffer::ConstPixel_uint32 (int32 *row*, int32 *col*, uint32 *plane* = 0) const [inline]

Get read-only uint32 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as uint32 *.

References ConstPixel().

6.41.2.17 uint32* dng_pixel_buffer::DirtyPixel_uint32 (int32 *row*, int32 *col*, uint32 *plane* = 0) [inline]

Get a writable uint32 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as uint32 *.

References DirtyPixel().

6.41.2.18 const int32* dng_pixel_buffer::ConstPixel_int32 (int32 *row*, int32 *col*, uint32 *plane* = 0) const [inline]

Get read-only int32 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as int32 *.

References ConstPixel().

6.41.2.19 int32* dng_pixel_buffer::DirtyPixel_int32 (int32 *row*, int32 *col*, uint32 *plane* = 0) [inline]

Get a writable int32 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as int32 *.

References DirtyPixel().

6.41.2.20 const real32* dng_pixel_buffer::ConstPixel_real32 (int32 *row*, int32 *col*, uint32 *plane* = 0) const [inline]

Get read-only real32 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as real32 *.

References ConstPixel().

6.41.2.21 real32* dng_pixel_buffer::DirtyPixel_real32 (int32 *row*, int32 *col*, uint32 *plane* = 0) [inline]

Get a writable real32 * to pixel data starting at a specific pixel in the buffer.

Parameters:

row Start row for buffer pointer.

col Start column for buffer pointer.

plane Start plane for buffer pointer.

Return values:

Pointer to pixel data as real32 *.

References DirtyPixel().

6.41.2.22 void dng_pixel_buffer::SetConstant (const dng_rect & *area*, uint32 *plane*, uint32 *planes*, uint32 *value*)

Initialize a rectangular area of pixel buffer to a constant.

Parameters:

area Rectangle of pixel buffer to set.

plane Plane to start filling on.

planes Number of planes to fill.

value Constant value to set pixels to.

References DirtyPixel(), and ThrowNotYetImplemented().

Referenced by SetConstant_int16(), SetConstant_real32(), SetConstant_uint16(), SetConstant_uint32(), SetConstant_uint8(), and SetZero().

6.41.2.23 void dng_pixel_buffer::SetConstant_uint8 (const dng_rect & *area*, uint32 *plane*, uint32 *planes*, uint8 *value*) [inline]

Initialize a rectangular area of pixel buffer to a constant unsigned 8-bit value.

Parameters:

area Rectangle of pixel buffer to set.

plane Plane to start filling on.

planes Number of planes to fill.

value Constant uint8 value to set pixels to.

References DNG_ASSERT, and SetConstant().

6.41.2.24 void dng_pixel_buffer::SetConstant_uint16 (const dng_rect & *area*, uint32 *plane*, uint32 *planes*, uint16 *value*) [inline]

Initialize a rectangular area of pixel buffer to a constant unsigned 16-bit value.

Parameters:

area Rectangle of pixel buffer to set.

plane Plane to start filling on.

planes Number of planes to fill.

value Constant uint16 value to set pixels to.

References DNG_ASSERT, and SetConstant().

6.41.2.25 void dng_pixel_buffer::SetConstant_int16 (const dng_rect & *area*, uint32 *plane*, uint32 *planes*, int16 *value*) [inline]

Initialize a rectangular area of pixel buffer to a constant signed 16-bit value.

Parameters:

area Rectangle of pixel buffer to set.

plane Plane to start filling on.

planes Number of planes to fill.

value Constant int16 value to set pixels to.

References DNG_ASSERT, and SetConstant().

6.41.2.26 void dng_pixel_buffer::SetConstant_uint32 (const dng_rect & *area*, uint32 *plane*, uint32 *planes*, uint32 *value*) [inline]

Initialize a rectangular area of pixel buffer to a constant unsigned 32-bit value.

Parameters:

area Rectangle of pixel buffer to set.

plane Plane to start filling on.

planes Number of planes to fill.

value Constant uint32 value to set pixels to.

References DNG_ASSERT, and SetConstant().

6.41.2.27 void dng_pixel_buffer::SetConstant_real32 (const dng_rect & *area*, uint32 *plane*, uint32 *planes*, real32 *value*) [inline]

Initialize a rectangular area of pixel buffer to a constant real 32-bit value.

Parameters:

area Rectangle of pixel buffer to set.

plane Plane to start filling on.

planes Number of planes to fill.

value Constant real32 value to set pixels to.

References DNG_ASSERT, and SetConstant().

6.41.2.28 void dng_pixel_buffer::SetZero (const dng_rect & *area*, uint32 *plane*, uint32 *planes*)

Initialize a rectangular area of pixel buffer to zeros.

Parameters:

area Rectangle of pixel buffer to zero.

area Area to zero

plane Plane to start filling on.

planes Number of planes to fill.

References SetConstant(), and ThrowNotYetImplemented().

6.41.2.29 void dng_pixel_buffer::CopyArea (const dng_pixel_buffer & *src*, const dng_rect & *area*, uint32 *srcPlane*, uint32 *dstPlane*, uint32 *planes*)

Copy image data from an area of one pixel buffer to same area of another.

Parameters:

src Buffer to copy from.

area Rectangle of pixel buffer to copy.

srcPlane Plane to start copy in src.

dstPlane Plane to start copy in dst.

planes Number of planes to copy.

References ConstPixel(), DirtyPixel(), fColStep, fPixelSize, fPixelType, fPlaneStep, fRowStep, PixelRange(), and ThrowNotYetImplemented().

Referenced by CopyArea(), and dng_image::CopyArea().

6.41.2.30 void dng_pixel_buffer::CopyArea (const dng_pixel_buffer & *src*, const dng_rect & *area*, uint32 *plane*, uint32 *planes*) [inline]

Copy image data from an area of one pixel buffer to same area of another.

Parameters:

src Buffer to copy from.

area Rectangle of pixel buffer to copy.

plane Plane to start copy in src and this.

planes Number of planes to copy.

References CopyArea().

6.41.2.31 dng_point dng_pixel_buffer::RepeatPhase (const dng_rect & *srcArea*, const dng_rect & *dstArea*) [static]

Calculate the offset phase of destination rectangle relative to source rectangle. Phase is based on a 0,0 origin and the notion of repeating srcArea across dstArea. It is the number of pixels into srcArea to start repeating from when tiling dstArea.

Return values:

dng_point containing horizontal and vertical phase.

Referenced by RepeatArea().

6.41.2.32 void dng_pixel_buffer::RepeatArea (const dng_rect & *srcArea*, const dng_rect & *dstArea*)

Repeat the image data in srcArea acros dstArea. (Generally used for padding operations.)

Parameters:

srcArea Area to repeat from.

dstArea Area to fill with data from srcArea.

References ConstPixel(), DirtyPixel(), RepeatPhase(), and ThrowNotYetImplemented().

6.41.2.33 void dng_pixel_buffer::ShiftRight (uint32 *shift*)

Apply a right shift (C++ oerpator >>) to all pixel values. Only implemented for 16-bit (signed or unsigned) pixel buffers.

Parameters:

shift Number of bits by which to right shift each pixel value.

References DirtyPixel(), and ThrowNotYetImplemented().

6.41.2.34 void dng_pixel_buffer::FlipH ()

Change metadata so pixels are iterated in opposite horizontal order. This operation does not require movement of actual pixel data.

6.41.2.35 void dng_pixel_buffer::FlipV ()

Change metadata so pixels are iterated in opposite vertical order. This operation does not require movement of actual pixel data.

6.41.2.36 void dng_pixel_buffer::FlipZ ()

Change metadata so pixels are iterated in opposite plane order. This operation does not require movement of actual pixel data.

6.41.2.37 bool dng_pixel_buffer::EqualArea (const dng_pixel_buffer & *rhs*, const dng_rect & *area*, uint32 *plane*, uint32 *planes*) const

Return true if the contents of an area of the pixel buffer area are the same as those of another.

Parameters:

rhs Buffer to compare against.

area Rectangle of pixel buffer to test.

plane Plane to start comparing.

planes Number of planes to compare.

References ConstPixel(), fColStep, fPixelType, fPlaneStep, fRowStep, and ThrowNotYetImplemented().

Referenced by dng_image::EqualArea().

The documentation for this class was generated from the following files:

- dng_pixel_buffer.h
- dng_pixel_buffer.cpp

6.42 dng_render Class Reference

Class used to render digital negative to displayable image.

`#include <dng_render.h>`

Public Member Functions

- dng_render (dng_host &host, const dng_negative &negative)
- void SetWhiteXY (const dng_xy_coord &white)
- const dng_xy_coord WhiteXY () const
- void SetExposure (real64 exposure)
- real64 Exposure () const
- void SetShadows (real64 shadows)
- real64 Shadows () const
- void SetToneCurve (const dng_1d_function &curve)
- const dng_1d_function & ToneCurve () const
- void SetFinalSpace (const dng_color_space &space)
- const dng_color_space & FinalSpace () const
- void SetFinalPixelType (uint32 type)
- uint32 FinalPixelType () const
- void SetMaximumSize (uint32 size)
- uint32 MaximumSize () const
- virtual dng_image * Render ()

Protected Attributes

- dng_host & **fHost**
- const dng_negative & **fNegative**
- dng_xy_coord **fWhiteXY**
- real64 **fExposure**
- real64 **fShadows**
- const dng_1d_function * **fToneCurve**
- const dng_color_space * **fFinalSpace**
- uint32 **fFinalPixelType**
- uint32 **fMaximumSize**

6.42.1 Detailed Description

Class used to render digital negative to displayable image.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 dng_render::dng_render (dng_host & *host*, const dng_negative & *negative*)

Construct a rendering instance that will be used to convert a given digital negative.

Parameters:

host The host to use for memory allocation, progress updates, and abort testing.

negative The digital negative to convert to a displayable image.

References dng_negative::ColorimetricReference(), AutoPtr< T >::Get(), dng_1d_identity::Get(), dng_negative::ProfileByID(), AutoPtr< T >::Reset(), and dng_camera_profile::ToneCurve().

6.42.3 Member Function Documentation

6.42.3.1 void dng_render::SetWhiteXY (const dng_xy_coord & *white*) [inline]

Set the white point to be used for conversion.

Parameters:

white White point to use.

6.42.3.2 const dng_xy_coord dng_render::WhiteXY () const [inline]

Get the white point to be used for conversion.

Return values:

White point to use.

6.42.3.3 void dng_render::SetExposure (real64 *exposure*) [inline]

Set exposure compensation.

Parameters:

exposure Compensation value in stops, positive or negative.

6.42.3.4 real64 dng_render::Exposure () const [inline]

Get exposure compensation.

Return values:

Compensation value in stops, positive or negative.

6.42.3.5 void dng_render::SetShadows (real64 *shadows*) [inline]

Set shadow clip amount.

Parameters:

shadows Shadow clip amount.

6.42.3.6 real64 dng_render::Shadows () const [inline]

Get shadow clip amount.

Return values:

Shadow clip amount.

6.42.3.7 void dng_render::SetToneCurve (const dng_1d_function & *curve*) [inline]

Set custom tone curve for conversion.

Parameters:

curve 1D function that defines tone mapping to use during conversion.

6.42.3.8 const dng_1d_function& dng_render::ToneCurve () const [inline]

Get custom tone curve for conversion.

Return values:

1D function that defines tone mapping to use during conversion.

6.42.3.9 void dng_render::SetFinalSpace (const dng_color_space & *space*) [inline]

Set final color space in which resulting image data should be represented. (See dng_color_space.h for possible values.)

Parameters:

space Color space to use.

6.42.3.10 const dng_color_space& dng_render::FinalSpace () const [inline]

Get final color space in which resulting image data should be represented.

Return values:

Color space to use.

Referenced by Render().

6.42.3.11 void dng_render::SetFinalPixelType (uint32 *type*) [inline]

Set pixel type of final image data. Can be ttByte (default), ttShort, or ttFloat.

Parameters:

type Pixel type to use.

6.42.3.12 uint32 dng_render::FinalPixelType () const [inline]

Get pixel type of final image data. Can be ttByte (default), ttShort, or ttFloat.

Return values:

Pixel type to use.

Referenced by Render().

6.42.3.13 void dng_render::SetMaximumSize (uint32 *size*) [inline]

Set maximum dimension, in pixels, of resulting image. If final image would have either dimension larger than maximum, the larger of the two dimensions is set to this maximum size and the smaller dimension is adjusted to preserve aspect ratio.

Parameters:

size Maximum size to allow.

6.42.3.14 uint32 dng_render::MaximumSize () const [inline]

Get maximum dimension, in pixels, of resulting image. If the final image would have either dimension larger than this maximum, the larger of the two dimensions is set to this maximum size and the smaller dimension is adjusted to preserve the image's aspect ratio.

Return values:

Maximum allowed size.

Referenced by Render().

6.42.3.15 dng_image * dng_render::Render () [virtual]

Actually render a digital negative to a displayable image. Input digital negative is passed to the constructor of this dng_render class.

Return values:

The final resulting image.

References dng_negative::AspectRatio(), dng_negative::DefaultCropArea(), dng_negative::DefaultFinalHeight(), dng_negative::DefaultFinalWidth(), FinalPixelType(), FinalSpace(), AutoPtr< T >::Get(), dng_color_space::IsMonochrome(), dng_host::Make_dng_image(), MaximumSize(), dng_host::PerformAreaTask(), dng_image::PixelType(), dng_image::Planes(), AutoPtr< T >::Release(), AutoPtr< T >::Reset(), and dng_negative::Stage3Image().

The documentation for this class was generated from the following files:

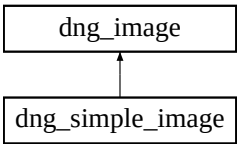
- dng_render.h
- dng_render.cpp

6.43 dng_simple_image Class Reference

dng_image derived class with simple Trim and Rotate functionality.

`#include <dng_simple_image.h>`

Inheritance diagram for dng_simple_image::



Public Member Functions

- **dng_simple_image** (const dng_rect &bounds, uint32 planes, uint32 pixelType, uint32 pixelRange, dng_memory_allocator &allocator)
- virtual void SetPixelType (uint32 pixelType)

  Setter for pixel type.

- virtual void SetPixelRange (uint32 pixelRange)

  Setter for pixel range.

- virtual void Trim (const dng_rect &r)

  Trim image data outside of given bounds. Memory is not reallocated or freed.

- virtual void Rotate (const dng_orientation &orientation)

  Rotate image according to orientation.

- void GetPixelBuffer (dng_pixel_buffer &buffer)

  Get the buffer for direct processing. (Unique to dng_simple_image.).

Protected Member Functions

- virtual void **AcquireTileBuffer** (dng_tile_buffer &buffer, const dng_rect &area, bool dirty) const

Protected Attributes

- dng_pixel_buffer **fBuffer**
- AutoPtr< dng_memory_block > **fMemory**

6.43.1 Detailed Description

dng_image derived class with simple Trim and Rotate functionality.

The documentation for this class was generated from the following files:

- dng_simple_image.h
- dng_simple_image.cpp

6.44 dng_sniffer_task Class Reference

Class to establish scope of a named subtask in DNG processing.

`#include <dng_abort_sniffer.h>`

Public Member Functions

- dng_sniffer_task (dng_abort_sniffer *sniffer, const char *name=NULL, real64 fract=0.0)
- void Sniff ()

  Check for pending user cancellation or other abort. ThrowUserCanceled will be called if one is pending.

- void UpdateProgress (real64 fract)
- void UpdateProgress (uint32 done, uint32 total)
- void Finish ()

  Signal task completed for progress purposes.

6.44.1 Detailed Description

Class to establish scope of a named subtask in DNG processing.

Instances of this class are intended to be stack allocated.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 dng_sniffer_task::dng_sniffer_task (dng_abort_sniffer * *sniffer*, const char * *name* = NULL, real64 *fract* = 0.0) [inline]

Inform a sniffer of a subtask in DNG processing.

Parameters:

sniffer The sniffer associated with the host on which this processing is occurring.

name The name of this subtask as a NUL terminated string.

fract Percentage of total processing this task is expected to take, from 0.0 to 1.0 .

References dng_abort_sniffer::StartTask().

6.44.3 Member Function Documentation

6.44.3.1 void dng_sniffer_task::UpdateProgress (real64 *fract*) [inline]

Update progress on this subtask.

Parameters:

fract Percentage of processing completed on current task, from 0.0 to 1.0 .

References dng_abort_sniffer::UpdateProgress().

Referenced by Finish(), and UpdateProgress().

6.44.3.2 void dng_sniffer_task::UpdateProgress (uint32 *done*, uint32 *total*) [inline]

Update progress on this subtask.

Parameters:

done Amount of task completed in arbitrary integer units.

total Total size of task in same arbitrary integer units as done.

References UpdateProgress().

The documentation for this class was generated from the following file:

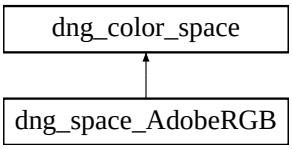
- dng_abort_sniffer.h

6.45 dng_space_AdobeRGB Class Reference

Singleton class for AdobeRGB color space.

`#include <dng_color_space.h>`

Inheritance diagram for dng_space_AdobeRGB::



Public Member Functions

- virtual const dng_1d_function & GammaFunction () const

  Returns dng_function_GammaEncode_1_8.

- virtual bool ICCProfile (uint32 &size, const uint8 *&data) const

  Returns AdobeRGB (1998) ICC profile.

Static Public Member Functions

- static const dng_color_space & Get ()

  Static method for getting single global instance of this color space.

6.45.1 Detailed Description

Singleton class for AdobeRGB color space.

The documentation for this class was generated from the following files:

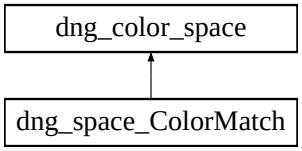
- dng_color_space.h
- dng_color_space.cpp

6.46 dng_space_ColorMatch Class Reference

Singleton class for ColorMatch color space.

`#include <dng_color_space.h>`

Inheritance diagram for dng_space_ColorMatch::



Public Member Functions

- virtual const dng_1d_function & GammaFunction () const

  Returns dng_function_GammaEncode_1_8.

- virtual bool ICCProfile (uint32 &size, const uint8 *&data) const

  Returns ColorMatch RGB ICC profile.

Static Public Member Functions

- static const dng_color_space & Get ()

  Static method for getting single global instance of this color space.

6.46.1 Detailed Description

Singleton class for ColorMatch color space.

The documentation for this class was generated from the following files:

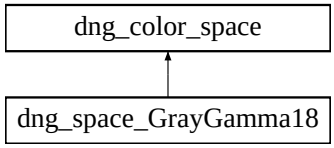
- dng_color_space.h
- dng_color_space.cpp

6.47 dng_space_GrayGamma18 Class Reference

Singleton class for gamma 1.8 grayscale color space.

`#include <dng_color_space.h>`

Inheritance diagram for dng_space_GrayGamma18::



Public Member Functions

- virtual const dng_1d_function & GammaFunction () const

  Returns dng_function_GammaEncode_1_8.

- virtual bool ICCProfile (uint32 &size, const uint8 *&data) const

  Returns simple grayscale gamma 1.8 ICC profile.

Static Public Member Functions

- static const dng_color_space & Get ()

  Static method for getting single global instance of this color space.

6.47.1 Detailed Description

Singleton class for gamma 1.8 grayscale color space.

The documentation for this class was generated from the following files:

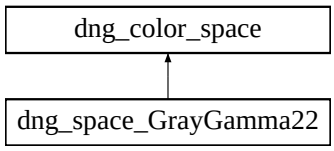
- dng_color_space.h
- dng_color_space.cpp

6.48 dng_space_GrayGamma22 Class Reference

Singleton class for gamma 2.2 grayscale color space.

`#include <dng_color_space.h>`

Inheritance diagram for dng_space_GrayGamma22::



Public Member Functions

- virtual const dng_1d_function & GammaFunction () const

  Returns dng_function_GammaEncode_2_2.

- virtual bool ICCProfile (uint32 &size, const uint8 *&data) const

  Returns simple grayscale gamma 2.2 ICC profile.

Static Public Member Functions

- static const dng_color_space & Get ()

  Static method for getting single global instance of this color space.

6.48.1 Detailed Description

Singleton class for gamma 2.2 grayscale color space.

The documentation for this class was generated from the following files:

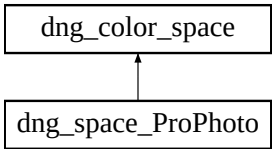
- dng_color_space.h
- dng_color_space.cpp

6.49 dng_space_ProPhoto Class Reference

Singleton class for ProPhoto RGB color space.

`#include <dng_color_space.h>`

Inheritance diagram for dng_space_ProPhoto::



Public Member Functions

- virtual const dng_1d_function & GammaFunction () const
  Returns dng_function_GammaEncode_1_8.
- virtual bool ICCProfile (uint32 &size, const uint8 *&data) const
  Returns ProPhoto RGB ICC profile.

Static Public Member Functions

- static const dng_color_space & Get ()
  Static method for getting single global instance of this color space.

6.49.1 Detailed Description

Singleton class for ProPhoto RGB color space.

The documentation for this class was generated from the following files:

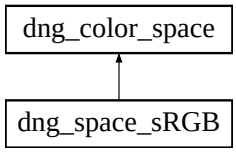
- dng_color_space.h
- dng_color_space.cpp

6.50 dng_space_sRGB Class Reference

Singleton class for sRGB color space.

`#include <dng_color_space.h>`

Inheritance diagram for dng_space_sRGB::



Public Member Functions

- virtual const dng_1d_function & GammaFunction () const
  Returns dng_function_GammaEncode_sRGB.
- virtual bool ICCProfile (uint32 &size, const uint8 *&data) const
  Returns sRGB IEC61966-2.1 ICC profile.

Static Public Member Functions

- static const dng_color_space & Get ()
  Static method for getting single global instance of this color space.

6.50.1 Detailed Description

Singleton class for sRGB color space.

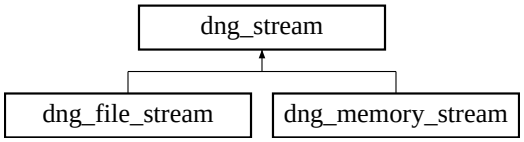
The documentation for this class was generated from the following files:

- dng_color_space.h
- dng_color_space.cpp

6.51 dng_stream Class Reference

`#include <dng_stream.h>`

Inheritance diagram for dng_stream::



Public Types

- enum { **kSmallBufferSize** = 4 ∗ 1024, **kBigBufferSize** = 64 ∗ 1024, **kDefaultBufferSize** = kSmallBufferSize }

Public Member Functions

- dng_stream (const void ∗data, uint32 count, uint64 offsetInOriginalFile=kDNGStreamInvalidOffset)
- bool SwapBytes () const
- void SetSwapBytes (bool swapBytes)
- bool BigEndian () const
- void SetBigEndian (bool bigEndian=true)
- bool LittleEndian () const
- void SetLittleEndian (bool littleEndian=true)
- uint32 BufferSize () const

  Returns the size of the buffer used by the stream.

- uint64 Length ()
- uint64 Position () const
- uint64 PositionInOriginalFile () const
- uint64 OffsetInOriginalFile () const
- const void ∗ Data () const
- dng_memory_block ∗ AsMemoryBlock (dng_memory_allocator &allocator)
- void SetReadPosition (uint64 offset)

  Seek to a new position in stream for reading.

- void Skip (uint64 delta)
- void Get (void ∗data, uint32 count)
- void SetWritePosition (uint64 offset)

  Seek to a new position in stream for writing.

- void Flush ()

  Force any stored data in stream to be written to underlying storage.

- void SetLength (uint64 length)
- void Put (const void ∗data, uint32 count)
- uint8 Get_uint8 ()
- void Put_uint8 (uint8 x)
- uint16 Get_uint16 ()
- void Put_uint16 (uint16 x)
- uint32 Get_uint32 ()
- void Put_uint32 (uint32 x)

- uint64 Get_uint64 ()
- void Put_uint64 (uint64 x)
- int8 Get_int8 ()
- void Put_int8 (int8 x)
- int16 Get_int16 ()
- void Put_int16 (int16 x)
- int32 Get_int32 ()
- void Put_int32 (int32 x)
- int64 Get_int64 ()
- void Put_int64 (int64 x)
- real32 Get_real32 ()
- void Put_real32 (real32 x)
- real64 Get_real64 ()
- void Put_real64 (real64 x)
- void Get_CString (char *data, uint32 maxLength)
- void Get_UString (char *data, uint32 maxLength)
- void PutZeros (uint64 count)
- void PadAlign2 ()

  Writes zeros to align the stream position to a multiple of 2.

- void PadAlign4 ()

  Writes zeros to align the stream position to a multiple of 4.

- uint32 TagValue_uint32 (uint32 tagType)
- int32 TagValue_int32 (uint32 tagType)
- dng_urational TagValue_urational (uint32 tagType)
- dng_srational TagValue_srational (uint32 tagType)
- real64 TagValue_real64 (uint32 tagType)
- dng_abort_sniffer * Sniffer () const
- void SetSniffer (dng_abort_sniffer *sniffer)
- virtual void CopyToStream (dng_stream &dstStream, uint64 count)
- void DuplicateStream (dng_stream &dstStream)

Protected Member Functions

- **dng_stream** (dng_abort_sniffer *sniffer=NULL, uint32 bufferSize=kDefaultBufferSize, uint64 offsetInOriginalFile=kDNGStreamInvalidOffset)
- virtual uint64 **DoGetLength** ()
- virtual void **DoRead** (void *data, uint32 count, uint64 offset)
- virtual void **DoSetLength** (uint64 length)
- virtual void **DoWrite** (const void *data, uint32 count, uint64 offset)

6.51.1 Detailed Description

Base stream abstraction. Has support for going between stream and pointer abstraction.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 dng_stream::dng_stream (const void * *data*, uint32 *count*, uint64 *offsetInOriginalFile* = kDNGStreamInvalidOffset)

Construct a stream with initial data.

Parameters:

data Pointer to initial contents of stream.

count Number of bytes data is valid for.

offsetInOriginalFile If data came from a file originally, offset can be saved here for later use.

6.51.3 Member Function Documentation

6.51.3.1 bool dng_stream::SwapBytes () const [inline]

Getter for whether stream is swapping byte order on input/output.

Return values:

If true, data will be swapped on input/output.

6.51.3.2 void dng_stream::SetSwapBytes (bool *swapBytes*) [inline]

Setter for whether stream is swapping byte order on input/output.

Parameters:

swapBytes If true, stream will swap byte order on input or output for future reads/writes.

6.51.3.3 bool dng_stream::BigEndian () const

Getter for whether data in stream is big endian.

Return values:

If true, data in stream is big endian.

Referenced by LittleEndian(), dng_image_writer::WriteDNG(), and dng_image_writer::WriteTIFF().

6.51.3.4 void dng_stream::SetBigEndian (bool *bigEndian* = `true`)

Setter for whether data in stream is big endian.

Parameters:

bigEndian If true, data in stream is big endian.

Referenced by dng_iptc::Parse(), dng_info::Parse(), SetLittleEndian(), and dng_iptc::Spool().

6.51.3.5 bool dng_stream::LittleEndian () const `[inline]`

Getter for whether data in stream is big endian.

Return values:

If true, data in stream is big endian.

References BigEndian().

6.51.3.6 void dng_stream::SetLittleEndian (bool *littleEndian* = `true`) `[inline]`

Setter for whether data in stream is big endian.

Parameters:

littleEndian If true, data in stream is big endian.

References SetBigEndian().

Referenced by dng_info::Parse().

6.51.3.7 uint64 dng_stream::Length () `[inline]`

Getter for length of data in stream.

Return values:

Length of readable data in stream.

Referenced by AsMemoryBlock(), dng_memory_stream::CopyToStream(), DuplicateStream(), Get(), dng_iptc::Parse(), dng_info::Parse(), Put(), Put_uint8(), SetLength(), and SetReadPosition().

6.51.3.8 uint64 dng_stream::Position () const [inline]

Getter for current offset in stream.

Return values:

current offset from start of stream.

Referenced by dng_memory_stream::CopyToStream(), PadAlign2(), PadAlign4(), dng_iptc::Parse(), dng_info::Parse(), PositionInOriginalFile(), Skip(), dng_image_writer::WriteDNG(), and dng_image_writer::WriteTIFF().

6.51.3.9 uint64 dng_stream::PositionInOriginalFile () const

Getter for current position in original file, taking into account OffsetInOriginalFile stream data was taken from.

Return values:

kInvalidOffset if no offset in original file is set, sum of offset in original file and current position otherwise.

References Position().

Referenced by dng_info::Parse().

6.51.3.10 uint64 dng_stream::OffsetInOriginalFile () const

Getter for offset in original file.

Return values:

kInvalidOffset if no offset in original file is set, offset in original file otherwise.

6.51.3.11 const void * dng_stream::Data () const

Return pointer to stream contents if the stream is entirely available as a single memory block, NULL otherwise.

6.51.3.12 dng_memory_block * dng_stream::AsMemoryBlock (dng_memory_allocator & *allocator*)

Return the entire stream as a single memory block. This works for all streams, but requires copying the data to a new buffer.

Parameters:

allocator Allocator used to allocate memory.

References dng_memory_allocator::Allocate(), Flush(), Get(), Length(), SetReadPosition(), and ThrowProgramError().

Referenced by dng_iptc::Spool().

6.51.3.13 void dng_stream::Skip (uint64 *delta*) [inline]

Skip forward in stream.

Parameters:

delta Number of bytes to skip forward.

References Position(), and SetReadPosition().

6.51.3.14 void dng_stream::Get (void * *data*, uint32 *count*)

Get data from stream. Exception is thrown and no data is read if insufficient data available in stream.

Parameters:

data Buffer to put data into. Must be valid for count bytes.

count Bytes of data to read.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Flush(), Length(), dng_abort_sniffer::SniffForAbort(), and ThrowEndOfFile().

Referenced by AsMemoryBlock(), CopyToStream(), Get_real64(), Get_uint16(), Get_uint32(), Get_uint64(), Get_uint8(), and dng_iptc::Parse().

6.51.3.15 void dng_stream::SetLength (uint64 *length*)

Set length of available data.

Parameters:

length Number of bytes of avialble data in stream.

References Flush(), and Length().

Referenced by DuplicateStream(), dng_image_writer::WriteDNG(), and dng_image_writer::WriteTIFF().

6.51.3.16 void dng_stream::Put (const void * *data*, uint32 *count*)

Write data to stream.

Parameters:

data Buffer of data to write to stream.

count Bytes of in data.

References Flush(), Length(), and dng_abort_sniffer::SniffForAbort().

Referenced by CopyToStream(), dng_memory_stream::CopyToStream(), Put_real32(), Put_real64(), Put_uint16(), Put_uint32(), Put_uint64(), Put_uint8(), PutZeros(), and dng_iptc::Spool().

6.51.3.17 uint8 dng_stream::Get_uint8 () [inline]

Get an unsigned 8-bit integer from stream and advance read position.

Return values:

One unsigned 8-bit integer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get().

Referenced by Get_CString(), Get_int8(), dng_iptc::Parse(), and TagValue_uint32().

6.51.3.18 void dng_stream::Put_uint8 (uint8 *x*) [inline]

Put an unsigned 8-bit integer to stream and advance write position.

Parameters:

x One unsigned 8-bit integer.

References Length(), and Put().

Referenced by Put_int8(), PutZeros(), and dng_iptc::Spool().

6.51.3.19 uint16 dng_stream::Get_uint16 ()

Get an unsigned 16-bit integer from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One unsigned 16-bit integer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get().

Referenced by Get_int16(), Get_UString(), dng_iptc::Parse(), dng_info::Parse(), and TagValue_uint32().

6.51.3.20 void dng_stream::Put_uint16 (uint16 *x*)

Put an unsigned 16-bit integer to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One unsigned 16-bit integer.

References Put().

Referenced by Put_int16(), dng_iptc::Spool(), dng_image_writer::WriteDNG(), and dng_image_writer::WriteTIFF().

6.51.3.21 uint32 dng_stream::Get_uint32 ()

Get an unsigned 32-bit integer from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One unsigned 32-bit integer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get().

Referenced by Get_int32(), Get_real32(), Get_real64(), Get_uint64(), dng_info::Parse(), TagValue_real64(), TagValue_uint32(), and TagValue_urational().

6.51.3.22 void dng_stream::Put_uint32 (uint32 *x*)

Put an unsigned 32-bit integer to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One unsigned 32-bit integer.

References Put().

Referenced by Put_int32(), Put_real32(), Put_real64(), Put_uint64(), dng_image_writer::WriteDNG(), and dng_image_writer::WriteTIFF().

6.51.3.23 uint64 dng_stream::Get_uint64 ()

Get an unsigned 64-bit integer from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One unsigned 64-bit integer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get(), and Get_uint32().

Referenced by Get_int64().

6.51.3.24 void dng_stream::Put_uint64 (uint64 *x*)

Put an unsigned 64-bit integer to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One unsigned 64-bit integer.

References Put(), and Put_uint32().

Referenced by Put_int64().

6.51.3.25 int8 dng_stream::Get_int8 () `[inline]`

Get one 8-bit integer from stream and advance read position.

Return values:

One 8-bit integer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get_uint8().

Referenced by dng_iptc::Parse(), and TagValue_int32().

6.51.3.26 void dng_stream::Put_int8 (int8 *x*) [inline]

Put one 8-bit integer to stream and advance write position.

Parameters:

x One 8-bit integer.

References Put_uint8().

6.51.3.27 int16 dng_stream::Get_int16 () [inline]

Get one 16-bit integer from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One 16-bit integer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get_uint16().

Referenced by TagValue_int32().

6.51.3.28 void dng_stream::Put_int16 (int16 *x*) [inline]

Put one 16-bit integer to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One 16-bit integer.

References Put_uint16().

6.51.3.29 int32 dng_stream::Get_int32 () [inline]

Get one 32-bit integer from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One 32-bit integer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get_uint32().

Referenced by TagValue_int32(), TagValue_real64(), TagValue_srational(), and TagValue_urational().

6.51.3.30 void dng_stream::Put_int32 (int32 *x*) [inline]

Put one 32-bit integer to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One 32-bit integer.

References Put_uint32().

6.51.3.31 int64 dng_stream::Get_int64 () [inline]

Get one 64-bit integer from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One 64-bit integer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get_uint64().

6.51.3.32 void dng_stream::Put_int64 (int64 *x*) [inline]

Put one 64-bit integer to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One 64-bit integer.

References Put_uint64().

6.51.3.33 real32 dng_stream::Get_real32 ()

Get one 32-bit IEEE floating-point number from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One 32-bit IEEE floating-point number.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get_uint32().

Referenced by dng_camera_profile::Parse(), and TagValue_real64().

6.51.3.34 void dng_stream::Put_real32 (real32 *x*)

Put one 32-bit IEEE floating-point number to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One 32-bit IEEE floating-point number.

References Put(), and Put_uint32().

6.51.3.35 real64 dng_stream::Get_real64 ()

Get one 64-bit IEEE floating-point number from stream and advance read position. Byte swap if byte swapping is turned on.

Return values:

One 64-bit IEEE floating-point number .

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get(), and Get_uint32().

Referenced by TagValue_real64().

6.51.3.36 void dng_stream::Put_real64 (real64 *x*)

Put one 64-bit IEEE floating-point number to stream and advance write position. Byte swap if byte swapping is turned on.

Parameters:

x One64-bit IEEE floating-point number.

References Put(), and Put_uint32().

6.51.3.37 void dng_stream::Get_CString (char * *data*, uint32 *maxLength*)

Get an 8-bit character string from stream and advance read position. Routine always reads until a NUL character (8-bits of zero) is read. (That is, only maxLength bytes will be returned in buffer, but the stream is always advanced until a NUL is read or EOF is reached.)

Parameters:

data Buffer in which string is returned.

maxLength Maximum number of bytes to place in buffer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if stream runs out before NUL is seen.

References Get_uint8().

6.51.3.38 void dng_stream::Get_UString (char * *data*, uint32 *maxLength*)

Get a 16-bit character string from stream and advance read position. 16-bit characters are truncated to 8-bits. Routine always reads until a NUL character (16-bits of zero) is read. (That is, only maxLength bytes will be returned in buffer, but the stream is always advanced until a NUL is read or EOF is reached.)

Parameters:

data Buffer to place string in.

maxLength Maximum number of bytes to place in buffer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if stream runs out before NUL is seen.

References Get_uint16().

6.51.3.39 void dng_stream::PutZeros (uint64 *count*)

Writes the specified number of zero bytes to stream.

Parameters:

count Number of zero bytes to write.

References dng_memory_data::Buffer(), Put(), and Put_uint8().

Referenced by PadAlign2(), and PadAlign4().

6.51.3.40 uint32 dng_stream::TagValue_uint32 (uint32 *tagType*)

Get a value of size indicated by tag type from stream and advance read position. Byte swap if byte swapping is turned on and tag type is larger than a byte. Value is returned as an unsigned 32-bit integer.

Parameters:

tagType Tag type of data stored in stream.

Return values:

One unsigned 32-bit integer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get_uint16(), Get_uint32(), Get_uint8(), and TagValue_real64().

Referenced by TagValue_real64(), and TagValue_urational().

6.51.3.41 int32 dng_stream::TagValue_int32 (uint32 *tagType*)

Get a value of size indicated by tag type from stream and advance read position. Byte swap if byte swapping is turned on and tag type is larger than a byte. Value is returned as a 32-bit integer.

Parameters:

tagType Tag type of data stored in stream.

Return values:

One 32-bit integer.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get_int16(), Get_int32(), Get_int8(), and TagValue_real64().

Referenced by TagValue_real64(), and TagValue_urational().

6.51.3.42 dng_urational dng_stream::TagValue_urational (uint32 *tagType*)

Get a value of size indicated by tag type from stream and advance read position. Byte swap if byte swapping is turned on and tag type is larger than a byte. Value is returned as a dng_urational.

Parameters:

tagType Tag type of data stored in stream.

Return values:

One dng_urational.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get_int32(), Get_uint32(), TagValue_int32(), TagValue_real64(), and TagValue_uint32().

6.51.3.43 dng_srational dng_stream::TagValue_srational (uint32 *tagType*)

Get a value of size indicated by tag type from stream and advance read position. Byte swap if byte swapping is turned on and tag type is larger than a byte. Value is returned as a dng_srational.

Parameters:

tagType Tag type of data stored in stream.

Return values:

One dng_srational.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get_int32(), and TagValue_real64().

6.51.3.44 real64 dng_stream::TagValue_real64 (uint32 *tagType*)

Get a value of size indicated by tag type from stream and advance read position. Byte swap if byte swapping is turned on and tag type is larger than a byte. Value is returned as a 64-bit IEEE floating-point number.

Parameters:

tagType Tag type of data stored in stream.

Return values:

One 64-bit IEEE floating-point number.

Exceptions:

dng_exception with fErrorCode equal to dng_error_end_of_file if not enough data in stream.

References Get_int32(), Get_real32(), Get_real64(), Get_uint32(), TagValue_int32(), and TagValue_uint32().

Referenced by TagValue_int32(), TagValue_srational(), TagValue_uint32(), and TagValue_urational().

6.51.3.45 dng_abort_sniffer* dng_stream::Sniffer () const [inline]

Getter for sniffer associated with stream.

Return values:

The sniffer for this stream.

6.51.3.46 void dng_stream::SetSniffer (dng_abort_sniffer * *sniffer*) [inline]

Putter for sniffer associated with stream.

Parameters:

sniffer The new sniffer to use (or NULL for none).

6.51.3.47 void dng_stream::CopyToStream (dng_stream & *dstStream*, uint64 *count*) [virtual]

Copy a specified number of bytes to a target stream.

Parameters:

dstStream The target stream.

count The number of bytes to copy.

Reimplemented in dng_memory_stream.

References dng_memory_data::Buffer(), Get(), and Put().

Referenced by dng_memory_stream::CopyToStream(), and DuplicateStream().

6.51.3.48 void dng_stream::DuplicateStream (dng_stream & *dstStream*)

Makes the target stream a copy of this stream.

Parameters:

dstStream The target stream.

References CopyToStream(), Flush(), Length(), SetLength(), SetReadPosition(), and SetWritePosition().

The documentation for this class was generated from the following files:

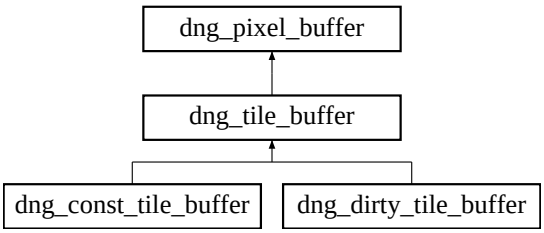
- dng_stream.h
- dng_stream.cpp

6.52 dng_tile_buffer Class Reference

Class to get resource acquisition is instantiation behavior for tile buffers. Can be dirty or constant tile access.

`#include <dng_image.h>`

Inheritance diagram for dng_tile_buffer::



Public Member Functions

- void **SetRefData** (void *refData)
- void * **GetRefData** () const

Protected Member Functions

- dng_tile_buffer (const dng_image &image, const dng_rect &tile, bool dirty)

Protected Attributes

- const dng_image & **fImage**
- void * **fRefData**

6.52.1 Detailed Description

Class to get resource acquisition is instantiation behavior for tile buffers. Can be dirty or constant tile access.

6.52.2 Constructor & Destructor Documentation

6.52.2.1 dng_tile_buffer::dng_tile_buffer (const dng_image & *image*, const dng_rect & *tile*, bool *dirty*) [protected]

Obtain a tile from an image.

Parameters:

image Image tile will come from.

tile Rectangle denoting extent of tile.

dirty Flag indicating whether this is read-only or read-write acesss.

References dng_image::AcquireTileBuffer().

The documentation for this class was generated from the following files:

- dng_image.h
- dng_image.cpp

6.53 dng_time_zone Class Reference

Class for holding a time zone.

`#include <dng_date_time.h>`

Public Member Functions

- void **Clear** ()
- void **SetOffsetHours** (int32 offset)
- void **SetOffsetMinutes** (int32 offset)
- void **SetOffsetSeconds** (int32 offset)
- bool **IsValid** () const
- bool **NotValid** () const
- int32 **OffsetMinutes** () const
- bool **IsExactHourOffset** () const
- int32 **ExactHourOffset** () const
- dng_string **Encode_ISO_8601** () const

6.53.1 Detailed Description

Class for holding a time zone.

The documentation for this class was generated from the following files:

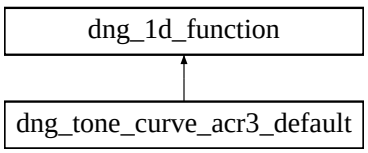
- dng_date_time.h
- dng_date_time.cpp

6.54 dng_tone_curve_acr3_default Class Reference

Default ACR3 tone curve.

`#include <dng_render.h>`

Inheritance diagram for dng_tone_curve_acr3_default::



Public Member Functions

- virtual real64 Evaluate (real64 x) const

  Returns output value for a given input tone.

- virtual real64 EvaluateInverse (real64 x) const

  Returns nearest input value for a given output tone.

Static Public Member Functions

- static const dng_1d_function & **Get** ()

6.54.1 Detailed Description

Default ACR3 tone curve.

The documentation for this class was generated from the following files:

- dng_render.h
- dng_render.cpp

7 File Documentation

7.1 dng_1d_function.h File Reference

Classes

- class dng_1d_function

  A 1D floating-point function.

- class dng_1d_identity

  An identity (x -> y such that x == y for all x) mapping function.

- class dng_1d_concatenate

  A dng_1d_function that represents the composition (curry) of two other dng_1d_-functions.

- class dng_1d_inverse

  A dng_1d_function that represents the inverse of another dng_1d_function.

7.1.1 Detailed Description

Classes for a 1D floating-point to floating-point function abstraction.

7.2 dng_1d_table.h File Reference

Classes

- class dng_1d_table

  A 1D floating-point lookup table using linear interpolation.

7.2.1 Detailed Description

Definition of a lookup table based 1D floating-point to floating-point function abstraction using linear interpolation.

7.3 dng_abort_sniffer.h File Reference

Classes

- class dng_abort_sniffer

  Class for signaling user cancellation and receiving progress updates.

- class dng_sniffer_task

  Class to establish scope of a named subtask in DNG processing.

7.3.1 Detailed Description

Classes supporting user cancellation and progress tracking.

7.4 dng_area_task.h File Reference

Classes

- class dng_area_task

  Abstract class for rectangular processing operations with support for partitioning across multiple processing resources and observing memory constraints.

7.4.1 Detailed Description

Class to handle partitioning a rectangular image processing operation taking into account multiple processing resources and memory constraints.

7.5 dng_assertions.h File Reference

Defines

- #define DNG_ASSERT(x, y)
- #define **DNG_REPORT**(x) DNG_ASSERT (false, x)

7.5.1 Detailed Description

Conditionally compiled assertion check support.

7.5.2 Define Documentation

7.5.2.1 #define DNG_ASSERT(x, y)

Conditionally compiled macro to check an assertion and display a message if it fails and assertions are compiled in via qDNGDebug

Parameters:

x Predicate which must be true.

y String to display if x is not true.

Referenced by dng_negative::AnalogBalance(), dng_color_spec::CameraToPCS(), dng_color_spec::CameraWhite(), dng_negative::DefaultCropArea(), dng_pixel_buffer::DirtyPixel(), dng_1d_table::Interpolate(), dng_pixel_buffer::SetConstant_int16(), dng_pixel_buffer::SetConstant_real32(), dng_pixel_buffer::SetConstant_uint16(), dng_pixel_buffer::SetConstant_uint32(), dng_pixel_buffer::SetConstant_uint8(), dng_iptc::Spool(), and dng_color_spec::WhiteXY().

7.6 dng_auto_ptr.h File Reference

Classes

- class AutoPtr< T >

  A class intended to be used in stack scope to hold a pointer from new. The held pointer will be deleted automatically if the scope is left without calling Release on the AutoPtr first.

7.6.1 Detailed Description

Class to implement std::auto_ptr like functionality even on platforms which do not have a full Standard C++ library.

7.7 dng_bottlenecks.h File Reference

Classes

- struct **dng_suite**

Typedefs

- typedef void(**ZeroBytesProc**)(void *dPtr, uint32 count)
- typedef void(**CopyBytesProc**)(const void *sPtr, void *dPtr, uint32 count)
- typedef void(**SwapBytes16Proc**)(uint16 *dPtr, uint32 count)
- typedef void(**SwapBytes32Proc**)(uint32 *dPtr, uint32 count)
- typedef void(**SetArea8Proc**)(uint8 *dPtr, uint8 value, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep)
- typedef void(**SetArea16Proc**)(uint16 *dPtr, uint16 value, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep)
- typedef void(**SetArea32Proc**)(uint32 *dPtr, uint32 value, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep)
- typedef void(**CopyArea8Proc**)(const uint8 *sPtr, uint8 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef void(**CopyArea16Proc**)(const uint16 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef void(**CopyArea32Proc**)(const uint32 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef void(**CopyArea8_16Proc**)(const uint8 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef void(**CopyArea8_S16Proc**)(const uint8 *sPtr, int16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef void(**CopyArea8_32Proc**)(const uint8 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef void(**CopyArea16_S16Proc**)(const uint16 *sPtr, int16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)

- typedef void(**CopyArea16_32Proc**)(const uint16 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef void(**CopyArea8_R32Proc**)(const uint8 *sPtr, real32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- typedef void(**CopyArea16_R32Proc**)(const uint16 *sPtr, real32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- typedef void(**CopyAreaS16_R32Proc**)(const int16 *sPtr, real32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- typedef void(**CopyAreaR32_8Proc**)(const real32 *sPtr, uint8 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- typedef void(**CopyAreaR32_16Proc**)(const real32 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- typedef void(**CopyAreaR32_S16Proc**)(const real32 *sPtr, int16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- typedef void(**RepeatArea8Proc**)(const uint8 *sPtr, uint8 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 repeatV, uint32 repeatH, uint32 phaseV, uint32 phaseH)
- typedef void(**RepeatArea16Proc**)(const uint16 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 repeatV, uint32 repeatH, uint32 phaseV, uint32 phaseH)
- typedef void(**RepeatArea32Proc**)(const uint32 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 repeatV, uint32 repeatH, uint32 phaseV, uint32 phaseH)
- typedef void(**ShiftRight16Proc**)(uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 shift)
- typedef void(**BilinearRow16Proc**)(const uint16 *sPtr, uint16 *dPtr, uint32 cols, uint32 patPhase, uint32 patCount, const uint32 *kernCounts, const int32 *const *kernOffsets, const uint16 *const *kernWeights, uint32 sShift)
- typedef void(**BilinearRow32Proc**)(const real32 *sPtr, real32 *dPtr, uint32 cols, uint32 patPhase, uint32 patCount, const uint32 *kernCounts, const int32 *const *kernOffsets, const real32 *const *kernWeights, uint32 sShift)

- typedef void(**BaselineABCtoRGBProc**)(const real32 *sPtrA, const real32 *sPtrB, const real32 *sPtrC, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_vector &cameraWhite, const dng_matrix &cameraToRGB)
- typedef void(**BaselineABCDtoRGBProc**)(const real32 *sPtrA, const real32 *sPtrB, const real32 *sPtrC, const real32 *sPtrD, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_vector &cameraWhite, const dng_matrix &cameraToRGB)
- typedef void(**BaselineHueSatMapProc**)(const real32 *sPtrR, const real32 *sPtrG, const real32 *sPtrB, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_hue_sat_map &lut)
- typedef void(**BaselineGrayToRGBProc**)(const real32 *sPtrR, const real32 *sPtrG, const real32 *sPtrB, real32 *dPtrG, uint32 count, const dng_matrix &matrix)
- typedef void(**BaselineRGBtoRGBProc**)(const real32 *sPtrR, const real32 *sPtrG, const real32 *sPtrB, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_matrix &matrix)
- typedef void(**Baseline1DTableProc**)(const real32 *sPtr, real32 *dPtr, uint32 count, const dng_1d_table &table)
- typedef void(**BaselineRGBToneProc**)(const real32 *sPtrR, const real32 *sPtrG, const real32 *sPtrB, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_1d_table &table)
- typedef void(**ResampleDown16Proc**)(const uint16 *sPtr, uint16 *dPtr, uint32 sCount, int32 sRowStep, const int16 *wPtr, uint32 wCount, uint32 pixelRange)
- typedef void(**ResampleDown32Proc**)(const real32 *sPtr, real32 *dPtr, uint32 sCount, int32 sRowStep, const real32 *wPtr, uint32 wCount)
- typedef void(**ResampleAcross16Proc**)(const uint16 *sPtr, uint16 *dPtr, uint32 dCount, const int32 *coord, const int16 *wPtr, uint32 wCount, uint32 wStep, uint32 pixelRange)
- typedef void(**ResampleAcross32Proc**)(const real32 *sPtr, real32 *dPtr, uint32 dCount, const int32 *coord, const real32 *wPtr, uint32 wCount, uint32 wStep)
- typedef bool(**EqualBytesProc**)(const void *sPtr, const void *dPtr, uint32 count)
- typedef bool(**EqualArea8Proc**)(const uint8 *sPtr, const uint8 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef bool(**EqualArea16Proc**)(const uint16 *sPtr, const uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- typedef bool(**EqualArea32Proc**)(const uint32 *sPtr, const uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)

Functions

- void **DoZeroBytes** (void *dPtr, uint32 count)

- void **DoCopyBytes** (const void *sPtr, void *dPtr, uint32 count)
- void **DoSwapBytes16** (uint16 *dPtr, uint32 count)
- void **DoSwapBytes32** (uint32 *dPtr, uint32 count)
- void **DoSetArea8** (uint8 *dPtr, uint8 value, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep)
- void **DoSetArea16** (uint16 *dPtr, uint16 value, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep)
- void **DoSetArea32** (uint32 *dPtr, uint32 value, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep)
- void **DoCopyArea8** (const uint8 *sPtr, uint8 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea16** (const uint16 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea32** (const uint32 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea8_16** (const uint8 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea8_S16** (const uint8 *sPtr, int16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea8_32** (const uint8 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea16_S16** (const uint16 *sPtr, int16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea16_32** (const uint16 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- void **DoCopyArea8_R32** (const uint8 *sPtr, real32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- void **DoCopyArea16_R32** (const uint16 *sPtr, real32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- void **DoCopyAreaS16_R32** (const int16 *sPtr, real32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- void **DoCopyAreaR32_8** (const real32 *sPtr, uint8 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)

- void **DoCopyAreaR32_16** (const real32 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- void **DoCopyAreaR32_S16** (const real32 *sPtr, int16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep, uint32 pixelRange)
- void **DoRepeatArea8** (const uint8 *sPtr, uint8 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 repeatV, uint32 repeatH, uint32 phaseV, uint32 phaseH)
- void **DoRepeatArea16** (const uint16 *sPtr, uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 repeatV, uint32 repeatH, uint32 phaseV, uint32 phaseH)
- void **DoRepeatArea32** (const uint32 *sPtr, uint32 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 repeatV, uint32 repeatH, uint32 phaseV, uint32 phaseH)
- void **DoShiftRight16** (uint16 *dPtr, uint32 rows, uint32 cols, uint32 planes, int32 rowStep, int32 colStep, int32 planeStep, uint32 shift)
- void **DoBilinearRow16** (const uint16 *sPtr, uint16 *dPtr, uint32 cols, uint32 patPhase, uint32 patCount, const uint32 *kernCounts, const int32 *const *kernOffsets, const uint16 *const *kernWeights, uint32 sShift)
- void **DoBilinearRow32** (const real32 *sPtr, real32 *dPtr, uint32 cols, uint32 patPhase, uint32 patCount, const uint32 *kernCounts, const int32 *const *kernOffsets, const real32 *const *kernWeights, uint32 sShift)
- void **DoBaselineABCtoRGB** (const real32 *sPtrA, const real32 *sPtrB, const real32 *sPtrC, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_vector &cameraWhite, const dng_matrix &cameraToRGB)
- void **DoBaselineABCDtoRGB** (const real32 *sPtrA, const real32 *sPtrB, const real32 *sPtrC, const real32 *sPtrD, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_vector &cameraWhite, const dng_matrix &cameraToRGB)
- void **DoBaselineHueSatMap** (const real32 *sPtrR, const real32 *sPtrG, const real32 *sPtrB, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_hue_sat_map &lut)
- void **DoBaselineRGBtoGray** (const real32 *sPtrR, const real32 *sPtrG, const real32 *sPtrB, real32 *dPtrG, uint32 count, const dng_matrix &matrix)
- void **DoBaselineRGBtoRGB** (const real32 *sPtrR, const real32 *sPtrG, const real32 *sPtrB, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_matrix &matrix)
- void **DoBaseline1DTable** (const real32 *sPtr, real32 *dPtr, uint32 count, const dng_1d_table &table)
- void **DoBaselineRGBTone** (const real32 *sPtrR, const real32 *sPtrG, const real32 *sPtrB, real32 *dPtrR, real32 *dPtrG, real32 *dPtrB, uint32 count, const dng_1d_table &table)
- void **DoResampleDown16** (const uint16 *sPtr, uint16 *dPtr, uint32 sCount, int32 sRowStep, const int16 *wPtr, uint32 wCount, uint32 pixelRange)

- void **DoResampleDown32** (const real32 ∗sPtr, real32 ∗dPtr, uint32 sCount, int32 sRowStep, const real32 ∗wPtr, uint32 wCount)
- void **DoResampleAcross16** (const uint16 ∗sPtr, uint16 ∗dPtr, uint32 dCount, const int32 ∗coord, const int16 ∗wPtr, uint32 wCount, uint32 wStep, uint32 pixelRange)
- void **DoResampleAcross32** (const real32 ∗sPtr, real32 ∗dPtr, uint32 dCount, const int32 ∗coord, const real32 ∗wPtr, uint32 wCount, uint32 wStep)
- bool **DoEqualBytes** (const void ∗sPtr, const void ∗dPtr, uint32 count)
- bool **DoEqualArea8** (const uint8 ∗sPtr, const uint8 ∗dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- bool **DoEqualArea16** (const uint16 ∗sPtr, const uint16 ∗dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)
- bool **DoEqualArea32** (const uint32 ∗sPtr, const uint32 ∗dPtr, uint32 rows, uint32 cols, uint32 planes, int32 sRowStep, int32 sColStep, int32 sPlaneStep, int32 dRowStep, int32 dColStep, int32 dPlaneStep)

Variables

- dng_suite **gDNGSuite**

7.7.1 Detailed Description

Indirection mechanism for performance-critical routines that might be replaced with hand-optimized or hardware-specific implementations.

7.8 dng_camera_profile.h File Reference

Classes

- class **dng_camera_profile_id**
- class dng_camera_profile

  Container for DNG camera color profile and calibration data.

Variables

- const char ∗ **kProfileName_Embedded**
- const char ∗ **kAdobeCalibrationSignature**

7.8.1 Detailed Description

Support for DNG camera color profile information. Per the DNG 1.1.0 specification, a DNG file can store up to two sets of color profile information for a camera in the DNG file from that camera. The second set is optional and when there are two sets, they represent profiles made under different illumination.

Profiling information is optionally separated into two parts. One part represents a profile for a reference camera. (ColorMatrix1 and ColorMatrix2 here.) The second is a per-camera calibration that takes into account unit-to-unit variation. This is designed to allow replacing the reference color matrix with one of one's own construction while maintaining any unit-specific calibration the camera manufacturer may have provided.

See Appendix 6 of the DNG 1.1.0 specification for more information.

7.9 dng_color_space.h File Reference

Classes

- class dng_function_GammaEncode_sRGB

  A dng_1d_function for gamma encoding in sRGB color space.

- class dng_function_GammaEncode_1_8

  A dng_1d_function for gamma encoding with 1.8 gamma.

- class dng_function_GammaEncode_2_2

  A dng_1d_function for gamma encoding with 2.2 gamma.

- class dng_color_space

  An abstract color space.

- class dng_space_sRGB

  Singleton class for sRGB color space.

- class dng_space_AdobeRGB

  Singleton class for AdobeRGB color space.

- class dng_space_ColorMatch

  Singleton class for ColorMatch color space.

- class dng_space_ProPhoto

  Singleton class for ProPhoto RGB color space.

- class dng_space_GrayGamma18

Singleton class for gamma 1.8 grayscale color space.

- class dng_space_GrayGamma22

  Singleton class for gamma 2.2 grayscale color space.

7.9.1 Detailed Description

Standard gamma functions and color spaces used within the DNG SDK.

7.10 dng_color_spec.h File Reference

Classes

- class dng_color_spec

Functions

- dng_matrix_3by3 MapWhiteMatrix (const dng_xy_coord &white1, const dng_xy_coord &white2)

  Compute a 3x3 matrix which maps colors from white point white1 to white point white2.

7.10.1 Detailed Description

Class for holding a specific color transform.

7.10.2 Function Documentation

7.10.2.1 dng_matrix_3by3 MapWhiteMatrix (const dng_xy_coord & *white1*, const dng_xy_coord & *white2*)

Compute a 3x3 matrix which maps colors from white point white1 to white point white2.

Uses linearized Bradford adaptation matrix to compute a mapping from colors measured with one white point (white1) to another (white2).

7.11 dng_date_time.h File Reference

Classes

- class dng_date_time

  Class for holding a date/time and converting to and from relevant date/time formats.

- class dng_time_zone

  Class for holding a time zone.

- class dng_date_time_info

  Class for holding complete data/time/zone information.

- class dng_date_time_storage_info

  Store file offset from which date was read.

Enumerations

- enum dng_date_time_format { **dng_date_time_format_unknown** = 0, dng_date_time_format_exif = 1, dng_date_time_format_unix_little_endian = 2, dng_date_time_format_unix_big_endian = 3 }

  Tag to encode date represenation format.

Functions

- void CurrentDateTimeAndZone (dng_date_time_info &info)
- void DecodeUnixTime (uint32 unixTime, dng_date_time &dt)

  Convert UNIX "seconds since Jan 1, 1970" time to a dng_date_time.

- dng_time_zone LocalTimeZone (const dng_date_time &dt)

7.11.1 Detailed Description

Functions and classes for working with dates and times in DNG files.

7.11.2 Enumeration Type Documentation

7.11.2.1 enum dng_date_time_format

Tag to encode date represenation format.

Enumerator:

dng_date_time_format_exif Date format not known.

dng_date_time_format_unix_little_endian EXIF date string.

dng_date_time_format_unix_big_endian 32-bit UNIX time as 4-byte little endian

7.11.3 Function Documentation

7.11.3.1 void CurrentDateTimeAndZone (dng_date_time_info & *info*)

Get the current date/time and timezone.

Parameters:

dt Receives current data/time/zone.

References dng_date_time::fDay, dng_date_time::fHour, dng_date_time::fMinute, dng_date_time::fMonth, dng_date_time::fSecond, dng_date_time::fYear, dng_date_time_info::SetDateTime(), dng_time_zone::SetOffsetMinutes(), and dng_date_time_info::SetZone().

7.11.3.2 dng_time_zone LocalTimeZone (const dng_date_time & *dt*)

Return timezone of current location at a given date.

Parameters:

dt Date at which to compute timezone difference. (For example, used to determine Daylight Savings, etc.)

Return values:

Time zone for date/time dt.

References dng_date_time::fDay, dng_date_time::fHour, dng_date_time::fMinute, dng_date_time::fMonth, dng_date_time::fSecond, dng_date_time::fYear, dng_time_zone::IsValid(), dng_date_time::IsValid(), dng_time_zone::SetOffsetSeconds(), and dng_date_time_info::TimeZone().

7.12 dng_errors.h File Reference

Typedefs

- typedef int32 dng_error_code

  Type for all errors used in DNG SDK. Generally held inside a dng_exception.

Enumerations

- enum {

  dng_error_none = 0, **dng_error_unknown** = 100000, **dng_error_not_yet_implemented**, **dng_error_silent**,

  dng_error_user_canceled, **dng_error_host_insufficient**, **dng_error_memory**, **dng_error_bad_format**,

  dng_error_matrix_math, **dng_error_open_file**, **dng_error_read_file**, **dng_error_write_file**,

  dng_error_end_of_file, **dng_error_bad_raw_digest**, **dng_error_bad_original_digest** }

7.12.1 Detailed Description

Error code values.

7.13 dng_exceptions.h File Reference

Classes

- class dng_exception

  All exceptions thrown by the DNG SDK use this exception class.

Functions

- void ReportWarning (const char *message, const char *sub_message=NULL)

  Display a warning message. Note that this may just eat the message.

- void ReportError (const char *message, const char *sub_message=NULL)

  Display an error message. Note that this may just eat the message.

- void Throw_dng_error (dng_error_code err, const char *message=NULL, const char *sub_message=NULL, bool silent=false)

  Throw an exception based on an arbitrary error code.

- void Fail_dng_error (dng_error_code err)

  Convenience function to throw dng_exception with error code if error_code is not dng_error_none .

- void ThrowProgramError (const char *sub_message=NULL)

Convenience function to throw dng_exception with error code dng_error_unknown .

- void ThrowNotYetImplemented (const char *sub_message=NULL)

  Convenience function to throw dng_exception with error code dng_error_not_yet_implemented .

- void ThrowSilentError ()

  Convenience function to throw dng_exception with error code dng_error_silent .

- void ThrowUserCanceled ()

  Convenience function to throw dng_exception with error code dng_error_user_canceled .

- void ThrowHostInsufficient (const char *sub_message=NULL)

  Convenience function to throw dng_exception with error code dng_error_host_insufficient .

- void ThrowMemoryFull (const char *sub_message=NULL)

  Convenience function to throw dng_exception with error code dng_error_memory .

- void ThrowBadFormat (const char *sub_message=NULL)

  Convenience function to throw dng_exception with error code dng_error_bad_format .

- void ThrowMatrixMath (const char *sub_message=NULL)

  Convenience function to throw dng_exception with error code dng_error_matrix_math .

- void ThrowOpenFile (const char *sub_message=NULL, bool silent=false)

  Convenience function to throw dng_exception with error code dng_error_open_file .

- void ThrowReadFile (const char *sub_message=NULL)

  Convenience function to throw dng_exception with error code dng_error_read_file .

- void ThrowWriteFile (const char *sub_message=NULL)

  Convenience function to throw dng_exception with error code dng_error_write_file .

- void ThrowEndOfFile (const char *sub_message=NULL)

  Convenience function to throw dng_exception with error code dng_error_end_of_file .

- void ThrowBadRawDigest ()

  Convenience function to throw dng_exception with error code dng_error_bad_raw_digest .

- void ThrowBadOriginalDigest ()

  Convenience function to throw dng_exception with error code dng_error_bad_original_digest .

7.13.1 Detailed Description

C++ exception support for DNG SDK.

7.14 dng_exif.h File Reference

Classes

- class dng_exif

  Container class for parsing and holding EXIF tags.

7.14.1 Detailed Description

EXIF read access support. See the EXIF specification for full description of tags.

7.15 dng_fast_module.h File Reference

7.15.1 Detailed Description

Include file to set optimization to highest level for performance-critical routines. Normal files should have otpimization set to normal level to save code size as there is less cache pollution this way.

7.16 dng_file_stream.h File Reference

Classes

- class dng_file_stream

  A stream to/from a disk file. See dng_stream for read/write interface.

7.16.1 Detailed Description

Simple, portable, file read/write support.

7.17 dng_filter_task.h File Reference

Classes

- class dng_filter_task

  Represents a task which filters an area of a source dng_image to an area of a destination dng_image.

7.17.1 Detailed Description

Specialization of dng_area_task for processing an area from one dng_image to an area of another.

7.18 dng_fingerprint.h File Reference

Classes

- class dng_fingerprint

  Container fingerprint (MD5 only at present).

- class **dng_md5_printer**
- class **dng_md5_printer_stream**

7.18.1 Detailed Description

Fingerprint (cryptographic hashing) support for generating strong hashes of image data.

7.19 dng_flags.h File Reference

Defines

- #define **qDNGDebug** 0
- #define **qDNGLittleEndian** !qDNGBigEndian
- #define **qDNG64Bit** 0
- #define **qDNGThreadSafe** (qMacOS || qWinOS)
- #define **qDNGValidateTarget** 0
- #define **qDNGValidate** qDNGValidateTarget
- #define **qDNGPrintMessages** qDNGValidate

7.19.1 Detailed Description

Conndditional compilation flags for DNG SDK.

All conditional compilation macros for the DNG SDK begin with a lowercase 'q'.

7.20 dng_globals.h File Reference

7.20.1 Detailed Description

Definitions of global variables controling DNG SDK behavior. Currenntly only used for validation control.

7.21 dng_host.h File Reference

Classes

- class dng_host

  The main class for communication between the application and the DNG SDK. Used to customize memory allocation and other behaviors.

7.21.1 Detailed Description

Class definition for dng_host, initial point of contact and control between host application and DNG SDK.

7.22 dng_ifd.h File Reference

Classes

- class **dng_preview_info**
- class dng_ifd

  Container for a single image file directory of a digital negative.

7.22.1 Detailed Description

DNG image file directory support.

7.23 dng_image.h File Reference

Classes

- class dng_tile_buffer

  Class to get resource acquisition is instantiation behavior for tile buffers. Can be dirty or constant tile access.

- class dng_const_tile_buffer

  Class to get resource acquisition is instantiation behavior for constant (read-only) tile buffers.

- class dng_dirty_tile_buffer

  Class to get resource acquisition is instantiation behavior for dirty (writable) tile buffers.

- class dng_image

  Base class for holding image data in DNG SDK. See dng_simple_image for derived class most often used in DNG SDK.

7.23.1 Detailed Description

Support for working with image data in DNG SDK.

7.24 dng_image_writer.h File Reference

Classes

- class **dng_resolution**
- class **tiff_tag**
- class **tag_data_ptr**
- class **tag_string**
- class **tag_encoded_text**
- class **tag_uint8**
- class **tag_uint8_ptr**
- class **tag_uint16**
- class **tag_int16_ptr**
- class **tag_uint16_ptr**
- class **tag_uint32**
- class **tag_uint32_ptr**
- class **tag_urational**
- class **tag_urational_ptr**

- class **tag_srational**
- class **tag_srational_ptr**
- class **tag_matrix**
- class **tag_icc_profile**
- class **tag_cfa_pattern**
- class **tag_exif_date_time**
- class **tag_iptc**
- class **tag_xmp**
- class **tag_adobe_data**
- class **dng_tiff_directory**
- class **dng_basic_tag_set**
- class **exif_tag_set**
- class **tiff_dng_extended_color_profile**
- class dng_image_writer

  Support for writing dng_image or dng_negative instances to a dng_stream in TIFF or DNG format.

7.24.1 Detailed Description

Support for writing DNG images to files.

7.25 dng_info.h File Reference

Classes

- class dng_info

  Top-level structure of DNG file with access to metadata.

7.25.1 Detailed Description

Class for holding top-level information about a DNG image.

7.26 dng_iptc.h File Reference

Classes

- class dng_iptc

  Class for reading and holding IPTC metadata associated with a DNG file.

7.26.1 Detailed Description

Support for IPTC metadata within DNG files.

7.27 dng_linearization_info.h File Reference

Classes

- class dng_linearization_info

  Class for managing data values related to DNG linearization.

7.27.1 Detailed Description

Support for linearization table and black level tags.

7.28 dng_matrix.h File Reference

Classes

- class **dng_matrix**
- class **dng_matrix_3by3**
- class **dng_matrix_4by3**
- class **dng_vector**
- class **dng_vector_3**

Functions

- dng_matrix **operator**∗ (const dng_matrix &A, const dng_matrix &B)
- dng_vector **operator**∗ (const dng_matrix &A, const dng_vector &B)
- dng_matrix **operator**∗ (real64 scale, const dng_matrix &A)
- dng_vector **operator**∗ (real64 scale, const dng_vector &A)
- dng_matrix **operator+** (const dng_matrix &A, const dng_matrix &B)
- dng_matrix **Transpose** (const dng_matrix &A)
- dng_matrix **Invert** (const dng_matrix &A)
- dng_matrix **Invert** (const dng_matrix &A, const dng_matrix &hint)
- real64 **MaxEntry** (const dng_matrix &A)
- real64 **MaxEntry** (const dng_vector &A)
- real64 **MinEntry** (const dng_matrix &A)
- real64 **MinEntry** (const dng_vector &A)

7.28.1 Detailed Description

Matrix and vector classes, including specialized 3x3 and 4x3 versions as well as length 3 vectors.

7.29 dng_memory_stream.h File Reference

Classes

- class dng_memory_stream

  A dng_stream which can be read from or written to memory.

7.29.1 Detailed Description

Stream abstraction to/from in-memory data.

7.30 dng_mosaic_info.h File Reference

Classes

- class dng_mosaic_info

  Support for describing color filter array patterns and manipulating mosaic sample data.

7.30.1 Detailed Description

Support for descriptive information about color filter array patterns.

7.31 dng_negative.h File Reference

Classes

- class dng_negative

  Main class for holding DNG image data and associated metadata.

7.31.1 Detailed Description

7.32 dng_pixel_buffer.h File Reference

Classes

- class dng_pixel_buffer

  Holds a buffer of pixel data with "pixel geometry" metadata.

Defines

- #define **qDebugPixelType** 0
- #define **ASSERT_PIXEL_TYPE**(typeVal) DNG_ASSERT (fPixelType == typeVal, "Pixel type access mismatch")

Functions

- void OptimizeOrder (const void ∗&sPtr, void ∗&dPtr, uint32 sPixelSize, uint32 dPixelSize, uint32 &count0, uint32 &count1, uint32 &count2, int32 &sStep0, int32 &sStep1, int32 &sStep2, int32 &dStep0, int32 &dStep1, int32 &dStep2)

  Compute best set of step values for a given source and destination area and stride.

- void **OptimizeOrder** (const void ∗&sPtr, uint32 sPixelSize, uint32 &count0, uint32 &count1, uint32 &count2, int32 &sStep0, int32 &sStep1, int32 &sStep2)

- void **OptimizeOrder** (void ∗&dPtr, uint32 dPixelSize, uint32 &count0, uint32 &count1, uint32 &count2, int32 &dStep0, int32 &dStep1, int32 &dStep2)

7.32.1 Detailed Description

Support for holding buffers of sample data.

7.33 dng_read_image.h File Reference

Classes

- class **dng_row_interleaved_image**
- class **dng_read_image**

7.33.1 Detailed Description

Support for DNG image reading.

7.34 dng_render.h File Reference

Classes

- class dng_function_exposure_ramp

  Curve for pre-exposure-compensation adjustment based on noise floor, shadows, and highlight level.

- class dng_function_exposure_tone

  Exposure compensation curve for a given compensation amount in stops using quadric for roll-off.

- class dng_tone_curve_acr3_default

  Default ACR3 tone curve.

- class dng_function_gamma_encode

  Encoding gamma curve for a given color space.

- class dng_render

  Class used to render digital negative to displayable image.

7.34.1 Detailed Description

Classes for conversion of RAW data to final image.

7.35 dng_sdk_limits.h File Reference

Variables

- const uint32 kMaxDNGPreviews = 20
- const uint32 kMaxSubIFDs = kMaxDNGPreviews + 1

  The maximum number of SubIFDs that will be parsed.

- const uint32 kMaxChainedIFDs = 10

  The maximum number of chained IFDs that will be parsed.

- const uint32 kMaxSamplesPerPixel = 4

The maximum number of samples per pixel.

- const uint32 kMaxColorPlanes = kMaxSamplesPerPixel

  Maximum number of color planes.

- const uint32 kMaxCFAPattern = 8

  The maximum size of a CFA repeating pattern.

- const uint32 kMaxBlackPattern = 8

  The maximum size of a black level repeating pattern.

- const uint32 kMaxMaskedAreas = 4

  The maximum number of masked area rectangles.

- const uint32 kMaxImageSide = 65000

  The maximum image size supported (pixels per side).

- const uint32 kMaxMPThreads = 8

  Maximum number of MP threads for dng_area_task operations.

7.35.1 Detailed Description

Collection of constants detailing maximum values used in processing in the DNG SDK.

7.35.2 Variable Documentation

7.35.2.1 const uint32 kMaxDNGPreviews = 20

The maximum number of previews (in addition to the main IFD's thumbnail) that we support embedded in a DNG.

Referenced by dng_image_writer::WriteDNG().

Index

~dng_host
    dng_host, 68

Allocate
    dng_host, 68
    dng_memory_allocator, 94
    dng_memory_data, 102
ApplyOrientation
    dng_negative, 124
Area
    dng_pixel_buffer, 127
AsMemoryBlock
    dng_stream, 156
AutoPtr, 12
    AutoPtr, 13

BestQualityFinalHeight
    dng_negative, 124
BestQualityFinalWidth
    dng_negative, 124
BigEndian
    dng_stream, 154
BlackLevel
    dng_linearization_info, 92
Buffer
    dng_memory_block, 96
    dng_memory_data, 102
Buffer_char
    dng_memory_block, 96, 97
    dng_memory_data, 102, 103
Buffer_int16
    dng_memory_block, 98
    dng_memory_data, 104
Buffer_int32
    dng_memory_block, 99
    dng_memory_data, 105
Buffer_int64
    dng_memory_data, 105, 106
Buffer_real32
    dng_memory_block, 99
    dng_memory_data, 106
Buffer_real64
    dng_memory_block, 99, 100
    dng_memory_data, 106
Buffer_uint16
    dng_memory_block, 97
    dng_memory_data, 103
Buffer_uint32
    dng_memory_block, 98
    dng_memory_data, 104
Buffer_uint64
    dng_memory_data, 105
Buffer_uint8
    dng_memory_block, 97
    dng_memory_data, 103

CalibrationIlluminant1
    dng_camera_profile, 33
CalibrationIlluminant2
    dng_camera_profile, 34
CalibrationTemperature1
    dng_camera_profile, 34
CalibrationTemperature2
    dng_camera_profile, 34
CameraToPCS
    dng_color_spec, 41
CameraWhite
    dng_color_spec, 40
Channels
    dng_color_spec, 40
Clear
    dng_memory_data, 102
ColumnBlack
    dng_linearization_info, 93
ConstPixel
    dng_pixel_buffer, 128
ConstPixel_int16
    dng_pixel_buffer, 131
ConstPixel_int32
    dng_pixel_buffer, 132
ConstPixel_int8
    dng_pixel_buffer, 130
ConstPixel_real32
    dng_pixel_buffer, 133
ConstPixel_uint16
    dng_pixel_buffer, 130

ConstPixel_uint32
dng_pixel_buffer, 132
ConstPixel_uint8
dng_pixel_buffer, 129
CopyArea
dng_image, 81
dng_pixel_buffer, 136
Copyright
dng_camera_profile, 36
CopyToStream
dng_memory_stream, 108
dng_stream, 168
CurrentDateTimeAndZone
dng_date_time.h, 183

Data
dng_stream, 156
DefaultCropArea
dng_negative, 124
DefaultScale
dng_negative, 124
DirtyPixel
dng_pixel_buffer, 128
DirtyPixel_int16
dng_pixel_buffer, 131
DirtyPixel_int32
dng_pixel_buffer, 133
DirtyPixel_int8
dng_pixel_buffer, 130
DirtyPixel_real32
dng_pixel_buffer, 133
DirtyPixel_uint16
dng_pixel_buffer, 131
DirtyPixel_uint32
dng_pixel_buffer, 132
DirtyPixel_uint8
dng_pixel_buffer, 129
dng_date_time.h
dng_date_time_format_exif, 183
dng_date_time_format_unix_big_-endian, 183
dng_date_time_format_unix_little_-endian, 183
dng_date_time_format_exif
dng_date_time.h, 183
dng_date_time_format_unix_big_endian
dng_date_time.h, 183
dng_date_time_format_unix_little_-endian
dng_date_time.h, 183
dng_image
edge_none, 78
edge_repeat, 78
edge_repeat_zero_last, 78
edge_zero, 78
dng_1d_concatenate, 13
dng_1d_concatenate, 14
dng_1d_concatenate, 14
Evaluate, 15
EvaluateInverse, 15
dng_1d_function, 15
Evaluate, 17
EvaluateInverse, 17
dng_1d_function.h, 171
dng_1d_identity, 18
dng_1d_inverse, 19
Evaluate, 19
EvaluateInverse, 20
dng_1d_table, 20
Initialize, 21
Interpolate, 21
dng_1d_table.h, 172
dng_abort_sniffer, 22
SniffForAbort, 23
StartTask, 23
UpdateProgress, 23
dng_abort_sniffer.h, 172
dng_area_task, 24
FindTileSize, 28
Finish, 28
MaxThreads, 25
MaxTileSize, 26
MinTaskArea, 25
Perform, 29
Process, 27
ProcessOnThread, 28
RepeatingTile1, 26
RepeatingTile2, 26
RepeatingTile3, 26
Start, 27
UnitCell, 25
dng_area_task.h, 172

DNG_ASSERT
dng_assertions.h, 173
dng_assertions.h, 173
DNG_ASSERT, 173
dng_auto_ptr.h, 173
dng_bottlenecks.h, 174
dng_camera_profile, 29
CalibrationIlluminant1, 33
CalibrationIlluminant2, 34
CalibrationTemperature1, 34
CalibrationTemperature2, 34
Copyright, 36
EqualData, 36
HueSatMapForWhite, 37
IsValid, 36
Name, 33
NameIsEmbedded, 33
ParseExtended, 37
ProfileID, 35
SetCalibrationIlluminant1, 33
SetCalibrationIlluminant2, 33
SetColorMatrix1, 34
SetColorMatrix2, 34
SetCopyright, 35
SetName, 33
SetReductionMatrix1, 35
SetReductionMatrix2, 35
SetUniqueCameraModelRestriction, 36
UniqueCameraModelRestriction, 36
dng_camera_profile.h, 179
dng_color_space, 37
ICCProfile, 39
dng_color_space.h, 180
dng_color_spec, 39
CameraToPCS, 41
CameraWhite, 40
Channels, 40
dng_color_spec, 40
dng_color_spec, 40
NeutralToXY, 41
SetWhiteXY, 40
WhiteXY, 40
dng_color_spec.h, 181
MapWhiteMatrix, 181
dng_const_tile_buffer, 41
dng_const_tile_buffer, 42
dng_const_tile_buffer, 42
dng_date_time, 42
dng_date_time, 43
dng_date_time, 43
IsValid, 44
NotValid, 44
Parse, 44
dng_date_time.h, 182
CurrentDateTimeAndZone, 183
dng_date_time_format, 182
LocalTimeZone, 183
dng_date_time_format
dng_date_time.h, 182
dng_date_time_info, 45
dng_date_time_storage_info, 45
Format, 46
IsValid, 46
Offset, 46
dng_dirty_tile_buffer, 47
dng_dirty_tile_buffer, 47
dng_dirty_tile_buffer, 47
dng_errors.h, 183
dng_exception, 48
dng_exception, 48
dng_exception, 48
ErrorCode, 48
dng_exceptions.h, 184
dng_exif, 49
dng_exif.h, 186
dng_fast_module.h, 186
dng_file_stream, 53
dng_file_stream, 53
dng_file_stream, 53
dng_file_stream.h, 186
dng_filter_task, 54
dng_filter_task, 55
dng_filter_task, 55
Process, 56
ProcessArea, 56
SrcArea, 55
SrcTileSize, 55
Start, 56
dng_filter_task.h, 187
dng_fingerprint, 57
dng_fingerprint.h, 187

dng_flags.h, 187
dng_function_exposure_ramp, 58
    Evaluate, 59
dng_function_exposure_tone, 59
dng_function_gamma_encode, 60
    Evaluate, 61
dng_function_GammaEncode_1_8, 61
    Evaluate, 62
    EvaluateInverse, 62
dng_function_GammaEncode_2_2, 63
    Evaluate, 63
    EvaluateInverse, 64
dng_function_GammaEncode_sRGB, 64
    Evaluate, 65
    EvaluateInverse, 65
dng_globals.h, 188
dng_host, 66
    ~dng_host, 68
    Allocate, 68
    dng_host, 68
    dng_host, 68
    ForPreview, 69
    IsTransientError, 71
    KeepStage1, 70
    KeepStage2, 71
    Make_dng_exif, 72
    Make_dng_ifd, 72
    Make_dng_image, 72
    Make_dng_negative, 72
    Make_dng_shared, 72
    PerformAreaTask, 71
    SetCropFactor, 70
    SetForPreview, 69
    SetKeepDNGPrivate, 71
    SetKeepOriginalFile, 71
    SetKeepStage1, 70
    SetKeepStage2, 71
    SetMaximumSize, 70
    SetMinimumSize, 69
    SetNeedsImage, 69
    SetNeedsMeta, 69
    SetPreferredSize, 70
    SniffForAbort, 69
dng_host.h, 188
dng_ifd, 73
dng_ifd.h, 188
dng_image, 76
    CopyArea, 81
    edge_option, 78
    EqualArea, 81
    Get, 79
    PixelRange, 79
    PixelSize, 79
    PixelType, 78
    Put, 80
    Rotate, 80
    SetPixelRange, 79
    SetPixelType, 79
    Trim, 80
dng_image.h, 189
dng_image_writer, 82
    WriteDNG, 84
    WriteTIFF, 83
dng_image_writer.h, 189
dng_info, 85
    IsValidDNG, 87
    Parse, 87
dng_info.h, 190
dng_iptc, 87
    IsEmpty, 89
    NotEmpty, 89
    Parse, 89
    Spool, 90
dng_iptc.h, 190
dng_linearization_info, 90
    BlackLevel, 92
    ColumnBlack, 93
    fActiveArea, 93
    fLinearizationTable, 94
    fMaskedArea, 93
    Linearize, 92
    RowBlack, 93
dng_linearization_info.h, 191
dng_matrix.h, 191
dng_memory_allocator, 94
    Allocate, 94
dng_memory_block, 95
    Buffer, 96
    Buffer_char, 96, 97
    Buffer_int16, 98
    Buffer_int32, 99
    Buffer_real32, 99

Buffer_real64, 99, 100
Buffer_uint16, 97
Buffer_uint32, 98
Buffer_uint8, 97
LogicalSize, 96
dng_memory_data, 100
Allocate, 102
Buffer, 102
Buffer_char, 102, 103
Buffer_int16, 104
Buffer_int32, 105
Buffer_int64, 105, 106
Buffer_real32, 106
Buffer_real64, 106
Buffer_uint16, 103
Buffer_uint32, 104
Buffer_uint64, 105
Buffer_uint8, 103
Clear, 102
dng_memory_data, 101
dng_memory_data, 101
dng_memory_stream, 107
CopyToStream, 108
dng_memory_stream, 108
dng_memory_stream, 108
dng_memory_stream.h, 192
dng_mosaic_info, 109
DownScale, 111
DstSize, 111
fBayerGreenSplit, 113
fCFALayout, 113
FullScale, 111
Interpolate, 113
InterpolateFast, 112
InterpolateGeneric, 112
IsColorFilterArray, 110
SetFourColorBayer, 110
dng_mosaic_info.h, 192
dng_negative, 114
ApplyOrientation, 124
BestQualityFinalHeight, 124
BestQualityFinalWidth, 124
DefaultCropArea, 124
DefaultScale, 124
SetCameraCalibration1, 124
SetCameraCalibration2, 125
dng_negative.h, 192
dng_pixel_buffer, 125
Area, 127
ConstPixel, 128
ConstPixel_int16, 131
ConstPixel_int32, 132
ConstPixel_int8, 130
ConstPixel_real32, 133
ConstPixel_uint16, 130
ConstPixel_uint32, 132
ConstPixel_uint8, 129
CopyArea, 136
DirtyPixel, 128
DirtyPixel_int16, 131
DirtyPixel_int32, 133
DirtyPixel_int8, 130
DirtyPixel_real32, 133
DirtyPixel_uint16, 131
DirtyPixel_uint32, 132
DirtyPixel_uint8, 129
EqualArea, 138
FlipH, 138
FlipV, 138
FlipZ, 138
PixelRange, 127
Planes, 127
PlaneStep, 128
RepeatArea, 137
RepeatPhase, 137
RowStep, 128
SetConstant, 134
SetConstant_int16, 135
SetConstant_real32, 135
SetConstant_uint16, 134
SetConstant_uint32, 135
SetConstant_uint8, 134
SetZero, 136
ShiftRight, 137
dng_pixel_buffer.h, 193
dng_read_image.h, 193
dng_render, 139
dng_render, 140
dng_render, 140
Exposure, 140
FinalPixelType, 142
FinalSpace, 142

MaximumSize, 142
Render, 143
SetExposure, 140
SetFinalPixelType, 142
SetFinalSpace, 141
SetMaximumSize, 142
SetShadows, 141
SetToneCurve, 141
SetWhiteXY, 140
Shadows, 141
ToneCurve, 141
WhiteXY, 140
dng_render.h, 194
dng_sdk_limits.h, 194
kMaxDNGPreviews, 195
dng_simple_image, 143
dng_sniffer_task, 144
dng_sniffer_task, 145
dng_sniffer_task, 145
UpdateProgress, 145, 146
dng_space_AdobeRGB, 146
dng_space_ColorMatch, 147
dng_space_GrayGamma18, 148
dng_space_GrayGamma22, 149
dng_space_ProPhoto, 150
dng_space_sRGB, 150
dng_stream, 151
AsMemoryBlock, 156
BigEndian, 154
CopyToStream, 168
Data, 156
dng_stream, 154
dng_stream, 154
DuplicateStream, 168
Get, 157
Get_CString, 164
Get_int16, 161
Get_int32, 161
Get_int64, 162
Get_int8, 160
Get_real32, 163
Get_real64, 163
Get_uint16, 158
Get_uint32, 159
Get_uint64, 160
Get_uint8, 158
Get_UString, 164
Length, 155
LittleEndian, 155
OffsetInOriginalFile, 156
Position, 155
PositionInOriginalFile, 156
Put, 157
Put_int16, 161
Put_int32, 162
Put_int64, 162
Put_int8, 161
Put_real32, 163
Put_real64, 164
Put_uint16, 159
Put_uint32, 159
Put_uint64, 160
Put_uint8, 158
PutZeros, 165
SetBigEndian, 154
SetLength, 157
SetLittleEndian, 155
SetSniffer, 167
SetSwapBytes, 154
Skip, 157
Sniffer, 167
SwapBytes, 154
TagValue_int32, 165
TagValue_real64, 167
TagValue_srational, 166
TagValue_uint32, 165
TagValue_urational, 166
dng_tile_buffer, 168
dng_tile_buffer, 169
dng_tile_buffer, 169
dng_time_zone, 170
dng_tone_curve_acr3_default, 170
DownScale
dng_mosaic_info, 111
DstSize
dng_mosaic_info, 111
DuplicateStream
dng_stream, 168

edge_none
dng_image, 78
edge_repeat

dng_image, 78
edge_repeat_zero_last
dng_image, 78
edge_zero
dng_image, 78
edge_option
dng_image, 78
EqualArea
dng_image, 81
dng_pixel_buffer, 138
EqualData
dng_camera_profile, 36
ErrorCode
dng_exception, 48
Evaluate
dng_1d_concatenate, 15
dng_1d_function, 17
dng_1d_inverse, 19
dng_function_exposure_ramp, 59
dng_function_gamma_encode, 61
dng_function_GammaEncode_1_8, 62
dng_function_GammaEncode_2_2, 63
dng_function_GammaEncode_-sRGB, 65
EvaluateInverse
dng_1d_concatenate, 15
dng_1d_function, 17
dng_1d_inverse, 20
dng_function_GammaEncode_1_8, 62
dng_function_GammaEncode_2_2, 64
dng_function_GammaEncode_-sRGB, 65
Exposure
dng_render, 140

fActiveArea
dng_linearization_info, 93
fBayerGreenSplit
dng_mosaic_info, 113
fCFALayout
dng_mosaic_info, 113
FinalPixelType
dng_render, 142
FinalSpace
dng_render, 142
FindTileSize
dng_area_task, 28
Finish
dng_area_task, 28
fLinearizationTable
dng_linearization_info, 94
FlipH
dng_pixel_buffer, 138
FlipV
dng_pixel_buffer, 138
FlipZ
dng_pixel_buffer, 138
fMaskedArea
dng_linearization_info, 93
Format
dng_date_time_storage_info, 46
ForPreview
dng_host, 69
FullScale
dng_mosaic_info, 111

Get
dng_image, 79
dng_stream, 157
Get_CString
dng_stream, 164
Get_int16
dng_stream, 161
Get_int32
dng_stream, 161
Get_int64
dng_stream, 162
Get_int8
dng_stream, 160
Get_real32
dng_stream, 163
Get_real64
dng_stream, 163
Get_uint16
dng_stream, 158
Get_uint32
dng_stream, 159
Get_uint64

dng_stream, 160
Get_uint8
dng_stream, 158
Get_UString
dng_stream, 164

HueSatMapForWhite
dng_camera_profile, 37

ICCProfile
dng_color_space, 39
Initialize
dng_1d_table, 21
Interpolate
dng_1d_table, 21
dng_mosaic_info, 113
InterpolateFast
dng_mosaic_info, 112
InterpolateGeneric
dng_mosaic_info, 112
IsColorFilterArray
dng_mosaic_info, 110
IsEmpty
dng_iptc, 89
IsTransientError
dng_host, 71
IsValid
dng_camera_profile, 36
dng_date_time, 44
dng_date_time_storage_info, 46
IsValidDNG
dng_info, 87

KeepStage1
dng_host, 70
KeepStage2
dng_host, 71
kMaxDNGPreviews
dng_sdk_limits.h, 195

Length
dng_stream, 155
Linearize
dng_linearization_info, 92
LittleEndian
dng_stream, 155
LocalTimeZone
dng_date_time.h, 183
LogicalSize
dng_memory_block, 96

Make_dng_exif
dng_host, 72
Make_dng_ifd
dng_host, 72
Make_dng_image
dng_host, 72
Make_dng_negative
dng_host, 72
Make_dng_shared
dng_host, 72
MapWhiteMatrix
dng_color_spec.h, 181
MaximumSize
dng_render, 142
MaxThreads
dng_area_task, 25
MaxTileSize
dng_area_task, 26
MinTaskArea
dng_area_task, 25

Name
dng_camera_profile, 33
NameIsEmbedded
dng_camera_profile, 33
NeutralToXY
dng_color_spec, 41
NotEmpty
dng_iptc, 89
NotValid
dng_date_time, 44

Offset
dng_date_time_storage_info, 46
OffsetInOriginalFile
dng_stream, 156

Parse
dng_date_time, 44
dng_info, 87
dng_iptc, 89

ParseExtended
dng_camera_profile, 37
Perform
dng_area_task, 29
PerformAreaTask
dng_host, 71
PixelRange
dng_image, 79
dng_pixel_buffer, 127
PixelSize
dng_image, 79
PixelType
dng_image, 78
Planes
dng_pixel_buffer, 127
PlaneStep
dng_pixel_buffer, 128
Position
dng_stream, 155
PositionInOriginalFile
dng_stream, 156
Process
dng_area_task, 27
dng_filter_task, 56
ProcessArea
dng_filter_task, 56
ProcessOnThread
dng_area_task, 28
ProfileID
dng_camera_profile, 35
Put
dng_image, 80
dng_stream, 157
Put_int16
dng_stream, 161
Put_int32
dng_stream, 162
Put_int64
dng_stream, 162
Put_int8
dng_stream, 161
Put_real32
dng_stream, 163
Put_real64
dng_stream, 164
Put_uint16
dng_stream, 159
Put_uint32
dng_stream, 159
Put_uint64
dng_stream, 160
Put_uint8
dng_stream, 158
PutZeros
dng_stream, 165

Render
dng_render, 143
RepeatArea
dng_pixel_buffer, 137
RepeatingTile1
dng_area_task, 26
RepeatingTile2
dng_area_task, 26
RepeatingTile3
dng_area_task, 26
RepeatPhase
dng_pixel_buffer, 137
Rotate
dng_image, 80
RowBlack
dng_linearization_info, 93
RowStep
dng_pixel_buffer, 128

SetBigEndian
dng_stream, 154
SetCalibrationIlluminant1
dng_camera_profile, 33
SetCalibrationIlluminant2
dng_camera_profile, 33
SetCameraCalibration1
dng_negative, 124
SetCameraCalibration2
dng_negative, 125
SetColorMatrix1
dng_camera_profile, 34
SetColorMatrix2
dng_camera_profile, 34
SetConstant
dng_pixel_buffer, 134
SetConstant_int16

dng_pixel_buffer, 135
SetConstant_real32
dng_pixel_buffer, 135
SetConstant_uint16
dng_pixel_buffer, 134
SetConstant_uint32
dng_pixel_buffer, 135
SetConstant_uint8
dng_pixel_buffer, 134
SetCopyright
dng_camera_profile, 35
SetCropFactor
dng_host, 70
SetExposure
dng_render, 140
SetFinalPixelType
dng_render, 142
SetFinalSpace
dng_render, 141
SetForPreview
dng_host, 69
SetFourColorBayer
dng_mosaic_info, 110
SetKeepDNGPrivate
dng_host, 71
SetKeepOriginalFile
dng_host, 71
SetKeepStage1
dng_host, 70
SetKeepStage2
dng_host, 71
SetLength
dng_stream, 157
SetLittleEndian
dng_stream, 155
SetMaximumSize
dng_host, 70
dng_render, 142
SetMinimumSize
dng_host, 69
SetName
dng_camera_profile, 33
SetNeedsImage
dng_host, 69
SetNeedsMeta
dng_host, 69
SetPixelRange
dng_image, 79
SetPixelType
dng_image, 79
SetPreferredSize
dng_host, 70
SetReductionMatrix1
dng_camera_profile, 35
SetReductionMatrix2
dng_camera_profile, 35
SetShadows
dng_render, 141
SetSniffer
dng_stream, 167
SetSwapBytes
dng_stream, 154
SetToneCurve
dng_render, 141
SetUniqueCameraModelRestriction
dng_camera_profile, 36
SetWhiteXY
dng_color_spec, 40
dng_render, 140
SetZero
dng_pixel_buffer, 136
Shadows
dng_render, 141
ShiftRight
dng_pixel_buffer, 137
Skip
dng_stream, 157
Sniffer
dng_stream, 167
SniffForAbort
dng_abort_sniffer, 23
dng_host, 69
Spool
dng_iptc, 90
SrcArea
dng_filter_task, 55
SrcTileSize
dng_filter_task, 55
Start
dng_area_task, 27
dng_filter_task, 56
StartTask

dng_abort_sniffer, 23
SwapBytes
dng_stream, 154

TagValue_int32
dng_stream, 165
TagValue_real64
dng_stream, 167
TagValue_srational
dng_stream, 166
TagValue_uint32
dng_stream, 165
TagValue_urational
dng_stream, 166
ToneCurve
dng_render, 141
Trim
dng_image, 80

UniqueCameraModelRestriction
dng_camera_profile, 36
UnitCell
dng_area_task, 25
UpdateProgress
dng_abort_sniffer, 23
dng_sniffer_task, 145, 146

WhiteXY
dng_color_spec, 40
dng_render, 140
WriteDNG
dng_image_writer, 84
WriteTIFF
dng_image_writer, 83