MinUtils Library Reference

version 1.0

Generated by Doxygen 1.7.5.1

Fri Oct 7 2011 20:44:52

Contents

1 Overview

MinUtils is an open-source platform-independent library that contains the definition of image representation container MinImg and a set of auxiliary miscellaneous entities such as a list of possible return values, basic geometry data structures, smart pointers, and others.

The **MinUtils** library is header based, consisting of definitions, structures, classes with inline members, inline functions and templates, and as such do not need to be built in advance of its use.

The library is written in C++ and can be compiled under Linux (GCC) and Windows (MSVC 8 and later).

2 MinUtils License Agreements

2.1 Library License Agreement

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

2.2 Documentation License Agreement

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

3 Module Documentation

3.1 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.

3.1.1 Data Structure Documentation

3.1.1.1 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.

3.1.1.2 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.

3.1.2 Enumeration Type Documentation

3.1.2.1 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.

3.1.2.2 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.

3.2 Multi-channel Array Representation

The module specifies a multi-dimensional dense multi-channel array representation.

Data Structures

- struct MinArr

  A multi-dimensional dense multi-channel array representation. More...

3.2.1 Data Structure Documentation

3.2.1.1 struct MinArr

The struct `MinArr` represents a multi-dimensional dense numerical single-channel or multi-channel array. The data layout of the array is defined by the field `MinArr::pStrides`. Let us M is an instance of `MinArr`. So the address of element $(i_0,...,i_{M.dim-1})$, where $0 \leq i_k \leq M.pSizes[k]$ is computed as:

$$M.pStart + M.pStrides[0] * i_0 + ... + M.pStrides[M.dim-1] * i_{M.dim-1}$$

For example, in the case of 2-dimensional array the above formula is reduced to:

$$M.pStart + M.pStrides[0] * i + M.pStrides[1] * j$$

Note that $M.pStrides[i] \geq M.pStrides[i+1]$ that is, 2-dimensional matrices are stored row-by-row, 3-dimensional matrices are stored plane-by plane etc. $M.pStrides[M.dim-1]$ is minimal and always equal to the element size $M.channelDepth$.

Definition at line 70 of file minarr.h.

Data Fields

- int32_t dim

  The number of array dimensions. It must be positive.
- int32_t * pSizes

  Array size of each dimension. It must be nonnegative.
- int32_t * pStrides

  Full row length (in bytes) for each dimension. It must be positive.
- int32_t channelDepth

  Size of element in bytes. It must be positive.
- MinFmt format

  Element format (supported formats are represented by MinFmt).
- uint8_t * pStart

  The pointer to the (0, 0, ... 0) element.

3.3 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.

3.3.1 Define Documentation

3.3.1.1 #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.

3.3.1.2 #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.

3.3.2 Enumeration Type Documentation

3.3.2.1 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.

3.4 Basic Geometry Data Structures

The module specifies basic geometry data structures and auxiliary inline functions using for convenient construction of them.

Data Structures

- struct MinPoint

  2D point with integer coordinates (zero-based). More...
- struct MinSize

  Size of an image or a rectangle. More...
- struct MinRect

  Offset and size of a rectangle. More...

Functions

- static MinPoint minPoint (int32_t x, int32_t y)

  Inline constructors for `MinPoint` data type.
- static MinSize minSize (int32_t width, int32_t height)

  Inline constructors for `MinSize` data type.
- static MinRect minRect (int32_t x, int32_t y, int32_t width, int32_t height)

  Inline constructors for `MinRect` data type.

3.4.1 Data Structure Documentation

3.4.1.1 struct MinPoint

The structure `MinPoint` represents a 2D point as a tuple (x, y), where x and y are integers.

Definition at line 61 of file mingeo.h.

Data Fields

- int32_t x

  The x-coordinate of the point.
- int32_t y

  The y-coordinate of the point.

3.4.1.2 struct MinSize

The struct `MinSize` represents a size of an image or a rectangle as a tuple (width, height), where width and height are integers.

Definition at line 73 of file mingeo.h.

Data Fields

- int32_t width

  The width of the rectangle or the image.
- int32_t height

  The height of the rectangle or the image.

3.4.1.3 struct MinRect

The struct `MinRect` represents a rectangle as a tuple (x, y, width, height), where all are integers.

Definition at line 85 of file mingeo.h.

Data Fields

- int32_t x

  The x-coordinate of the top-left corner.
- int32_t y

  The y-coordinate of the top-left corner.
- int32_t width

  The width of the rectangle.
- int32_t height

  The height of the rectangle.

3.4.2 Function Documentation

3.4.2.1 static MinPoint minPoint (int32_t *x*, int32_t *y*) [inline, static]

Parameters

x	The x-coordinate of the point.
y	The y-coordinate of the point.

Returns

A new `MinPoint` object.

The function constructs a new `MinPoint` object using x-coordinate and y-coordinate values.

Definition at line 103 of file mingeo.h.

3.4.2.2 static MinRect minRect (int32_t *x*, int32_t *y*, int32_t *width*, int32_t *height*) [inline, static]

Parameters

x	The x-coordinate of the top-left corner.
y	The x-coordinate of the top-left corner.
width	The width of the rectangle.
height	The height of the rectangle.

Returns

A new `MinRect` object.

The function constructs a new `MinRect` object using x-coordinate, y-coordinate, width, and height values.

Definition at line 137 of file mingeo.h.

3.4.2.3 static MinSize minSize (int32_t *width*, int32_t *height*) [inline, static]

Parameters

width	The width of the size object.
height	The height of the size object.

Returns

A new `MinSize` object.

The function constructs a new `MinSize` object using width and height values.

Definition at line 119 of file mingeo.h.

3.5 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.

3.5.1 Enumeration Type Documentation

3.5.1.1 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.

3.5.1.2 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.

3.5.1.3 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.

3.5.1.4 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.

3.5.1.5 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.

3.6 Mathematical operations

The module specifies mathematical operations which can be used in image processing functions. All operations are specified by the follow constants: OP_MIN (binary minimum), OP_MAX (binary maximum), OP_ADD (binary addition), OP_DIF (binary difference), OP_ADF (binary absolute difference), OP_MUL (binary multiplication), OP_AVE (binary average), and OP_EUC (binary Euclidean norm). Additionally, the operations are grouped into several enums in accordance with their properties: all binary operations (BiOp), associative operations (AsOp), commutative operations (CoOp), associative-commutative operations (AsCoOp) and idempotent operations (IdOp).

Enumerations

- enum BiOp

  Specifies binary operations.
- enum AsOp

  Specifies associative operations.
- enum CoOp

  Specifies commutative operations.
- enum AsCoOp

  Specifies associative-commutative operations.
- enum IdOp

  Specifies idempotent operations.

Variables

- const int OP_MIN = 1

  Specifies binary minimum operation.
- const int OP_MAX = 2

  Specifies binary maximum operation.
- const int OP_ADD = 3

  Specifies binary addition operation.
- const int OP_DIF = 4

  Specifies binary difference operation.
- const int OP_ADF = 5

  Specifies binary absolute difference operation.
- const int OP_MUL = 6

  Specifies binary multiplication operation.
- const int OP_AVE = 7

  Specifies binary average operation.
- const int OP_EUC = 8

  Specifies binary Euclidean norm operation.

3.6.1 Enumeration Type Documentation

3.6.1.1 enum AsCoOp

The enum specifies associative-commutative operations, that is such ones which have both associative and commutative properties. Formally, a binary operation $\circ$ on a set S is called associative-commutative if it satisfies both the associative and the commutative laws:

$$f(f(x,y),z) = f(x,f(y,z)) \quad \forall x,y,z \in S$$
$$f(x,y) = f(y,x) \quad \forall x,y \in S$$

Enumerator:

ASCOOP_MIN Binary minimum operation (see OP_MIN).

ASCOOP_MAX Binary maximum operation (see OP_MAX).

ASCOOP_ADD Binary addition operation (see OP_ADD).

ASCOOP_MUL Binary multiplication operation (see OP_MUL).

ASCOOP_EUC Binary Euclidean norm operation (see OP_EUC).

Definition at line 192 of file mathoper.h.

3.6.1.2 enum AsOp

The enum specifies associative operations that is such ones which can be freely re-grouped without altering result. Formally, a binary operation f on a set S is called associative if it satisfies the associative law:

$$f(f(x,y),z) = f(x,f(y,z)) \quad \forall x,y,z \in S$$

Enumerator:

ASOP_MIN Binary minimum operation (see OP_MIN).

ASOP_MAX Binary maximum operation (see OP_MAX).

ASOP_ADD Binary addition operation (see OP_ADD).

ASOP_MUL Binary multiplication operation (see OP_MUL).

ASOP_EUC Binary Euclidean norm operation (see OP_EUC).

Definition at line 152 of file mathoper.h.

3.6.1.3 enum BiOp

The enum specifies binary operations, that is such ones which involve two operands. Formally, a binary operation f on a set S is a binary relation that maps elements of the Cartesian product $S \times S$ to S:

$$f : S \times S \to S$$

Enumerator:

BIOP_MIN Binary minimum operation (see OP_MIN).

BIOP_MAX Binary maximum operation (see OP_MAX).

BIOP_ADD Binary addition operation (see OP_ADD).

BIOP_DIF Binary difference operation (see OP_DIF).

BIOP_ADF Binary absolute difference operation (see OP_ADF).

BIOP_MUL Binary multiplication operation (see OP_MUL).

BIOP_AVE Binary average operation (see OP_AVE).

BIOP_EUC Binary Euclidean norm operation (see OP_EUC).

Definition at line 131 of file mathoper.h.

3.6.1.4 enum CoOp

The enum specifies commutative operations, that is such ones which do not depend on the order of the input parameters. Formally, a binary operation f on a set S is called commutative if it satisfies the commutative law:

$$f(x,y) = f(y,x) \quad \forall x,y \in S$$

Enumerator:

COOP_MIN Binary minimum operation (see OP_MIN).

COOP_MAX Binary maximum operation (see OP_MAX).

COOP_ADD Binary addition operation (see OP_ADD).

COOP_ADF Binary absolute difference operation (see OP_ADF).

COOP_MUL Binary multiplication operation (see OP_MUL).

COOP_AVE Binary average operation (see OP_AVE).

COOP_EUC Binary Euclidean norm operation (see OP_EUC).

Definition at line 170 of file mathoper.h.

3.6.1.5 enum IdOp

The enum specifies idempotent operations, that is such ones which can be applied multiple times without changing the result. Formally, a binary operation f on a set S is called idempotent if, whenever it is applied to two equal values $x \in S$, it gives that value as the result:

$$f(x,x) = x \quad \forall x \in S$$

Enumerator:

IDOP_MIN Binary minimum operation (see OP_MIN).

IDOP_MAX Binary maximum operation (see OP_MAX).

Definition at line 211 of file mathoper.h.

3.6.2 Variable Documentation

3.6.2.1 const int OP_ADD = 3

The constant specifies the binary addition operation f that is defined as follows:

$$f(x,y) = x+y$$

Definition at line 80 of file mathoper.h.

3.6.2.2 const int OP_ADF = 5

The constant specifies the binary absolute difference operation f that is defined as follows:

$$f(x,y) = \mid x-y \mid$$

Definition at line 96 of file mathoper.h.

3.6.2.3 const int OP_AVE = 7

The constant specifies the binary average operation f that is defined as follows:

$$f(x,y) = \frac{x+y}{2}$$

Definition at line 112 of file mathoper.h.

3.6.2.4 const int OP_DIF = 4

The constant specifies the binary difference operation f that is defined as follows:

$$f(x,y) = x-y$$

Definition at line 88 of file mathoper.h.

3.6.2.5 const int OP_EUC = 8

The constant specifies the binary Euclidean norm operation f that is defined as follows:

$$f(x,y) = \sqrt{x^2+y^2}$$

Definition at line 120 of file mathoper.h.

3.6.2.6 const int OP_MAX = 2

The constant specifies the binary maximum operation f that is defined as follows:

$$f(x,y) = \max(x,y)$$

Definition at line 72 of file mathoper.h.

3.6.2.7 const int OP_MIN = 1

The constant specifies the binary minimum operation f that is defined as follows:

$$f(x,y) = \min(x,y)$$

Definition at line 64 of file mathoper.h.

3.6.2.8 const int OP_MUL = 6

The constant specifies the binary multiplication operation f that is defined as follows:

$$f(x,y) = x \cdot y$$

Definition at line 104 of file mathoper.h.

3.7 Smart Pointers

The module specifies classes which will take care about freeing memory of allocated array when the scope of such array ends.

Data Structures

- class scoped_c_array

  Specifies a class which will take care about freeing memory with `free()` function. More...
- class scoped_cpp_array

  Specifies a class which will take care about freeing memory with `delete[]` function. More...

Defines

- #define DEFINE_SCOPED_OBJECT(name, freeing_proc)

  Defines a template class which will take care about freeing memory of allocated array when the scope of such array ends.

3.7.1 Data Structure Documentation

3.7.1.1 class scoped_c_array

The class takes care about freeing memory with `free()` function.

3.7.1.2 class scoped_cpp_array

The class takes care about freeing memory with `delete`[] function.

3.7.2 Define Documentation

3.7.2.1 #define DEFINE_SCOPED_OBJECT(*name, freeing_proc*)

Value:

```
template<typename T> class name \
{ \
public: \
  explicit name(T *p = 0): p(p) {} \
  ~name() \
  { \
    freeing_proc(p); \
  } \
  operator T *() const \
  { \
    return p; \
  } \
private: \
  name(const name &); \
  void operator =(const name &); \
  T *p; \
};
```

There are at least two types of arrays: allocated by new[] call (see scoped_cpp_array class) and allocated by malloc-like call (see scoped_c_array class). The code of classes for both cases is the same but one line in destructor. So the macro is defined for that purpose.

Definition at line 61 of file smartptr.h.

3.8 Cross-platform Declarations

The module contains macros and functions which provides cross-platform compiling of the library.

Defines

- #define MUSTINLINE __attribute__((always_inline))

  Specifiers a cross-platform instruct the compiler to insert a copy of the function body into each place the function is called.
- #define STATIC_SPECIAL static

  On MSVC a specialization of template function does not inherit storage class. So one has to set storage class manually which is forbidden by C++ Standard and is not compileable on g++.
- #define alignedmalloc(size, alignment) _mm_malloc(size, alignment)

  Specifies cross-platform aligned allocation.
- #define alignedfree(ptr) _mm_free(ptr)

  Specifies cross-platform aligned deallocations.

3.9 C99 Standard Data Types

The module describes the `stdint.h` file with is a header file in the C standard library introduced in the C99 standard library section 7.18 to allow programmers to write more portable code by providing a set of typedefs that specify exact-width integer types, together with the defined minimum and maximum allowable values for each type, using macros. This header is particularly useful for embedded programming which often involves considerable manipulation of hardware specific I/O registers requiring integer data of fixed widths, specific locations and exact alignments.

Defines

- #define _W64

  Defines `_W64` macros to mark types changing their size, like `intptr_t` or `uintptr_t`.
- #define INT8_MIN ((int8_t)_I8_MIN)

  Defines a minimum value of a signed 8-bit integer.
- #define INT8_MAX _I8_MAX

  Defines a maximum value of a signed 8-bit integer.
- #define INT16_MIN ((int16_t)_I16_MIN)

  Defines a minimum value of a signed 16-bit integer.
- #define INT16_MAX _I16_MAX

  Defines a maximum value of a signed 16-bit integer.
- #define INT32_MIN ((int32_t)_I32_MIN)

  Defines a minimum value of a signed 32-bit integer.
- #define INT32_MAX _I32_MAX

  Defines a maximum value of a signed 32-bit integer.
- #define INT64_MIN ((int64_t)_I64_MIN)

  Defines a minimum value of a signed 64-bit integer.
- #define INT64_MAX _I64_MAX

  Defines a maximum value of a signed 64-bit integer.
- #define UINT8_MAX _UI8_MAX

  Defines a maximum value of an unsigned 8-bit integer.
- #define UINT16_MAX _UI16_MAX

  Defines a maximum value of an unsigned 16-bit integer.
- #define UINT32_MAX _UI32_MAX

  Defines a maximum value of an unsigned 32-bit integer.
- #define UINT64_MAX _UI64_MAX

  Defines a maximum value of an unsigned 64-bit integer.
- #define INT_LEAST8_MIN INT8_MIN

  Defines a minimum value of a signed integer with a width of at least 8 bits.
- #define INT_LEAST8_MAX INT8_MAX

  Defines a maximum value of a signed integer with a width of at least 8 bits.
- #define INT_LEAST16_MIN INT16_MIN

Defines a minimum value of a signed integer with a width of at least 16 bits.

- #define INT_LEAST16_MAX INT16_MAX

  Defines a maximum value of a signed integer with a width of at least 16 bits.
- #define INT_LEAST32_MIN INT32_MIN

  Defines a minimum value of a signed integer with a width of at least 32 bits.
- #define INT_LEAST32_MAX INT32_MAX

  Defines a maximum value of a signed integer with a width of at least 32 bits.
- #define INT_LEAST64_MIN INT64_MIN

  Defines a minimum value of a signed integer with a width of at least 64 bits.
- #define INT_LEAST64_MAX INT64_MAX

  Defines a maximum value of a signed integer with a width of at least 64 bits.
- #define UINT_LEAST8_MAX UINT8_MAX

  Defines a maximum value of an unsigned integer with a width of at least 8 bits.
- #define UINT_LEAST16_MAX UINT16_MAX

  Defines a maximum value of an unsigned integer with a width of at least 16 bits.
- #define UINT_LEAST32_MAX UINT32_MAX

  Defines a maximum value of an unsigned integer with a width of at least 32 bits.
- #define UINT_LEAST64_MAX UINT64_MAX

  Defines a maximum value of an unsigned integer with a width of at least 64 bits.
- #define INT_FAST8_MIN INT8_MIN

  Defines a minimum value of a fastest signed 8-bit integer.
- #define INT_FAST8_MAX INT8_MAX

  Defines a maximum value of a fastest signed 8-bit integer.
- #define INT_FAST16_MIN INT16_MIN

  Defines a minimum value of a fastest signed 16-bit integer.
- #define INT_FAST16_MAX INT16_MAX

  Defines a maximum value of a fastest signed 16-bit integer.
- #define INT_FAST32_MIN INT32_MIN

  Defines a minimum value of a fastest signed 32-bit integer.
- #define INT_FAST32_MAX INT32_MAX

  Defines a maximum value of a fastest signed 32-bit integer.
- #define INT_FAST64_MIN INT64_MIN

  Defines a minimum value of a fastest signed 64-bit integer.
- #define INT_FAST64_MAX INT64_MAX

  Defines a maximum value of a fastest signed 64-bit integer.
- #define UINT_FAST8_MAX UINT8_MAX

  Defines a maximum value of a fastest unsigned 8-bit integer.
- #define UINT_FAST16_MAX UINT16_MAX

  Defines a maximum value of a fastest unsigned 16-bit integer.
- #define UINT_FAST32_MAX UINT32_MAX

  Defines a maximum value of a fastest unsigned 32-bit integer.
- #define UINT_FAST64_MAX UINT64_MAX

  Defines a maximum value of a fastest unsigned 64-bit integer.

- #define INTPTR_MIN INT32_MIN

  Defines a minimum value of a signed integer which is guaranteed to hold the value of a pointer.
- #define INTPTR_MAX INT32_MAX

  Defines a maximum value of a signed integer which is guaranteed to hold the value of a pointer.
- #define UINTPTR_MAX UINT32_MAX

  Defines a maximum value of an unsigned integer which is guaranteed to hold the value of a pointer.
- #define INTMAX_MIN INT64_MIN

  Defines a minimum value of a signed integer which has the greatest limits.
- #define INTMAX_MAX INT64_MAX

  Defines a maximum value of a signed integer which has the greatest limits.
- #define UINTMAX_MAX UINT64_MAX

  Defines a maximum value of an unsigned integer which has the greatest limits.
- #define PTRDIFF_MIN _I32_MIN

  Defines a minimum value `ptrdiff_t` can hold.
- #define PTRDIFF_MAX _I32_MAX

  Defines a maximum value `ptrdiff_t` can hold.
- #define SIG_ATOMIC_MIN INT_MIN

  Defines a minimum value `sig_atomic_t` can hold.
- #define SIG_ATOMIC_MAX INT_MAX

  Defines a maximum value `sig_atomic_t` can hold.
- #define SIZE_MAX _UI32_MAX

  Defines a maximum value `size_t` can hold.
- #define WCHAR_MIN 0

  Defines a minimum value for type `wchar_t`.
- #define WCHAR_MAX _UI16_MAX

  Defines a maximum value for type `wchar_t`.
- #define WINT_MIN 0

  Defines a minimum value for type `wint_t`.
- #define WINT_MAX _UI16_MAX

  Defines a maximum value for type `wint_t`.
- #define INT8_C(val) val##i8

  Defines a macros which converts an integer literal to a signed integer with a width of at least 8 bits.
- #define INT16_C(val) val##i16

  Defines a macros which converts an integer literal to a signed integer with a width of at least 16 bits.
- #define INT32_C(val) val##i32

  Defines a macros which converts an integer literal to a signed integer with a width of at least 32 bits.
- #define INT64_C(val) val##i64

  Defines a macros which converts an integer literal to a signed integer with a width of at least 64 bits.

- #define UINT8_C(val) val##ui8

  Defines a macros which converts an integer literal to an unsigned integer with a width of at least 8 bits.
- #define UINT16_C(val) val##ui16

  Defines a macros which converts an integer literal to an unsigned integer with a width of at least 16 bits.
- #define UINT32_C(val) val##ui32

  Defines a macros which converts an integer literal to an unsigned integer with a width of at least 32 bits.
- #define UINT64_C(val) val##ui64

  Defines a macros which converts an integer literal to an unsigned integer with a width of at least 64 bits.
- #define INTMAX_C INT64_C

  Defines a macros which converts an integer literal to a signed integer which has the greatest limits.
- #define UINTMAX_C UINT64_C

  Defines a macros which converts an integer literal to an unsigned integer which has the greatest limits.

Typedefs

- typedef signed char int8_t

  Defines a signed integer type with a width of exactly 8 bits.
- typedef signed short int16_t

  Defines a signed integer type with a width of exactly 16 bits.
- typedef signed int int32_t

  Defines a signed integer type with a width of exactly 32 bits.
- typedef signed __int64 int64_t

  Defines a signed integer type with a width of exactly 64 bits.
- typedef unsigned char uint8_t

  Defines an unsigned integer type with a width of exactly 8 bits.
- typedef unsigned short uint16_t

  Defines an unsigned integer type with a width of exactly 16 bits.
- typedef unsigned int uint32_t

  Defines an unsigned integer type with a width of exactly 32 bits.
- typedef unsigned __int64 uint64_t

  Defines an unsigned integer type with a width of exactly 64 bits.
- typedef int8_t int_least8_t

  Defines a signed integer type with a width of at least 8 bits.
- typedef int16_t int_least16_t

  Defines a signed integer type with a width of at least 16 bits.
- typedef int32_t int_least32_t

  Defines a signed integer type with a width of at least 32 bits.
- typedef int64_t int_least64_t

Defines a signed integer type with a width of at least 64 bits.

- typedef uint8_t uint_least8_t

  Defines an unsigned integer type with a width of at least 8 bits.
- typedef uint16_t uint_least16_t

  Defines an unsigned integer type with a width of at least 16 bits.
- typedef uint32_t uint_least32_t

  Defines an unsigned integer type with a width of at least 32 bits.
- typedef uint64_t uint_least64_t

  Defines an unsigned integer type with a width of at least 64 bits.
- typedef int8_t int_fast8_t

  Defines a signed integer type being usually fastest with a width of at least 8 bits.
- typedef int16_t int_fast16_t

  Defines a signed integer type being usually fastest with a width of at least 16 bits.
- typedef int32_t int_fast32_t

  Defines a signed integer type being usually fastest with a width of at least 32 bits.
- typedef int64_t int_fast64_t

  Defines a signed integer type being usually fastest with a width of at least 64 bits.
- typedef uint8_t uint_fast8_t

  Defines an unsigned integer type being usually fastest with a width of at least 8 bits.
- typedef uint16_t uint_fast16_t

  Defines an unsigned integer type being usually fastest with a width of at least 16 bits.
- typedef uint32_t uint_fast32_t

  Defines an unsigned integer type being usually fastest with a width of at least 32 bits.
- typedef uint64_t uint_fast64_t

  Defines an unsigned integer type being usually fastest with a width of at least 64 bits.
- typedef _W64 signed int intptr_t

  Defines a signed integer type which is guaranteed to hold the value of a pointer.
- typedef _W64 unsigned int uintptr_t

  Defines an unsigned integer type which is guaranteed to hold the value of a pointer.
- typedef int64_t intmax_t

  Defines a signed integer type which has the greatest limits.
- typedef uint64_t uintmax_t

  Defines an unsigned integer type which has the greatest limits.

Data Structure Documentation

4 File Documentation

4.1 crossplat.h File Reference

Different cross-platform declarations.

Defines

- #define MUSTINLINE __attribute__((always_inline))

  Specifiers a cross-platform instruct the compiler to insert a copy of the function body into each place the function is called.
- #define STATIC_SPECIAL static

  On MSVC a specialization of template function does not inherit storage class. So one has to set storage class manually which is forbidden by C++ Standard and is not compileable on g++.
- #define alignedmalloc(size, alignment) _mm_malloc(size, alignment)

  Specifies cross-platform aligned allocation.
- #define alignedfree(ptr) _mm_free(ptr)

  Specifies cross-platform aligned deallocations.

4.1.1 Detailed Description

Definition in file crossplat.h.

4.2 crossplat.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, 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 #ifndef CROSSPLAT_H_INCLUDED
00039 #define CROSSPLAT_H_INCLUDED
00040
00075 #ifdef _MSC_VER
00076
00081 #endif
00082
00083 #ifndef NO_INLINE
00084 #  ifndef MUSTINLINE
00085 #    ifdef _MSC_VER
00086 #      define MUSTINLINE __forceinline
00087 #    else
00088 #      define MUSTINLINE __attribute__((always_inline))
00089 #    endif
00090 #  endif // MUSTINLINE
00091 #else
00092 # undef MUSTINLINE
00093 # define MUSTINLINE
00094 #endif // NO_INLINE
00095
00096 #ifndef STATIC_SPECIAL
00097 #  if defined(_MSC_VER) || (__GNUC__ < 4) || ((__GNUC__ == 4) &&
       (__GNUC_MINOR__ < 3))
00098 #    define STATIC_SPECIAL static
00099 #  else
00100 #    define STATIC_SPECIAL
00101 #  endif
00102 #endif // STATIC_SPECIAL
00103
00104 #if defined(__MINGW32__)
00105 #  include <malloc.h>
00106 #  define alignedmalloc(size, alignment) __mingw_aligned_malloc(size,
       alignment)
00107 #  define alignedfree(ptr)               __mingw_aligned_free(ptr)
00108 #elif defined(ALCHEMY)
00109 #  include <malloc.h>
00110 #  define alignedmalloc(size, alignment) malloc(size)
00111 #  define alignedfree(ptr)               free(ptr)
00112 #else
00113 #  include <emmintrin.h>
00114 #  define alignedmalloc(size, alignment) _mm_malloc(size, alignment)
00115 #  define alignedfree(ptr)               _mm_free(ptr)
00116 #endif
00117
00118 #ifdef _MSC_VER
00119 # define snprintf _snprintf
00120 #endif
00121
00122
00123 #endif // CROSSPLAT_H_INCLUDED
00124
```

4.3 mathoper.h File Reference

Definition of mathematical operations.

Enumerations

- enum BiOp

  Specifies binary operations.
- enum AsOp

  Specifies associative operations.
- enum CoOp

  Specifies commutative operations.
- enum AsCoOp

  Specifies associative-commutative operations.
- enum IdOp

  Specifies idempotent operations.

Variables

- const int OP_MIN = 1

  Specifies binary minimum operation.
- const int OP_MAX = 2

  Specifies binary maximum operation.
- const int OP_ADD = 3

  Specifies binary addition operation.
- const int OP_DIF = 4

  Specifies binary difference operation.
- const int OP_ADF = 5

  Specifies binary absolute difference operation.
- const int OP_MUL = 6

  Specifies binary multiplication operation.
- const int OP_AVE = 7

  Specifies binary average operation.
- const int OP_EUC = 8

  Specifies binary Euclidean norm operation.

4.3.1 Detailed Description

Definition in file mathoper.h.

4.4 mathoper.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
00038 #pragma once
00039
00040 #ifndef MATHOPER_H_INCLUDED
00041 #define MATHOPER_H_INCLUDED
00042
00064 const int OP_MIN = 1;
00065
00072 const int OP_MAX = 2;
00073
00080 const int OP_ADD = 3;
00081
00088 const int OP_DIF = 4;
00089
00096 const int OP_ADF = 5;
00097
00104 const int OP_MUL = 6;
00105
00112 const int OP_AVE = 7;
00113
00120 const int OP_EUC = 8;
00121
00131 typedef enum
00132 {
00133   BIOP_MIN = OP_MIN,
00134   BIOP_MAX = OP_MAX,
00135   BIOP_ADD = OP_ADD,
00136   BIOP_DIF = OP_DIF,
00137   BIOP_ADF = OP_ADF,
00138   BIOP_MUL = OP_MUL,
00139   BIOP_AVE = OP_AVE,
00140   BIOP_EUC = OP_EUC
00141 } BiOp;
00142
00152 typedef enum
00153 {
00154   ASOP_MIN = OP_MIN,
```

```
00155   ASOP_MAX = OP_MAX,
00156   ASOP_ADD = OP_ADD,
00157   ASOP_MUL = OP_MUL,
00158   ASOP_EUC = OP_EUC
00159 } AsOp;
00160
00170 typedef enum
00171 {
00172   COOP_MIN = OP_MIN,
00173   COOP_MAX = OP_MAX,
00174   COOP_ADD = OP_ADD,
00175   COOP_ADF = OP_ADF,
00176   COOP_MUL = OP_MUL,
00177   COOP_AVE = OP_AVE,
00178   COOP_EUC = OP_EUC
00179 } CoOp;
00180
00192 typedef enum
00193 {
00194   ASCOOP_MIN = OP_MIN,
00195   ASCOOP_MAX = OP_MAX,
00196   ASCOOP_ADD = OP_ADD,
00197   ASCOOP_MUL = OP_MUL,
00198   ASCOOP_EUC = OP_EUC
00199 } AsCoOp;
00200
00211 typedef enum
00212 {
00213   IDOP_MIN = OP_MIN,
00214   IDOP_MAX = OP_MAX
00215 } IdOp;
00216
00217 #endif  /* MATHOPER_H_INCLUDED */
```

4.5 minarr.h File Reference

Definition of a multi-dimensional dense multi-channel array.

Data Structures

- struct MinArr

  A multi-dimensional dense multi-channel array representation. *More...*

4.5.1 Detailed Description

Definition in file minarr.h.

4.6 minarr.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 MINARR_H_INCLUDED
00042 #define MINARR_H_INCLUDED
00043
00044 #include <minutils/mintyp.h>
00045
00070 typedef struct
00071 {
00072   int32_t  dim;
00073
00074   int32_t *pSizes;
00075
00076   int32_t *pStrides;
00077
00078   int32_t  channelDepth;
00079   MinFmt   format;
00080
00081   uint8_t *pStart;
00082 } MinArr;
00083
00084 #endif  /* MINARR_H_INCLUDED */
```

4.7 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.

4.7.1 Detailed Description

Definition in file minerr.h.

4.8 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 */
```

4.9 mingeo.h File Reference

Definition of basic geometry data structures.

Data Structures

- struct MinPoint

  2D point with integer coordinates (zero-based). *More...*
- struct MinSize

  Size of an image or a rectangle. *More...*
- struct MinRect

  Offset and size of a rectangle. *More...*

Functions

- static MinPoint minPoint (int32_t x, int32_t y)

  Inline constructors for `MinPoint` *data type.*
- static MinSize minSize (int32_t width, int32_t height)

  Inline constructors for `MinSize` *data type.*
- static MinRect minRect (int32_t x, int32_t y, int32_t width, int32_t height)

  Inline constructors for `MinRect` *data type.*

4.9.1 Detailed Description

Definition in file mingeo.h.

4.10 mingeo.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 #ifndef MINGEO_H_INCLUDED
00041 #define MINGEO_H_INCLUDED
00042
00043 #ifdef _MSC_VER
00044 #  include <minutils/stdint-vc.h>
00045 #else
00046 #  include <stdint.h>
00047 #endif // _MSC_VER
00048
00061 typedef struct
00062 {
00063   int32_t x;
00064   int32_t y;
00065 } MinPoint;
00066
00073 typedef struct
00074 {
00075   int32_t width;
00076   int32_t height;
00077 } MinSize;
00078
00085 typedef struct
00086 {
00087   int32_t x;
00088   int32_t y;
00089   int32_t width;
00090   int32_t height;
00091 } MinRect;
00092
00103 static inline MinPoint minPoint(int32_t x, int32_t y)
00104 {
00105   MinPoint point = {x, y};
00106   return point;
00107 }
00108
00119 static inline MinSize minSize(int32_t width, int32_t height)
00120 {
00121   MinSize size = {width, height};
00122   return size;
00123 }
00124
00137 static inline MinRect minRect(int32_t x, int32_t y, int32_t width, int32_t
      height)
00138 {
00139   MinRect rect = {x, y, width, height};
00140   return rect;
00141 }
00142
00143 #endif  // MINGEO_H_INCLUDED
```

4.11 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...

4.11.1 Detailed Description

Definition in file minimg.h.

4.12 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 */
```

4.13 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.

4.13.1 Detailed Description

Definition in file minopt.h.

4.14 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
```

4.15 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.

4.15.1 Detailed Description

Definition in file mintyp.h.

4.16 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 */
```

4.17 smartptr.h File Reference

Different cross-platform declarations.

Defines

- #define DEFINE_SCOPED_OBJECT(name, freeing_proc)

  Defines a template class which will take care about freeing memory of allocated array when the scope of such array ends.

4.17.1 Detailed Description

Definition in file smartptr.h.

4.18 smartptr.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, 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 SMARTPTR_H_INCLUDED
00041 #define SMARTPTR_H_INCLUDED
00042
00043 #include <cstdlib>
00044
00061 #define DEFINE_SCOPED_OBJECT(name, freeing_proc) \
00062 template<typename T> class name \
00063 { \
00064 public: \
00065   explicit name(T *p = 0): p(p) {} \
00066   ~name() \
00067   { \
00068     freeing_proc(p); \
00069   } \
00070   operator T *() const \
00071   { \
00072     return p; \
00073   } \
00074 private: \
00075   name(const name &); \
00076   void operator =(const name &); \
00077   T *p; \
00078 };
00079
00088 DEFINE_SCOPED_OBJECT(scoped_c_array, free)
00089
00090
00098 DEFINE_SCOPED_OBJECT(scoped_cpp_array, delete[])
00099
```

```
00100 #endif  /* SMARTPTR_H_INCLUDED */
```

4.19 minutils.cpp File Reference

Main source file.

4.19.1 Detailed Description

Definition in file minutils.cpp.

4.20 minutils.cpp

```
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
```

4.21 stdint-doc.h File Reference

Documentation for <stdint.h> members.

4.21.1 Detailed Description

Definition in file stdint-doc.h.

4.22 stdint-doc.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 STDINT_VC_DOC_H_INCLUDED
00042 #define STDINT_VC_DOC_H_INCLUDED
00043
00558 #endif  // STDINT_VC_DOC_H_INCLUDED
```

4.23 stdint-vc.h File Reference

C99 standard library header file for use with MS VC++.

Defines

- #define _W64

  Defines `_W64` macros to mark types changing their size, like `intptr_t` or `uintptr_t`.
- #define INT8_MIN ((int8_t)_I8_MIN)

  Defines a minimum value of a signed 8-bit integer.
- #define INT8_MAX _I8_MAX

  Defines a maximum value of a signed 8-bit integer.
- #define INT16_MIN ((int16_t)_I16_MIN)

  Defines a minimum value of a signed 16-bit integer.
- #define INT16_MAX _I16_MAX

Defines a maximum value of a signed 16-bit integer.

- #define INT32_MIN ((int32_t)_I32_MIN)

  Defines a minimum value of a signed 32-bit integer.
- #define INT32_MAX _I32_MAX

  Defines a maximum value of a signed 32-bit integer.
- #define INT64_MIN ((int64_t)_I64_MIN)

  Defines a minimum value of a signed 64-bit integer.
- #define INT64_MAX _I64_MAX

  Defines a maximum value of a signed 64-bit integer.
- #define UINT8_MAX _UI8_MAX

  Defines a maximum value of an unsigned 8-bit integer.
- #define UINT16_MAX _UI16_MAX

  Defines a maximum value of an unsigned 16-bit integer.
- #define UINT32_MAX _UI32_MAX

  Defines a maximum value of an unsigned 32-bit integer.
- #define UINT64_MAX _UI64_MAX

  Defines a maximum value of an unsigned 64-bit integer.
- #define INT_LEAST8_MIN INT8_MIN

  Defines a minimum value of a signed integer with a width of at least 8 bits.
- #define INT_LEAST8_MAX INT8_MAX

  Defines a maximum value of a signed integer with a width of at least 8 bits.
- #define INT_LEAST16_MIN INT16_MIN

  Defines a minimum value of a signed integer with a width of at least 16 bits.
- #define INT_LEAST16_MAX INT16_MAX

  Defines a maximum value of a signed integer with a width of at least 16 bits.
- #define INT_LEAST32_MIN INT32_MIN

  Defines a minimum value of a signed integer with a width of at least 32 bits.
- #define INT_LEAST32_MAX INT32_MAX

  Defines a maximum value of a signed integer with a width of at least 32 bits.
- #define INT_LEAST64_MIN INT64_MIN

  Defines a minimum value of a signed integer with a width of at least 64 bits.
- #define INT_LEAST64_MAX INT64_MAX

  Defines a maximum value of a signed integer with a width of at least 64 bits.
- #define UINT_LEAST8_MAX UINT8_MAX

  Defines a maximum value of an unsigned integer with a width of at least 8 bits.
- #define UINT_LEAST16_MAX UINT16_MAX

  Defines a maximum value of an unsigned integer with a width of at least 16 bits.
- #define UINT_LEAST32_MAX UINT32_MAX

  Defines a maximum value of an unsigned integer with a width of at least 32 bits.
- #define UINT_LEAST64_MAX UINT64_MAX

  Defines a maximum value of an unsigned integer with a width of at least 64 bits.
- #define INT_FAST8_MIN INT8_MIN

  Defines a minimum value of a fastest signed 8-bit integer.

- #define INT_FAST8_MAX INT8_MAX

  Defines a maximum value of a fastest signed 8-bit integer.
- #define INT_FAST16_MIN INT16_MIN

  Defines a minimum value of a fastest signed 16-bit integer.
- #define INT_FAST16_MAX INT16_MAX

  Defines a maximum value of a fastest signed 16-bit integer.
- #define INT_FAST32_MIN INT32_MIN

  Defines a minimum value of a fastest signed 32-bit integer.
- #define INT_FAST32_MAX INT32_MAX

  Defines a maximum value of a fastest signed 32-bit integer.
- #define INT_FAST64_MIN INT64_MIN

  Defines a minimum value of a fastest signed 64-bit integer.
- #define INT_FAST64_MAX INT64_MAX

  Defines a maximum value of a fastest signed 64-bit integer.
- #define UINT_FAST8_MAX UINT8_MAX

  Defines a maximum value of a fastest unsigned 8-bit integer.
- #define UINT_FAST16_MAX UINT16_MAX

  Defines a maximum value of a fastest unsigned 16-bit integer.
- #define UINT_FAST32_MAX UINT32_MAX

  Defines a maximum value of a fastest unsigned 32-bit integer.
- #define UINT_FAST64_MAX UINT64_MAX

  Defines a maximum value of a fastest unsigned 64-bit integer.
- #define INTPTR_MIN INT32_MIN

  Defines a minimum value of a signed integer which is guaranteed to hold the value of a pointer.
- #define INTPTR_MAX INT32_MAX

  Defines a maximum value of a signed integer which is guaranteed to hold the value of a pointer.
- #define UINTPTR_MAX UINT32_MAX

  Defines a maximum value of an unsigned integer which is guaranteed to hold the value of a pointer.
- #define INTMAX_MIN INT64_MIN

  Defines a minimum value of a signed integer which has the greatest limits.
- #define INTMAX_MAX INT64_MAX

  Defines a maximum value of a signed integer which has the greatest limits.
- #define UINTMAX_MAX UINT64_MAX

  Defines a maximum value of an unsigned integer which has the greatest limits.
- #define PTRDIFF_MIN _I32_MIN

  Defines a minimum value `ptrdiff_t` can hold.
- #define PTRDIFF_MAX _I32_MAX

  Defines a maximum value `ptrdiff_t` can hold.
- #define SIG_ATOMIC_MIN INT_MIN

  Defines a minimum value `sig_atomic_t` can hold.
- #define SIG_ATOMIC_MAX INT_MAX

Defines a maximum value `sig_atomic_t` can hold.

- #define SIZE_MAX _UI32_MAX

  Defines a maximum value `size_t` can hold.
- #define WCHAR_MIN 0

  Defines a minimum value for type `wchar_t`.
- #define WCHAR_MAX _UI16_MAX

  Defines a maximum value for type `wchar_t`.
- #define WINT_MIN 0

  Defines a minimum value for type `wint_t`.
- #define WINT_MAX _UI16_MAX

  Defines a maximum value for type `wint_t`.
- #define INT8_C(val) val##i8

  Defines a macros which converts an integer literal to a signed integer with a width of at least 8 bits.
- #define INT16_C(val) val##i16

  Defines a macros which converts an integer literal to a signed integer with a width of at least 16 bits.
- #define INT32_C(val) val##i32

  Defines a macros which converts an integer literal to a signed integer with a width of at least 32 bits.
- #define INT64_C(val) val##i64

  Defines a macros which converts an integer literal to a signed integer with a width of at least 64 bits.
- #define UINT8_C(val) val##ui8

  Defines a macros which converts an integer literal to an unsigned integer with a width of at least 8 bits.
- #define UINT16_C(val) val##ui16

  Defines a macros which converts an integer literal to an unsigned integer with a width of at least 16 bits.
- #define UINT32_C(val) val##ui32

  Defines a macros which converts an integer literal to an unsigned integer with a width of at least 32 bits.
- #define UINT64_C(val) val##ui64

  Defines a macros which converts an integer literal to an unsigned integer with a width of at least 64 bits.
- #define INTMAX_C INT64_C

  Defines a macros which converts an integer literal to a signed integer which has the greatest limits.
- #define UINTMAX_C UINT64_C

  Defines a macros which converts an integer literal to an unsigned integer which has the greatest limits.

Typedefs

- typedef signed char int8_t

  Defines a signed integer type with a width of exactly 8 bits.
- typedef signed short int16_t

  Defines a signed integer type with a width of exactly 16 bits.
- typedef signed int int32_t

  Defines a signed integer type with a width of exactly 32 bits.
- typedef unsigned char uint8_t

  Defines an unsigned integer type with a width of exactly 8 bits.
- typedef unsigned short uint16_t

  Defines an unsigned integer type with a width of exactly 16 bits.
- typedef unsigned int uint32_t

  Defines an unsigned integer type with a width of exactly 32 bits.
- typedef signed __int64 int64_t

  Defines a signed integer type with a width of exactly 64 bits.
- typedef unsigned __int64 uint64_t

  Defines an unsigned integer type with a width of exactly 64 bits.
- typedef int8_t int_least8_t

  Defines a signed integer type with a width of at least 8 bits.
- typedef int16_t int_least16_t

  Defines a signed integer type with a width of at least 16 bits.
- typedef int32_t int_least32_t

  Defines a signed integer type with a width of at least 32 bits.
- typedef int64_t int_least64_t

  Defines a signed integer type with a width of at least 64 bits.
- typedef uint8_t uint_least8_t

  Defines an unsigned integer type with a width of at least 8 bits.
- typedef uint16_t uint_least16_t

  Defines an unsigned integer type with a width of at least 16 bits.
- typedef uint32_t uint_least32_t

  Defines an unsigned integer type with a width of at least 32 bits.
- typedef uint64_t uint_least64_t

  Defines an unsigned integer type with a width of at least 64 bits.
- typedef int8_t int_fast8_t

  Defines a signed integer type being usually fastest with a width of at least 8 bits.
- typedef int16_t int_fast16_t

  Defines a signed integer type being usually fastest with a width of at least 16 bits.
- typedef int32_t int_fast32_t

  Defines a signed integer type being usually fastest with a width of at least 32 bits.
- typedef int64_t int_fast64_t

  Defines a signed integer type being usually fastest with a width of at least 64 bits.
- typedef uint8_t uint_fast8_t

Defines an unsigned integer type being usually fastest with a width of at least 8 bits.

- typedef uint16_t uint_fast16_t

  Defines an unsigned integer type being usually fastest with a width of at least 16 bits.
- typedef uint32_t uint_fast32_t

  Defines an unsigned integer type being usually fastest with a width of at least 32 bits.
- typedef uint64_t uint_fast64_t

  Defines an unsigned integer type being usually fastest with a width of at least 64 bits.
- typedef _W64 signed int intptr_t

  Defines a signed integer type which is guaranteed to hold the value of a pointer.
- typedef _W64 unsigned int uintptr_t

  Defines an unsigned integer type which is guaranteed to hold the value of a pointer.
- typedef int64_t intmax_t

  Defines a signed integer type which has the greatest limits.
- typedef uint64_t uintmax_t

  Defines an unsigned integer type which has the greatest limits.

4.23.1 Detailed Description

Author

Alexander Chemeris

Definition in file stdint-vc.h.

4.24 stdint-vc.h

```
00001 // ISO C9x  compliant stdint.h for Microsoft Visual Studio
00002 // Based on ISO/IEC 9899:TC2 Committee draft (May 6, 2005) WG14/N1124
00003 //
00004 //  Copyright (c) 2006-2008 Alexander Chemeris
00005 //
00006 // Redistribution and use in source and binary forms, with or without
00007 // modification, are permitted provided that the following conditions are met:
00008 //
00009 //   1. Redistributions of source code must retain the above copyright notice,
00010 //      this list of conditions and the following disclaimer.
00011 //
00012 //   2. Redistributions in binary form must reproduce the above copyright
00013 //      notice, this list of conditions and the following disclaimer in the
00014 //      documentation and/or other materials provided with the distribution.
00015 //
00016 //   3. The name of the author may be used to endorse or promote products
00017 //      derived from this software without specific prior written permission.
00018 //
00019 // THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED
00020 // WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
00021 // MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
00022 // EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024 // PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
00025 // OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
00026 // WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
00027 // OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
00028 // ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00031
```

```
00032 #ifndef _MSC_VER // [
00033 #error "Use this header only with Microsoft Visual C++ compilers!"
00034 #endif // _MSC_VER ]
00035
00036 #ifndef _MSC_STDINT_H_ // [
00037 #define _MSC_STDINT_H_
00038
00039 #if _MSC_VER > 1000
00040 #pragma once
00041 #endif
00042
00043 #include <limits.h>
00044
00045 // For Visual Studio 6 in C++ mode and for many Visual Studio versions when
00046 // compiling for ARM we should wrap <wchar.h> include with 'extern "C++" {}'
00047 // or compiler give many errors like this:
00048 //   error C2733: second C linkage of overloaded function 'wmemchr' not allowed
00049 #ifdef __cplusplus
00050 extern "C++" {
00051 #endif
00052 #  include <wchar.h>
00053 #ifdef __cplusplus
00054 }
00055 #endif
00056
00057 // Define _W64 macros to mark types changing their size, like intptr_t.
00058 #ifndef _W64
00059 #  if !defined(__midl) && (defined(_X86_) || defined(_M_IX86)) && _MSC_VER >=
      1300
00060 #     define _W64 __w64
00061 #  else
00062 #     define _W64
00063 #  endif
00064 #endif
00065
00066
00067 // 7.18.1 Integer types
00068
00069 // 7.18.1.1 Exact-width integer types
00070
00071 // Visual Studio 6 and Embedded Visual C++ 4 doesn't
00072 // realize that, e.g. char has the same size as __int8
00073 // so we give up on __intX for them.
00074 #if (_MSC_VER < 1300)
00075    typedef signed char       int8_t;
00076    typedef signed short      int16_t;
00077    typedef signed int        int32_t;
00078    typedef unsigned char     uint8_t;
00079    typedef unsigned short    uint16_t;
00080    typedef unsigned int      uint32_t;
00081 #else
00082    typedef signed __int8     int8_t;
00083    typedef signed __int16    int16_t;
00084    typedef signed __int32    int32_t;
00085    typedef unsigned __int8   uint8_t;
00086    typedef unsigned __int16  uint16_t;
00087    typedef unsigned __int32  uint32_t;
00088 #endif
00089 typedef signed __int64       int64_t;
00090 typedef unsigned __int64     uint64_t;
00091
00092
00093 // 7.18.1.2 Minimum-width integer types
00094 typedef int8_t    int_least8_t;
00095 typedef int16_t   int_least16_t;
00096 typedef int32_t   int_least32_t;
00097 typedef int64_t   int_least64_t;
00098 typedef uint8_t   uint_least8_t;
00099 typedef uint16_t  uint_least16_t;
00100 typedef uint32_t  uint_least32_t;
00101 typedef uint64_t  uint_least64_t;
00102
00103 // 7.18.1.3 Fastest minimum-width integer types
00104 typedef int8_t    int_fast8_t;
```

```
00105 typedef int16_t   int_fast16_t;
00106 typedef int32_t   int_fast32_t;
00107 typedef int64_t   int_fast64_t;
00108 typedef uint8_t   uint_fast8_t;
00109 typedef uint16_t  uint_fast16_t;
00110 typedef uint32_t  uint_fast32_t;
00111 typedef uint64_t  uint_fast64_t;
00112
00113 // 7.18.1.4 Integer types capable of holding object pointers
00114 #ifdef _WIN64 // [
00115    typedef signed __int64    intptr_t;
00116    typedef unsigned __int64  uintptr_t;
00117 #else // _WIN64 ][
00118    typedef _W64 signed int   intptr_t;
00119    typedef _W64 unsigned int uintptr_t;
00120 #endif // _WIN64 ]
00121
00122 // 7.18.1.5 Greatest-width integer types
00123 typedef int64_t   intmax_t;
00124 typedef uint64_t  uintmax_t;
00125
00126
00127 // 7.18.2 Limits of specified-width integer types
00128
00129 #if !defined(__cplusplus) || defined(__STDC_LIMIT_MACROS) // [   See footnote
       220 at page 257 and footnote 221 at page 259
00130
00131 // 7.18.2.1 Limits of exact-width integer types
00132 #define INT8_MIN     ((int8_t)_I8_MIN)
00133 #define INT8_MAX     _I8_MAX
00134 #define INT16_MIN    ((int16_t)_I16_MIN)
00135 #define INT16_MAX    _I16_MAX
00136 #define INT32_MIN    ((int32_t)_I32_MIN)
00137 #define INT32_MAX    _I32_MAX
00138 #define INT64_MIN    ((int64_t)_I64_MIN)
00139 #define INT64_MAX    _I64_MAX
00140 #define UINT8_MAX    _UI8_MAX
00141 #define UINT16_MAX   _UI16_MAX
00142 #define UINT32_MAX   _UI32_MAX
00143 #define UINT64_MAX   _UI64_MAX
00144
00145 // 7.18.2.2 Limits of minimum-width integer types
00146 #define INT_LEAST8_MIN    INT8_MIN
00147 #define INT_LEAST8_MAX    INT8_MAX
00148 #define INT_LEAST16_MIN   INT16_MIN
00149 #define INT_LEAST16_MAX   INT16_MAX
00150 #define INT_LEAST32_MIN   INT32_MIN
00151 #define INT_LEAST32_MAX   INT32_MAX
00152 #define INT_LEAST64_MIN   INT64_MIN
00153 #define INT_LEAST64_MAX   INT64_MAX
00154 #define UINT_LEAST8_MAX   UINT8_MAX
00155 #define UINT_LEAST16_MAX  UINT16_MAX
00156 #define UINT_LEAST32_MAX  UINT32_MAX
00157 #define UINT_LEAST64_MAX  UINT64_MAX
00158
00159 // 7.18.2.3 Limits of fastest minimum-width integer types
00160 #define INT_FAST8_MIN    INT8_MIN
00161 #define INT_FAST8_MAX    INT8_MAX
00162 #define INT_FAST16_MIN   INT16_MIN
00163 #define INT_FAST16_MAX   INT16_MAX
00164 #define INT_FAST32_MIN   INT32_MIN
00165 #define INT_FAST32_MAX   INT32_MAX
00166 #define INT_FAST64_MIN   INT64_MIN
00167 #define INT_FAST64_MAX   INT64_MAX
00168 #define UINT_FAST8_MAX   UINT8_MAX
00169 #define UINT_FAST16_MAX  UINT16_MAX
00170 #define UINT_FAST32_MAX  UINT32_MAX
00171 #define UINT_FAST64_MAX  UINT64_MAX
00172
00173 // 7.18.2.4 Limits of integer types capable of holding object pointers
00174 #ifdef _WIN64 // [
00175 #  define INTPTR_MIN   INT64_MIN
00176 #  define INTPTR_MAX   INT64_MAX
00177 #  define UINTPTR_MAX  UINT64_MAX
```

```
00178 #else // _WIN64 ][
00179 #  define INTPTR_MIN   INT32_MIN
00180 #  define INTPTR_MAX   INT32_MAX
00181 #  define UINTPTR_MAX  UINT32_MAX
00182 #endif // _WIN64 ]
00183
00184 // 7.18.2.5 Limits of greatest-width integer types
00185 #define INTMAX_MIN   INT64_MIN
00186 #define INTMAX_MAX   INT64_MAX
00187 #define UINTMAX_MAX  UINT64_MAX
00188
00189 // 7.18.3 Limits of other integer types
00190
00191 #ifdef _WIN64 // [
00192 #  define PTRDIFF_MIN  _I64_MIN
00193 #  define PTRDIFF_MAX  _I64_MAX
00194 #else  // _WIN64 ][
00195 #  define PTRDIFF_MIN  _I32_MIN
00196 #  define PTRDIFF_MAX  _I32_MAX
00197 #endif  // _WIN64 ]
00198
00199 #define SIG_ATOMIC_MIN  INT_MIN
00200 #define SIG_ATOMIC_MAX  INT_MAX
00201
00202 #ifndef SIZE_MAX // [
00203 #  ifdef _WIN64 // [
00204 #     define SIZE_MAX  _UI64_MAX
00205 #  else // _WIN64 ][
00206 #     define SIZE_MAX  _UI32_MAX
00207 #  endif // _WIN64 ]
00208 #endif // SIZE_MAX ]
00209
00210 // WCHAR_MIN and WCHAR_MAX are also defined in <wchar.h>
00211 #ifndef WCHAR_MIN // [
00212 #  define WCHAR_MIN  0
00213 #endif  // WCHAR_MIN ]
00214 #ifndef WCHAR_MAX // [
00215 #  define WCHAR_MAX  _UI16_MAX
00216 #endif  // WCHAR_MAX ]
00217
00218 #define WINT_MIN  0
00219 #define WINT_MAX  _UI16_MAX
00220
00221 #endif // __STDC_LIMIT_MACROS ]
00222
00223
00224 // 7.18.4 Limits of other integer types
00225
00226 #if !defined(__cplusplus) || defined(__STDC_CONSTANT_MACROS) // [   See
      footnote 224 at page 260
00227
00228 // 7.18.4.1 Macros for minimum-width integer constants
00229
00230 #define INT8_C(val)  val##i8
00231 #define INT16_C(val) val##i16
00232 #define INT32_C(val) val##i32
00233 #define INT64_C(val) val##i64
00234
00235 #define UINT8_C(val)  val##ui8
00236 #define UINT16_C(val) val##ui16
00237 #define UINT32_C(val) val##ui32
00238 #define UINT64_C(val) val##ui64
00239
00240 // 7.18.4.2 Macros for greatest-width integer constants
00241 #define INTMAX_C   INT64_C
00242 #define UINTMAX_C  UINT64_C
00243
00244 #endif // __STDC_CONSTANT_MACROS ]
00245
00246
00247 #endif // _MSC_STDINT_H_ ]
```