MinImgAPI Library Reference

version 1.0

Generated by Doxygen 1.7.5.1

Fri Oct 7 2011 20:40:37

Contents

1 Overview

MinImgAPI is an open-source platform-independent library that contains image processing functions which treat the image as a matrix. That is, these functions know nothing about "pixel" essence. Examples of such functions are: allocation memory for image data, copying images, rotating an image by right angle and others.

For the internal representation of images is used cross-platform open-source container - MinImg (see Image Representation section for more information). The advantages of this container are the using a minimal number of fields needed to represent the bitmap image and the easy way to cast it to other standard and popular containers (for instance, Windows DIB, GDI+ BitmapData, Intel/OpenCV IplImage).

The library is written in C++ and can be compiled under Linux (GCC) and Windows (MSVC 8 and later). Though the library has been written in C++, it has C interface, so it can be embedded in different systems.

2 Quick Tutorial

This tutorial is intended to get you start using **MinImgAPI** library. The tutorial demonstrates popular use cases of library usages, therefore it is not a complete or detailed documentation. Note also, that some secondary operations will be purposely omitted for brevity.

2.1 Allocate and Deallocate Images

This is the most popular use case of usage the library. To do that you should define image header at first and then allocate memory for image data. The following example shows the way to allocate 24-bit RGB image of 640x480 size:

```
// Define header
MinImg image = {0};
image.width = 640;
image.height = 480;
```

```
image.channels = 3;
image.channelDepth = 1;
image.format = FMT_UINT;

// Allocates the memory for the image data
PROPAGATE_ERROR(AllocMinImage(&image, 16));
```

If you use `AllocMinImage()` for allocation of memory then you **have to** use `FreeMinImage()` to deallocate that. The following example demonstates the usage of `FreeMinImage()` function:

```
PROPAGATE_ERROR(FreeMinImage(&image));
```

2.2 Copy Images

Another popular use case is cloning the image. Let we have `sourceImage` and want to clone it. The following code shows the proper way to do that:

```
// Define clone image
MinImg cloneImage = {0};

// Make a copy of the header and allocate it
PROPAGATE_ERROR(CloneMinImagePrototype(&cloneImage, &sourceImage));

// Copy image data
PROPAGATE_ERROR(CopyMinImage(&cloneImage, &sourceImage));
```

3 MinImgAPI License Agreements

3.1 Library License Agreement

MinImgAPI is released under FreeBSD License. It is free for both academic and commercial use.

```
ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are those
of the authors and should not be interpreted as representing official policies,
either expressed or implied, of copyright holders.
```

3.2 Documentation License Agreement

This documentation is released under FreeBSD Documentation License. It is free for both academic and commercial use.

```
Copyright (c) 2011, Smart Engines Limited. All rights reserved.

All rights reserved.

Redistribution and use in source (doxygen documentation blocks) and 'compiled'
forms (HTML, PDF, PostScript, RTF and so forth) with or without modification,
are permitted provided that the following conditions are met:

   1. Redistributions of source code (doxygen documentation blocks) must retain

      the above copyright notice, this list of conditions and the following
      disclaimer as the first lines of this file unmodified.

   2. Redistributions in compiled form (converted to PDF, PostScript, RTF and
      other formats) must reproduce the above copyright notice, this list of
      conditions and the following disclaimer in the documentation and/or other
      materials provided with the distribution.

THIS DOCUMENTATION IS PROVIDED BY COPYRIGHT HOLDERS "AS IS" AND ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
      EVENT
SHALL COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF
ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

4 Module Documentation

4.1 MinImgAPI Library API

This section describes an application programming interface (API) of **MinImgAPI** library. Though **MinImgAPI** has been written in C++, it has C interface to make it easy embedding the library in different systems.

Defines

- #define IS_BY_DEFAULT(a)

  Specifies a default value for a parameter of a function.
- #define MINIMGAPI_API

  Specifies storage-class information (only for MSC).

Functions

- int AllocMinImage (MinImg *pImg, int alignment)

  Allocates an image.
- int FreeMinImage (MinImg *pImg)

  Deallocates an image.
- int GetMinImageType (const MinImg *pImg)

  Returns type of an image channel element.
- int SetMinImageType (MinImg *pImg, MinTyp type)

  Assigns type to the image.
- int CopyMinImage (const MinImg *pDst, const MinImg *pSrc)

  Copies one image to another.
- int FlipMinImage (const MinImg *pDst, const MinImg *pSrc, DirectionOption dir)

  Flips an image around vertical or horizontal axis.
- int TransposeMinImage (const MinImg *pDst, const MinImg *pSrc)

  Transposes an image.
- int RotateMinImageBy90 (const MinImg *pDst, const MinImg *pSrc, int nRotations)

  Rotates an image by 90 degrees (clockwise).
- int FillMinImage (const MinImg *pImg, const void *pCanvasValue, int nValueLength)

  Sets every element of an image to a given value.
- int CopyMinImageChannels (const MinImg *pDst, const MinImg *pSrc, const int *pDstChnls, const int *pSrcChnls, int nChnls)

  Copies specified channels of an image to another one.
- int CloneMinImagePrototype (MinImg *pDst, const MinImg *pSrc, AllocationOption allocation)

  Makes a copy of the image header.
- int CloneTransposedMinImagePrototype (MinImg *pDst, const MinImg *pSrc, AllocationOption allocation)

  Makes a copy of the transposed image header.
- int CloneRetypifiedMinImagePrototype (MinImg *pDst, const MinImg *pSrc, MinTyp type, AllocationOption allocation)

  Makes a copy of the image header with another type.
- int CloneDimensionedMinImagePrototype (MinImg *pDst, const MinImg *pSrc, int channels, AllocationOption allocation)

  Makes a copy of the image header with another number of channels.
- int CloneResizedMinImagePrototype (MinImg *pDst, const MinImg *pSrc, int width, int height, AllocationOption allocation)

  Makes a copy of the image header with another size.
- void * GetMinImageLine (const MinImg *pImg, int y, BorderOption border, void *pCanvasLine)

  Returns a pointer to the specified image line.

- int GetMinImageRegion (MinImg *pDst, const MinImg *pSrc, int x0, int y0, int width, int height, RulesOption rules)

  Gets a region of an image.
- int VerFlipMinImageRegion (MinImg *pDst, const MinImg *pSrc)

  Flips an image in vertical without copying.
- int CompareMinImagePrototypes (const MinImg *pDst, const MinImg *pSrc)

  Compares headers of two images.

4.1.1 Function Documentation

4.1.1.1 int AllocMinImage (MinImg * *pImg,* int *alignment*)

Parameters

pImg	The image to be allocated.
alignment	Alignment for image rows, by default 16 bytes.

Returns

`NO_ERRORS` on success or an error code otherwise (see `MinErr`).

The function allocates the memory for the image data. The memory block size to allocate is specified by the "header fields" of the `pImg`. On success the function updates `pImg->pScan0` and `pImg->stride` fields in accordance with allocated memory block.

4.1.1.2 int CloneDimensionedMinImagePrototype (MinImg * *pDst,* const MinImg * *pSrc,* int *channels,* AllocationOption *allocation*)

Parameters

pDst	The destination image.
pSrc	The source image.
channels	The required number of channels of the destination image.
allocation	Specifies whether the destination image should be allocated.

Returns

`NO_ERRORS` on success or an error code otherwise (see `MinErr`).

The function makes a full copy of the image header with required number of channels. If `allocation` is set to `AO_PREALLOCATED` (the default) then a new image will also allocated.

4.1.1.3 int CloneMinImagePrototype (MinImg * *pDst,* const MinImg * *pSrc,* AllocationOption *allocation*)

Parameters

pDst	The destination image.
pSrc	The source image.
allocation	Specifies whether the destination image should be allocated.

Returns

NO_ERRORS on success or an error code otherwise (see MinErr).

The function makes a full copy of the image header. If allocation is set to AO_PREALLOCATED (the default) then a new image will also allocated.

4.1.1.4 int CloneResizedMinImagePrototype (MinImg * *pDst*, const MinImg * *pSrc*, int *width*, int *height*, AllocationOption *allocation*)

Parameters

pDst	The destination image.
pSrc	The source image.
width	The required width of the destination image.
height	The required height of the destination image.
allocation	Specifies whether the destination image should be allocated.

Returns

NO_ERRORS on success or an error code otherwise (see MinErr).

The function makes a full copy of the image header with required size. If allocation is set to AO_PREALLOCATED (the default) then a new image will also allocated.

4.1.1.5 int CloneRetypifiedMinImagePrototype (MinImg * *pDst*, const MinImg * *pSrc*, MinTyp *type*, AllocationOption *allocation*)

Parameters

pDst	The destination image.
pSrc	The source image.
type	The required type of the destination image.
allocation	Specifies whether the destination image should be allocated.

Returns

NO_ERRORS on success or an error code otherwise (see MinErr).

The function makes a full copy of the image header with required type. If allocation is set to AO_PREALLOCATED (the default) then a new image will also allocated.

4.1.1.6 int CloneTransposedMinImagePrototype (MinImg * *pDst*, const MinImg * *pSrc*, AllocationOption *allocation*)

Parameters

pDst	The destination image.
pSrc	The source image.
allocation	Specifies whether the destination image should be allocated.

Returns

`NO_ERRORS` on success or an error code otherwise (see `MinErr`).

The function makes a full copy of the transposed image header (that is, `pDst->width = pSrc->height` and `pDst->height = pSrc->width`). If `allocation` is set to `AO_PREALLOCATED` (the default) then a new image will also allocated.

4.1.1.7 int CompareMinImagePrototypes (const MinImg * *pDst*, const MinImg * *pSrc*)

Parameters

pDst	The destination image.
pSrc	The source image.

Returns

Zero if the headers equals or a positive value otherwise.

The function just compare the header information (that is width, height, number of channels, channel depth and format) of the source and destination images. It does not matter whether the images are allocated or not.

4.1.1.8 int CopyMinImage (const MinImg * *pDst*, const MinImg * *pSrc*)

Parameters

pDst	The destination image.
pSrc	The source image.

Returns

`NO_ERRORS` on success or an error code otherwise (see `MinErr`).

Remarks

The destination image must be already allocated.
Both source and destination images must have the same size, the same format, and the same number of channels.

The function copies all elements from the source image to the destination one:

$$pDst(i,j) = pSrc(i,j)$$

4.1.1.9 int CopyMinImageChannels (const MinImg * *pDst*, const MinImg * *pSrc*, const int * *pDstChnls*, const int * *pSrcChnls*, int *nChnls*)

Parameters

pDst	The destination image.
pSrc	The source image.
pDstChnls	0-based destination channel indices.
pSrcChnls	0-based source channel indices.
nChnls	The number of channels to copy.

Returns

NO_ERRORS on success or an error code otherwise (see MinErr).

Remarks

The destination image must be already allocated.
Both source and destination images must have the same size and the same format.

The function copies the specified channels of the source image to the destination one.

4.1.1.10 int FillMinImage (const MinImg * *pImg*, const void * *pCanvasValue*, int *nValueLength*)

Parameters

pImg	The input image.
pCanvas-Value	The pointer to the fill value.
nValue-Length	The size of the fill value. If it is equal to zero then pixel size is used.

Returns

NO_ERRORS on success or an error code otherwise (see MinErr).

Remarks

The input image must be already allocated.

The function copies the scalar value to every element of the input image:

$$pImg(i, j) = pPixel$$

4.1.1.11 int FlipMinImage (const MinImg * *pDst*, const MinImg * *pSrc*, DirectionOption *dir*)

Parameters

pDst	The destination image.
pSrc	The source image
dir	Specifies how to flip the image (see DirectionOption).

Returns

NO_ERRORS on success or an error code otherwise (see MinErr).

Remarks

The destination image must be already allocated.
Both source and destination images must have the same size, the same format, and the same number of channels.

The function flips the image around vertical or horizontal axis. That is $pDst(i,j) = pSrc(pSrc->height - i - 1, j)$ for vertical flipping and $pDst(i,j) = pSrc(i, pSrc->width - j - 1)$ for horizontal flipping.

4.1.1.12 int FreeMinImage (MinImg * *pImg*)

Parameters

pImg	The image to be deallocated.

Returns

NO_ERRORS on success or an error code otherwise (see MinErr).

The function deallocates the image data and clean pImg->pScan0 and pImg->stride fields.

4.1.1.13 void* GetMinImageLine (const MinImg * *pImg*, int *y*, BorderOption *border*, void * *pCanvasLine*)

Parameters

pImg	The input image.
y	0-based line index.
border	The border condition (see BorderOption).
pCanvas-Line	The line to be used if the border is BO_CONSTANT.

Returns

A pointer to the specified line on success or NULL otherwise.

The function returns a pointer to the specified image line. If the y is out of the range then the function will return the pointer in accordance with the specified border condition (see BorderOption).

4.1.1.14 int GetMinImageRegion (MinImg * *pDst*, const MinImg * *pSrc*, int *x0*, int *y0*, int *width*, int *height*, RulesOption *rules*)

Parameters

pDst	The destination image.
pSrc	The source image.
x0	The x-coordinate of the top-left corner of the region.
y0	The y-coordinate of the top-left corner of the region.
width	The width of the region.
height	The height of the region.
rules	The degree of validation.

Returns

`NO_ERRORS` on success or an error code otherwise (see `MinErr`).

The function get a subimage from the source image. Note, that the function **does not** makes a copy of the specified region. Therefore, **it is strongly forbidden** to call `FreeMinImage()` for the `pDst`.

4.1.1.15 int GetMinImageType (const MinImg * *pImg*)

Parameters

pImg	The input image.

Returns

Appropriate image type or an error code otherwise (see `MinErr`).

The function analyze `pImg->format` and `pImg->channelDepth` fields and returns the type of the input image elements (see `MinTyp`).

4.1.1.16 int RotateMinImageBy90 (const MinImg * *pDst*, const MinImg * *pSrc*, int *nRotations*)

Parameters

pDst	The destination image.
pSrc	The source image.
nRotations	The multiplication factor.

Returns

`NO_ERRORS` on success or an error code otherwise (see `MinErr`).

Remarks

The destination image must be already allocated.
Both source and destination images must have the same format and the same number of channels.

The function rotates the image `pSrc` clockwise by `nRotation*90` degrees.

4.1.1.17 int SetMinImageType (MinImg * *pImg*, MinTyp *type*)

Parameters

pImg	The input image.
type	New image element type.

Returns

`NO_ERRORS` on success or an error code otherwise (see `MinErr`).

The function updates `pImg->format` and `pImg->channelDepth` field values according to assignable image element type.

4.1.1.18 int TransposeMinImage (const MinImg * *pDst*, const MinImg * *pSrc*)

Parameters

pDst	The destination image.
pSrc	The source image.

Returns

`NO_ERRORS` on success or an error code otherwise (see `MinErr`).

Remarks

The destination image must be already allocated.
Both source and destination images must have the same format and the same number of channels.

The function transpose the source image:

$$pDst(i,j) = pSrc(j,i)$$

4.1.1.19 int VerFlipMinImageRegion (MinImg * *pDst*, const MinImg * *pSrc*)

Parameters

pDst	The destination image.
pSrc	The source image.

Returns

NO_ERRORS on success or an error code otherwise (see MinErr).

This function flips the source image in vertical direction. Note, that the function **does not** makes a copy of the specified region. Therefore, **it is strongly forbidden** to call FreeMinImage() for the pDst.

4.2 MinImgAPI Library Utility

This section describes different utility functions and classes.

Data Structures

- class imgGuard

  Specifies a class which is used to avoid "free image" problems. More...

4.2.1 Data Structure Documentation

4.2.1.1 class imgGuard

Definition at line 49 of file imgguard.hpp.

Public Member Functions

- imgGuard (const MinImg &image)

  Constructor. Setups the image.

Private Member Functions

- imgGuard (const imgGuard &)

  Forbidden copy constructor.
- imgGuard & operator= (const imgGuard &)

  Forbidden assignment operator.

Private Attributes

- MinImg image

  The image to be freed while a function exit.

4.3 Possible Return Codes

The module specifies the return values used thought the library. Every function in the library follows the rule: it returns integer value. Meanwhile, a nonnegative return value indicates that the function completed successfully whereas a negative value indicates erroneous execution and specifies the error code. The enum `MinErr` contains codes for the most common errors. It is convenient enough to use special defines for handling return codes. Below you can find two defines which are widely used in the library.

Defines

- #define PROPAGATE_ERROR(call)

  If function failed then propagate the error code.
- #define SHOULD_WORK(call)

  If function failed then propagate INTERNAL_ERROR.

Enumerations

- enum MinErr

  Specifies basic error codes.

4.3.1 Define Documentation

4.3.1.1 #define PROPAGATE_ERROR(*call*)

Value:

```
{ \
  int res = call; \
  if (res < 0) \
    return res; \
}
```

This define macro describes a code that helps to propagate an exception if an error occurs.

Definition at line 85 of file minerr.h.

4.3.1.2 #define SHOULD_WORK(*call*)

Value:

```
{ \
  int res = call; \
  if (res < 0) \
    return INTERNAL_ERROR; \
}
```

This define macro describes a code that propagate INTERNAL_ERROR exception if an error occurs.

Definition at line 98 of file minerr.h.

4.3.2 Enumeration Type Documentation

4.3.2.1 enum MinErr

The enum specifies a list of basic error codes that is such ones which can be returned by any function in the library.

Enumerator:

NO_ERRORS No error has occurred. It indicates that the function completed successfully.

BAD_ARGS This error indicates that one or more arguments passed to the function are not correct.

NO_MEMORY Not enough memory is available. This can result from low memory conditions.

NOT_IMPLEMENTED This error indicates that the requested function is not implemented.

INTERNAL_ERROR An internal error has occurred. This error indicates that something went wrong.

FILE_ERROR An error occurred while working with files. The most likely cause is a full disk or a corrupted file to be open.

Definition at line 62 of file minerr.h.

4.4 Image Representation

The module specifies the image representation format. Every function in the library expects an input image in the form of `MinImg` object. `MinImg` is a cross-platform open-source container. The advantages of this container are the using minimal number of fields needed to represent the bitmap image and the easy way to cast it to other standard and popular view (for instance, Windows DIB, GDI+ BitmapData, Intel/OpenCV IplImage).

Data Structures

- struct MinImg

  A low-level universal representation of a bitmap image. More...
- struct float16

  Specifies half-precision floating point. More...

Enumerations

- enum MinFmt

  Specifies acceptable element formats of each individual channel.
- enum MinTyp

  Specifies acceptable element types of each individual channel.

4.4.1 Data Structure Documentation

4.4.1.1 struct MinImg

The struct `MinImg` represents a 2D dense numerical with additional fields needed for image representations (format and channel number). The struct `MinImg` allows to describe single-channel and multi- channel images in a wide range of different image types. Herewith, the format of the image is specified by two values: depth of the channel (see `MinImg::channelDepth`) and channel element format (see `MinImg::format`). To represent a binary image you should set `MinImg::format` to `FMT_UINT` and `MinImg::channelDepth` to 0.

Definition at line 69 of file minimg.h.

Data Fields

- int32_t width

  The image width in pixels. It must be positive.
- int32_t height

  The image height in pixels. It must be positive.
- int32_t stride

  The width of a single row of pixels in bytes.
- int32_t channels

The number of channels per pixel. It must be positive.

- int32_t channelDepth

  The channel depth in bytes. It must be nonnegative.

- MinFmt format

  The channel element format (see `MinFmt`).

- uint8_t * pScan0

  The pointer to the first pixel of the first row.

4.4.1.2 struct float16

The struct `float16` represents half-precision floating point.

Definition at line 90 of file mintyp.h.

Data Fields

- uint16_t significand: 10

  The mantissa of the number.

- uint16_t exponent: 5

  The magnitude of the number.

- uint16_t sign: 1

  The sing of the number.

4.4.2 Enumeration Type Documentation

4.4.2.1 enum MinFmt

The enum specifies acceptable element formats of each individual channel.

Enumerator:

FMT_UINT Unsigned integer.

FMT_INT Signed integer.

FMT_FLOAT Floating point.

Definition at line 56 of file mintyp.h.

4.4.2.2 enum MinTyp

The enum specifies acceptable element types (that is format + size) of each individual channel.

Enumerator:

TYP_UINT1 1-bit logical.

TYP_UINT8 Unsigned 8-bit integer.

TYP_INT8 Signed 8-bit integer.

TYP_UINT16 Unsigned 16-bit integer.

TYP_INT16 Signed 16-bit integer.

TYP_FLOAT16 Half-precision floating point.

TYP_UINT32 Unsigned 32-bit integer.

TYP_INT32 Signed 32-bit integer.

TYP_FLOAT32 Single-precision floating point.

TYP_UINT64 Unsigned 64-bit integer.

TYP_INT64 Signed 64-bit integer.

TYP_FLOAT64 Double-precision floating point.

Definition at line 69 of file mintyp.h.

4.5 Miscellaneous Options

The module contains miscellaneous options used throughout the library.

Enumerations

- enum AllocationOption

  Specifies allocation options.
- enum BorderOption

  Specifies border acceptable border conditions.
- enum DirectionOption

  Specifies acceptable directions.
- enum QualityOption

  Specifies interpolation methods.
- enum RulesOption

  Specifies the degree of rules validation.

4.5.1 Enumeration Type Documentation

4.5.1.1 enum AllocationOption

The enum specifies whether the new object should be allocated. This is used in various create- and clone-functions.

Enumerator:

AO_EMPTY The object should stay empty (without allocation).

AO_PREALLOCATED The object should be allocated.

Definition at line 56 of file minopt.h.

4.5.1.2 enum BorderOption

The enum specifies acceptable options for border condition. If a function needs pixels outside of an image, then they are reconstructed according to one the following modes (that is, fill the "image border").

Enumerator:

BO_IGNORE Ignores the image size and allows out of memory reading.

BO_REPEAT The bounding pixels are replicated to fill the border.

BO_SYMMETRIC The border is filled with the mirror-imaged pixels.

BO_CYCLIC Fill the border using a cyclic repetition of image pixels.

BO_CONSTANT The border is filled with the fixed value.

BO_VOID Fill the border with void pixels.

Definition at line 69 of file minopt.h.

4.5.1.3 enum DirectionOption

The enum specifies directions which can be used in image transformation, image filtration, calculation orientation and other functions.

Enumerator:

DO_VERTICAL Vertical transformation.

DO_HORIZONTAL Horizontal transformation.

Definition at line 85 of file minopt.h.

4.5.1.4 enum QualityOption

The enum specifies acceptable interpolation methods. This can be used in the the case when pixel values at fractional coordinates needs to be retrieved (for example, affine or projective transformation functions).

Enumerator:

QO_PIXEL Nearest-neighbor interpolation.

QO_SUBPIXEL Bilinear interpolation.

Definition at line 98 of file minopt.h.

4.5.1.5 enum RulesOption

The enum specifies the degree of rules validation. This can be used, for example, to choose a proper way of input arguments validation.

Enumerator:

RO_WEAK Miss uncritical validations.

RO_STRICT Validate each rule in a proper way.

Definition at line 110 of file minopt.h.

Data Structure Documentation

5 File Documentation

5.1 imgguard.hpp File Reference

Definition of utility classes.

Data Structures

- class imgGuard

  Specifies a class which is used to avoid "free image" problems. More...

5.1.1 Detailed Description

Definition in file imgguard.hpp.

5.2 imgguard.hpp

```
00001 /*
00002
00003 Copyright (c) 2011, Smart Engines Limited. All rights reserved.
00004
00005 All rights reserved.
00006
00007 Redistribution and use in source and binary forms, with or without
       modification, are
00008 permitted provided that the following conditions are met:
00009
00010    1. Redistributions of source code must retain the above copyright notice,
       this list of
00011       conditions and the following disclaimer.
00012
00013    2. Redistributions in binary form must reproduce the above copyright notice,
       this list
00014       of conditions and the following disclaimer in the documentation and/or
       other materials
00015       provided with the distribution.
00016
00017 THIS SOFTWARE IS PROVIDED BY COPYRIGHT HOLDERS ``AS IS'' AND ANY EXPRESS OR
       IMPLIED
00018 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
       MERCHANTABILITY AND
00019 FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL COPYRIGHT
       HOLDERS OR
00020 CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
       EXEMPLARY, OR
00021 CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
       GOODS OR
00022 SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
       CAUSED AND ON
00023 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
       (INCLUDING
00024 NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
       EVEN IF
00025 ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00026
00027 The views and conclusions contained in the software and documentation are those
```

```
       of the
00028 authors and should not be interpreted as representing official policies, either
       expressed
00029 or implied, of copyright holders.
00030
00031 */
00032
00038 #pragma once
00039
00040 #ifndef IMGGUARD_INCLUDED
00041 #define IMGGUARD_INCLUDED
00042
00043 #include <minimgapi/minimgapi.h>
00044
00049 class imgGuard
00050 {
00051 public:
00052   imgGuard(const MinImg &image) : image(image) {}
00053   virtual ~imgGuard()
00054   {
00055     FreeMinImage(&image);
00056   }
00057 private:
00058   imgGuard(const imgGuard &);
00059   imgGuard &operator =(const imgGuard &);
00060 private:
00061   MinImg image;
00062 };
00063
00064 #endif  /* IMGGUARD_INCLUDED */
```

5.3 minimgapi.h File Reference

MinImgAPI library application programming interface.

Defines

- #define IS_BY_DEFAULT(a)

  Specifies a default value for a parameter of a function.
- #define MINIMGAPI_API

  Specifies storage-class information (only for MSC).

Functions

- int AllocMinImage (MinImg *pImg, int alignment)

  Allocates an image.
- int FreeMinImage (MinImg *pImg)

  Deallocates an image.
- int GetMinImageType (const MinImg *pImg)

  Returns type of an image channel element.
- int SetMinImageType (MinImg *pImg, MinTyp type)

  Assigns type to the image.
- int CopyMinImage (const MinImg *pDst, const MinImg *pSrc)

  Copies one image to another.

- int FlipMinImage (const MinImg *pDst, const MinImg *pSrc, DirectionOption dir)

  Flips an image around vertical or horizontal axis.
- int TransposeMinImage (const MinImg *pDst, const MinImg *pSrc)

  Transposes an image.
- int RotateMinImageBy90 (const MinImg *pDst, const MinImg *pSrc, int nRotations)

  Rotates an image by 90 degrees (clockwise).
- int FillMinImage (const MinImg *pImg, const void *pCanvasValue, int nValueLength)

  Sets every element of an image to a given value.
- int CopyMinImageChannels (const MinImg *pDst, const MinImg *pSrc, const int *pDstChnls, const int *pSrcChnls, int nChnls)

  Copies specified channels of an image to another one.
- int CloneMinImagePrototype (MinImg *pDst, const MinImg *pSrc, AllocationOption allocation)

  Makes a copy of the image header.
- int CloneTransposedMinImagePrototype (MinImg *pDst, const MinImg *pSrc, AllocationOption allocation)

  Makes a copy of the transposed image header.
- int CloneRetypifiedMinImagePrototype (MinImg *pDst, const MinImg *pSrc, MinTyp type, AllocationOption allocation)

  Makes a copy of the image header with another type.
- int CloneDimensionedMinImagePrototype (MinImg *pDst, const MinImg *pSrc, int channels, AllocationOption allocation)

  Makes a copy of the image header with another number of channels.
- int CloneResizedMinImagePrototype (MinImg *pDst, const MinImg *pSrc, int width, int height, AllocationOption allocation)

  Makes a copy of the image header with another size.
- void * GetMinImageLine (const MinImg *pImg, int y, BorderOption border, void *pCanvasLine)

  Returns a pointer to the specified image line.
- int GetMinImageRegion (MinImg *pDst, const MinImg *pSrc, int x0, int y0, int width, int height, RulesOption rules)

  Gets a region of an image.
- int VerFlipMinImageRegion (MinImg *pDst, const MinImg *pSrc)

  Flips an image in vertical without copying.
- int CompareMinImagePrototypes (const MinImg *pDst, const MinImg *pSrc)

  Compares headers of two images.

5.3.1 Detailed Description

Definition in file minimgapi.h.

5.4 minimgapi.h

```
00001 /*
00002
00003 Copyright (c) 2011, Smart Engines Limited. All rights reserved.
00004
00005 All rights reserved.
00006
00007 Redistribution and use in source and binary forms, with or without
       modification,
00008 are permitted provided that the following conditions are met:
00009
00010    1. Redistributions of source code must retain the above copyright notice,
00011       this list of conditions and the following disclaimer.
00012
00013    2. Redistributions in binary form must reproduce the above copyright notice,
00014       this list of conditions and the following disclaimer in the documentation
00015       and/or other materials provided with the distribution.
00016
00017 THIS SOFTWARE IS PROVIDED BY COPYRIGHT HOLDERS "AS IS" AND ANY EXPRESS OR
00018 IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
00019 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
       EVENT
00020 SHALL COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
00021 INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00022 LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00023 PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
00024 LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
00025 OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
00026 ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00027
00028 The views and conclusions contained in the software and documentation are those
00029 of the authors and should not be interpreted as representing official policies,
00030 either expressed or implied, of copyright holders.
00031
00032 */
00033
00039 #pragma once
00040
00041 #ifndef MINIMGAPI_H_INCLUDED
00042 #define MINIMGAPI_H_INCLUDED
00043
00044 #include <cstdlib>
00045 #include <minutils/minimg.h>
00046 #include <minutils/mintyp.h>
00047 #include <minutils/minopt.h>
00048
00137 #ifdef IS_BY_DEFAULT
00138 #  undef IS_BY_DEFAULT
00139 #endif
00140
00141 #ifdef __cplusplus
00142 #  define IS_BY_DEFAULT(a) = a
00143 extern "C" {
00144 #else
00145 #  define IS_BY_DEFAULT(a)
00146 #endif
00147
00154 #ifdef _MSC_VER
00155 #  ifdef MINIMGAPI_EXPORTS
00156 #    define MINIMGAPI_API __declspec(dllexport)
00157 #  else
00158 #    define MINIMGAPI_API __declspec(dllimport)
00159 #  endif
00160 #else
00161 #  define MINIMGAPI_API
00162 #endif
00163
00189 MINIMGAPI_API int AllocMinImage
00190 (
00191   MinImg *pImg,
00192   int     alignment IS_BY_DEFAULT(16)
00193 );
```

```
00194
00204 MINIMGAPI_API int FreeMinImage
00205 (
00206   MinImg *pImg
00207 );
00208
00218 MINIMGAPI_API int GetMinImageType
00219 (
00220   const MinImg *pImg
00221 );
00222
00233 MINIMGAPI_API int SetMinImageType
00234 (
00235   MinImg *pImg,
00236   MinTyp  type
00237 );
00238
00252 MINIMGAPI_API int CopyMinImage
00253 (
00254   const MinImg *pDst,
00255   const MinImg *pSrc
00256 );
00257
00273 MINIMGAPI_API int FlipMinImage
00274 (
00275   const MinImg   *pDst,
00276   const MinImg   *pSrc,
00277   DirectionOption dir
00278 );
00279
00292 MINIMGAPI_API int TransposeMinImage
00293 (
00294   const MinImg *pDst,
00295   const MinImg *pSrc
00296 );
00297
00311 MINIMGAPI_API int RotateMinImageBy90
00312 (
00313   const MinImg *pDst,
00314   const MinImg *pSrc,
00315   int           nRotations
00316 );
00317
00331 MINIMGAPI_API int FillMinImage
00332 (
00333   const MinImg *pImg,
00334   const void   *pCanvasValue,
00335   int           nValueLength IS_BY_DEFAULT(0)
00336 );
00337
00354 MINIMGAPI_API int CopyMinImageChannels
00355 (
00356   const MinImg *pDst,
00357   const MinImg *pSrc,
00358   const int    *pDstChnls,
00359   const int    *pSrcChnls,
00360   int           nChnls
00361 );
00362
00374 MINIMGAPI_API int CloneMinImagePrototype
00375 (
00376   MinImg          *pDst,
00377   const MinImg    *pSrc,
00378   AllocationOption allocation IS_BY_DEFAULT(AO_PREALLOCATED)
00379 );
00380
00394 MINIMGAPI_API int CloneTransposedMinImagePrototype
00395 (
00396   MinImg          *pDst,
00397   const MinImg    *pSrc,
00398   AllocationOption allocation IS_BY_DEFAULT(AO_PREALLOCATED)
00399 );
00400
00414 MINIMGAPI_API int CloneRetypifiedMinImagePrototype
```

```
00415 (
00416   MinImg          *pDst,
00417   const MinImg    *pSrc,
00418   MinTyp           type,
00419   AllocationOption allocation IS_BY_DEFAULT(AO_PREALLOCATED)
00420 );
00421
00435 MINIMGAPI_API int CloneDimensionedMinImagePrototype
00436 (
00437   MinImg          *pDst,
00438   const MinImg    *pSrc,
00439   int              channels,
00440   AllocationOption allocation IS_BY_DEFAULT(AO_PREALLOCATED)
00441 );
00442
00457 MINIMGAPI_API int CloneResizedMinImagePrototype
00458 (
00459   MinImg          *pDst,
00460   const MinImg    *pSrc,
00461   int              width,
00462   int              height,
00463   AllocationOption allocation IS_BY_DEFAULT(AO_PREALLOCATED)
00464 );
00465
00479 MINIMGAPI_API void *GetMinImageLine
00480 (
00481   const MinImg *pImg,
00482   int           y,
00483   BorderOption  border      IS_BY_DEFAULT(BO_VOID),
00484   void         *pCanvasLine IS_BY_DEFAULT(NULL)
00485 );
00486
00503 MINIMGAPI_API int GetMinImageRegion
00504 (
00505   MinImg       *pDst,
00506   const MinImg *pSrc,
00507   int           x0,
00508   int           y0,
00509   int           width,
00510   int           height,
00511   RulesOption   rules IS_BY_DEFAULT(RO_STRICT)
00512 );
00513
00525 MINIMGAPI_API int VerFlipMinImageRegion
00526 (
00527   MinImg       *pDst,
00528   const MinImg *pSrc
00529 );
00530
00542 MINIMGAPI_API int CompareMinImagePrototypes
00543 (
00544   const MinImg *pDst,
00545   const MinImg *pSrc
00546 );
00547
00548 #ifdef __cplusplus
00549 } // extern "C"
00550 #  undef IS_BY_DEFAULT
00551 #endif
00552
00553 #endif // MINIMGAPI_H_INCLUDED
```

5.5 minutils/minerr.h File Reference

Definition of possible return values.

Defines

- #define PROPAGATE_ERROR(call)

  If function failed then propagate the error code.
- #define SHOULD_WORK(call)

  If function failed then propagate INTERNAL_ERROR.

Enumerations

- enum MinErr

  Specifies basic error codes.

5.5.1 Detailed Description

Definition in file minerr.h.

5.6 minutils/minerr.h

```
00001 /*
00002
00003 Copyright (c) 2011, Smart Engines Limited. All rights reserved.
00004
00005 All rights reserved.
00006
00007 Redistribution and use in source and binary forms, with or without
       modification,
00008 are permitted provided that the following conditions are met:
00009
00010    1. Redistributions of source code must retain the above copyright notice,
00011       this list of conditions and the following disclaimer.
00012
00013    2. Redistributions in binary form must reproduce the above copyright notice,
00014       this list of conditions and the following disclaimer in the documentation
00015       and/or other materials provided with the distribution.
00016
00017 THIS SOFTWARE IS PROVIDED BY COPYRIGHT HOLDERS ``AS IS'' AND ANY EXPRESS OR
00018 IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
00019 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
       EVENT
00020 SHALL COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
00021 INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00022 LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00023 PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
00024 LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
00025 OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
00026 ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00027
00028 The views and conclusions contained in the software and documentation are those
00029 of the authors and should not be interpreted as representing official policies,
00030 either expressed or implied, of copyright holders.
00031
00032 */
00033
00039 #pragma once
00040
00041 #ifndef MINERR_H_INCLUDED
00042 #define MINERR_H_INCLUDED
00043
00062 typedef enum
00063 {
00064   NO_ERRORS       =  0,
```

```
00065
00066   BAD_ARGS        = -1,
00067
00068   NO_MEMORY       = -2,
00069
00070   NOT_IMPLEMENTED = -3,
00071
00072   INTERNAL_ERROR  = -4,
00073
00074   FILE_ERROR      = -5
00075
00076
00077 } MinErr;
00078
00085 #define PROPAGATE_ERROR(call) \
00086 { \
00087   int res = call; \
00088   if (res < 0) \
00089     return res; \
00090 }
00091
00098 #define SHOULD_WORK(call) \
00099 { \
00100   int res = call; \
00101   if (res < 0) \
00102     return INTERNAL_ERROR; \
00103 }
00104
00105 #endif  /* MINERR_H_INCLUDED */
```

5.7 minutils/minimg.h File Reference

Definition of a low-level representation of a bitmap image.

Data Structures

- struct MinImg

  A low-level universal representation of a bitmap image. More...

5.7.1 Detailed Description

Definition in file minimg.h.

5.8 minutils/minimg.h

```
00001 /*
00002
00003 Copyright (c) 2011, Smart Engines Limited. All rights reserved.
00004
00005 All rights reserved.
00006
00007 Redistribution and use in source and binary forms, with or without
       modification,
00008 are permitted provided that the following conditions are met:
00009
00010    1. Redistributions of source code must retain the above copyright notice,
00011       this list of conditions and the following disclaimer.
00012
00013    2. Redistributions in binary form must reproduce the above copyright notice,
00014       this list of conditions and the following disclaimer in the documentation
00015       and/or other materials provided with the distribution.
```

```
00016
00017 THIS SOFTWARE IS PROVIDED BY COPYRIGHT HOLDERS "AS IS" AND ANY EXPRESS OR
00018 IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
00019 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
       EVENT
00020 SHALL COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
00021 INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00022 LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00023 PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
00024 LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
00025 OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
00026 ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00027
00028 The views and conclusions contained in the software and documentation are those
00029 of the authors and should not be interpreted as representing official policies,
00030 either expressed or implied, of copyright holders.
00031
00032 */
00033
00039 #pragma once
00040
00041 #ifndef MINIMG_H_INCLUDED
00042 #define MINIMG_H_INCLUDED
00043
00044 #include <minutils/mintyp.h>
00045
00069 typedef struct
00070 {
00071   int32_t  width;
00072   int32_t  height;
00073   int32_t  stride;
00074   int32_t  channels;
00075   int32_t  channelDepth;
00076   MinFmt   format;
00077   uint8_t *pScan0;
00078 } MinImg;
00079
00080 #endif  /* MINIMG_H_INCLUDED */
```

5.9 minutils/minopt.h File Reference

Definition of various options.

Enumerations

- enum AllocationOption

  Specifies allocation options.
- enum BorderOption

  Specifies border acceptable border conditions.
- enum DirectionOption

  Specifies acceptable directions.
- enum QualityOption

  Specifies interpolation methods.
- enum RulesOption

  Specifies the degree of rules validation.

5.9.1 Detailed Description

Definition in file minopt.h.

5.10 minutils/minopt.h

```
00001 /*
00002
00003 Copyright (c) 2011, Smart Engines Limited. All rights reserved.
00004
00005 All rights reserved.
00006
00007 Redistribution and use in source and binary forms, with or without
       modification,
00008 are permitted provided that the following conditions are met:
00009
00010    1. Redistributions of source code must retain the above copyright notice,
00011       this list of conditions and the following disclaimer.
00012
00013    2. Redistributions in binary form must reproduce the above copyright notice,
00014       this list of conditions and the following disclaimer in the documentation
00015       and/or other materials provided with the distribution.
00016
00017 THIS SOFTWARE IS PROVIDED BY COPYRIGHT HOLDERS "AS IS" AND ANY EXPRESS OR
00018 IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
00019 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
       EVENT
00020 SHALL COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
00021 INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00022 LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00023 PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
00024 LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
00025 OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
00026 ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00027
00028 The views and conclusions contained in the software and documentation are those
00029 of the authors and should not be interpreted as representing official policies,
00030 either expressed or implied, of copyright holders.
00031
00032 */
00033
00039 #pragma once
00040
00041 #ifndef MINUTILS_MINOPT_H_INCLUDED
00042 #define MINUTILS_MINOPT_H_INCLUDED
00043
00056 typedef enum
00057 {
00058   AO_EMPTY,
00059   AO_PREALLOCATED
00060 } AllocationOption;
00061
00069 typedef enum
00070 {
00071   BO_IGNORE,
00072   BO_REPEAT,
00073   BO_SYMMETRIC,
00074   BO_CYCLIC,
00075   BO_CONSTANT,
00076   BO_VOID
00077 } BorderOption;
00078
00085 typedef enum
00086 {
00087   DO_VERTICAL,
00088   DO_HORIZONTAL
00089 } DirectionOption;
00090
00098 typedef enum
00099 {
00100   QO_PIXEL,
00101   QO_SUBPIXEL
00102 } QualityOption;
00103
00110 typedef enum
00111 {
00112   RO_WEAK,
```

```
00113   RO_STRICT
00114 } RulesOption;
00115
00116 #endif // MINUTILS_MINOPT_H_INCLUDED
```

5.11 minutils/mintyp.h File Reference

Definition of acceptable image types.

Data Structures

- struct float16

  Specifies half-precision floating point. More...

Typedefs

- typedef float16 float16_t

  Specifies `float16` as `float16_t`.
- typedef float float32_t

  Specifies `float` as `float32_t` type.
- typedef double float64_t

  Specifies `double` as `float64_t` type.

Enumerations

- enum MinFmt

  Specifies acceptable element formats of each individual channel.
- enum MinTyp

  Specifies acceptable element types of each individual channel.

5.11.1 Detailed Description

Definition in file mintyp.h.

5.12 minutils/mintyp.h

```
00001 /*
00002
00003 Copyright (c) 2011, Smart Engines Limited. All rights reserved.
00004
00005 All rights reserved.
00006
00007 Redistribution and use in source and binary forms, with or without
       modification,
00008 are permitted provided that the following conditions are met:
00009
00010    1. Redistributions of source code must retain the above copyright notice,
00011       this list of conditions and the following disclaimer.
00012
```

```
00013    2. Redistributions in binary form must reproduce the above copyright notice,
00014       this list of conditions and the following disclaimer in the documentation
00015       and/or other materials provided with the distribution.
00016
00017 THIS SOFTWARE IS PROVIDED BY COPYRIGHT HOLDERS ``AS IS'' AND ANY EXPRESS OR
00018 IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
00019 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
      EVENT
00020 SHALL COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
00021 INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00022 LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00023 PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
00024 LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
00025 OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
00026 ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00027
00028 The views and conclusions contained in the software and documentation are those
00029 of the authors and should not be interpreted as representing official policies,
00030 either expressed or implied, of copyright holders.
00031
00032 */
00033
00039 #pragma once
00040
00041 #ifndef MINTYP_H_INCLUDED
00042 #define MINTYP_H_INCLUDED
00043
00044 #ifdef _MSC_VER
00045 #include <minutils/stdint-vc.h>
00046 #else
00047 #include <stdint.h>
00048 #endif // _MSC_VER
00049
00056 typedef enum
00057 {
00058   FMT_UINT,
00059   FMT_INT,
00060   FMT_FLOAT
00061 } MinFmt;
00062
00069 typedef enum
00070 {
00071   TYP_UINT1,
00072   TYP_UINT8,
00073   TYP_INT8,
00074   TYP_UINT16,
00075   TYP_INT16,
00076   TYP_FLOAT16,
00077   TYP_UINT32,
00078   TYP_INT32,
00079   TYP_FLOAT32,
00080   TYP_UINT64,
00081   TYP_INT64,
00082   TYP_FLOAT64
00083 } MinTyp;
00084
00090 typedef struct
00091 {
00092   uint16_t significand : 10;
00093   uint16_t exponent    : 5;
00094   uint16_t sign        : 1;
00095 } float16;
00096
00097 typedef float16 float16_t;
00098 typedef float   float32_t;
00099 typedef double  float64_t;
00100
00101 #endif  /* MINTYP_H_INCLUDED */
```