

# Remap Reference Manual

0

Generated by Doxygen 1.3.9.1

Wed Apr 19 14:36:57 2006



# Contents

<b>1</b>	<b>Remap Directory Hierarchy</b>	<b>1</b>
1.1	Remap Directories . . . . .	1
<b>2</b>	<b>Remap Hierarchical Index</b>	<b>3</b>
2.1	Remap Class Hierarchy . . . . .	3
<b>3</b>	<b>Remap Class Index</b>	<b>5</b>
3.1	Remap Class List . . . . .	5
<b>4</b>	<b>Remap File Index</b>	<b>7</b>
4.1	Remap File List . . . . .	7
<b>5</b>	<b>Remap Directory Documentation</b>	<b>9</b>
5.1	src/Pixel/ Directory Reference . . . . .	9
5.2	src/VFiles/seviri_latlon/ Directory Reference . . . . .	10
5.3	src/ Directory Reference . . . . .	11
5.4	src/VFiles/ Directory Reference . . . . .	12
<b>6</b>	<b>Remap Class Documentation</b>	<b>13</b>
6.1	grid_type_ Struct Reference . . . . .	13
6.2	Nearer_from< Pixel_type > Class Template Reference . . . . .	17
6.3	Pixel_base< T, V > Class Template Reference . . . . .	18
6.4	PRODUCT Struct Reference . . . . .	22
6.5	VFile Class Reference . . . . .	23
6.6	VHdf Class Reference . . . . .	26
6.7	VHdf_Seviri Class Reference . . . . .	29
6.8	VIIR Class Reference . . . . .	30
6.9	VModis Class Reference . . . . .	31
6.10	VXRIT_SEVIRI Class Reference . . . . .	33

<b>7</b>	<b>Remap File Documentation</b>	<b>37</b>
7.1	src/allocation.hpp File Reference . . . . .	37
7.2	src/common.h File Reference . . . . .	38
7.3	src/debug.h File Reference . . . . .	40
7.4	src/Pixel/debug.h File Reference . . . . .	41
7.5	src/VFiles/debug.h File Reference . . . . .	42
7.6	src/filetypes.h File Reference . . . . .	43
7.7	src/grid.h File Reference . . . . .	45
7.8	src/hdf_utils.h File Reference . . . . .	50
7.9	src/parse_argument.h File Reference . . . . .	51
7.10	src/Pixel/Pixel.h File Reference . . . . .	52
7.11	src/reproject.h File Reference . . . . .	53
7.12	src/tokenize.h File Reference . . . . .	54
7.13	src/VFiles/normalize_cal_factors.h File Reference . . . . .	55
7.14	src/VFiles/seviri_latlon/geostat.h File Reference . . . . .	56
7.15	src/VFiles/VFile.h File Reference . . . . .	58
7.16	src/VFiles/VFiles.h File Reference . . . . .	59
7.17	src/VFiles/VHdf.h File Reference . . . . .	60
7.18	src/VFiles/VHdf_Seviri.h File Reference . . . . .	61
7.19	src/VFiles/VIIR.h File Reference . . . . .	62
7.20	src/VFiles/VModis.h File Reference . . . . .	63
7.21	src/VFiles/VModis_interpol.h File Reference . . . . .	64
7.22	src/VFiles/VModis_latlon_resolve.h File Reference . . . . .	66
7.23	src/VFiles/VXRIT_SEVIRI.h File Reference . . . . .	67

# Chapter 1

## Remap Directory Hierarchy

### 1.1 Remap Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

src . . . . .	11
Pixel . . . . .	9
VFiles . . . . .	12
seviri_latlon . . . . .	10



## Chapter 2

# Remap Hierarchical Index

### 2.1 Remap Class Hierarchy

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

grid_type_ . . . . .	13
Nearer_from< Pixel_type > . . . . .	17
Pixel_base< T, V > . . . . .	18
PRODUCT . . . . .	22
VFile . . . . .	23
VHdf . . . . .	26
VHdf_Seviri . . . . .	29
VIIR . . . . .	30
VModis . . . . .	31
VXRIT_SEVIRI . . . . .	33





# Chapter 3

## Remap Class Index

### 3.1 Remap Class List

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

<b>grid_type_</b> . . . . .	13
<b>Nearer_from&lt; Pixel_type &gt;</b> . . . . .	17
<b>Pixel_base&lt; T, V &gt;</b> . . . . .	18
<b>PRODUCT</b> . . . . .	22
<b>VFile</b> . . . . .	23
<b>VHdf</b> . . . . .	26
<b>VHdf_Seviri</b> . . . . .	29
<b>VIIR</b> . . . . .	30
<b>VModis</b> . . . . .	31
<b>VXRIT_SEVIRI</b> . . . . .	33



# Chapter 4

## Remap File Index

### 4.1 Remap File List

Here is a list of all files with brief descriptions:

src/ <b>allocation.hpp</b> . . . . .	37
src/ <b>common.h</b> . . . . .	38
src/ <b>debug.h</b> . . . . .	40
src/ <b>filetypes.h</b> . . . . .	43
src/ <b>grid.h</b> . . . . .	45
src/ <b>hdf_utils.h</b> . . . . .	50
src/ <b>parse_argument.h</b> . . . . .	51
src/ <b>reproject.h</b> . . . . .	53
src/ <b>tokenize.h</b> . . . . .	54
src/Pixel/ <b>debug.h</b> . . . . .	41
src/Pixel/ <b>Pixel.h</b> . . . . .	52
src/VFiles/ <b>debug.h</b> . . . . .	42
src/VFiles/ <b>normalize_cal_factors.h</b> . . . . .	55
src/VFiles/ <b>VFile.h</b> . . . . .	58
src/VFiles/ <b>VFiles.h</b> . . . . .	59
src/VFiles/ <b>VHdf.h</b> . . . . .	60
src/VFiles/ <b>VHdf_Seviri.h</b> . . . . .	61
src/VFiles/ <b>VIIR.h</b> . . . . .	62
src/VFiles/ <b>VModis.h</b> . . . . .	63
src/VFiles/ <b>VModis_interpol.h</b> . . . . .	64
src/VFiles/ <b>VModis_latlon_resolve.h</b> . . . . .	66
src/VFiles/ <b>VXRIT_SEVIRI.h</b> . . . . .	67
src/VFiles/seviri_latlon/ <b>geostat.h</b> . . . . .	56



## Chapter 5

# Remap Directory Documentation

### 5.1 src/Pixel/ Directory Reference

#### Files

- file `debug.h`
- file `Pixel.h`

## 5.2 src/VFiles/seviri\_latlon/ Directory Reference

### Files

- file `geostat.h`

## 5.3 src/ Directory Reference

### Directories

- directory **Pixel**
- directory **VFiles**

### Files

- file **allocation.hpp**
- file **common.h**
- file **debug.h**
- file **filetypes.h**
- file **grid.h**
- file **hdf\_utils.h**
- file **parse\_argument.h**
- file **reproject.h**
- file **tokenize.h**

## 5.4 src/VFiles/ Directory Reference

### Directories

- directory `seviri_latlon`

### Files

- file `debug.h`
- file `normalize_cal_factors.h`
- file `VFile.h`
- file `VFiles.h`
- file `VHdf.h`
- file `VHdf_Seviri.h`
- file `VIIR.h`
- file `VModis.h`
- file `VModis_interpol.h`
- file `VModis_latlon_resolve.h`
- file `VXRIT_SEVIRI.h`



# Chapter 6

## Remap Class Documentation

### 6.1 grid\_type\_ Struct Reference

```
#include <grid.h>
```

#### Public Attributes

- **char file** [STRING\_MAXLEN+1]  
*field used as the file source or target (for the grid contents) by functions **load\_grid**(p. 47) and **save\_grid**(p. 48)*
- **char input\_dataset** [STRING\_MAXLEN+1]  
*input dataset of the grid, in the input file (relevant for source and target grids)*
- **char output\_dataset** [STRING\_MAXLEN+1]  
*output dataset of the grid, in the output file (relevant for source and target grids), should be equal to input\_dataset by default*
- **int ichannel**  
*only relevant for 3-dimensional input data (stacks of 2-dimensional planes), e.g. Modis. The first plane is numbered 1. Should be set to 0 when unused.*
- **int rank**  
*rank of the grid buffers (should always be 2 for the time being)*
- **int nrows**  
*number of rows of the grid buffers (common to all the buffers lat, lon, tim, data, distance\_from\_ref, time\_from\_ref)*
- **int ncols**  
*number of columns of the grid (common to all the buffers lat, lon tim, data, distance\_from\_ref, time\_from\_ref)*
- **coord\_type \* lat**  
*buffer for latitudes (nrows\*ncols elements)*

- **coord\_type \* lon**  
*buffer for longitudes (nrows\*ncols elements)*
- **time\_type \* tim**  
*buffer for times of acquisition (nrows\*ncols elements)*
- **data\_type \* data**  
*buffer for measures, currently in 16-bit unsigned integer counts (nrows\*ncols elements)*
- **distance\_type \* distance\_from\_ref**  
*buffer for distances between pixels in the target (reference) grid and their nearest neighbour in the source grid (nrows\*ncols elements)*
- **time\_type \* time\_from\_ref**  
*buffer for time differences between pixels in the target (reference) grid and their nearest neighbour in the source grid (nrows\*ncols elements)*
- **float64 slope**  
*calibration scale factor (slope) to apply to a measure count to convert it into a physical value:  
 $phys\_val = slope * (count - offset)$*
- **float64 offset**  
*calibration offset factor to apply to a measure count to convert it into a physical value:  $phys\_val = slope * (count - offset)$*
- **bool is\_target**
- **int \* src\_irows**
- **int \* src\_icols**

### 6.1.1 Detailed Description

Grids are the main data structures handled by the software. They should have been designed as a class instead of a simple struct with functions to handle it, but by lack of time the software had to be delivered as is. A rewritten code with a grid class instead of a struct should be much clearer and easier to maintain, but will need a substantial amount of time to reimplement (and of course redocument !), that is not available today.

Grids are abstracts for reprojection. Their main purpose is to handle 2-dimensional buffers of geolocated data (measures along with their latitudes, longitudes and times of acquisition). The reprojection algorithm remaps a grid of data (from one instrument product) into another one. For convenience, origin and target of the data (files and datasets) are also maintained in the structure (although this is not a very clever design, I have to confess, I hope I'll have a chance to change this if more time is given to this project)

### 6.1.2 Member Data Documentation

#### 6.1.2.1 data\_type\* grid\_type::data

buffer for measures, currently in 16-bit unsigned integer counts (nrows\*ncols elements)

**6.1.2.2 distance\_type\* grid\_type ::distance\_from\_ref**

buffer for distances between pixels in the target (reference) grid and their nearest neighbour in the source grid (nrows\*ncols elements)

**6.1.2.3 char grid\_type ::file[STRING\_MAXLEN+1]**

field used as the file source or target (for the grid contents) by functions **load\_grid**(p. 47) and **save\_grid**(p. 48)

**6.1.2.4 int grid\_type ::ichannel**

only relevant for 3-dimensional input data (stacks of 2-dimensional planes), e.g. Modis. The first plane is numbered 1. Should be set to 0 when unused.

**6.1.2.5 char grid\_type ::input\_dataset[STRING\_MAXLEN+1]**

input dataset of the grid, in the input file (relevant for source and target grids)

**6.1.2.6 bool grid\_type ::is\_target**

specifies if a grid is a source (is\_target == false) or a target (is\_target == true) currently not used (intended as a future optimization of the reprojection code, in order to reuse precomputed source rows and cols instead of recomputing them again and again)

**6.1.2.7 coord\_type\* grid\_type ::lat**

buffer for latitudes (nrows\*ncols elements)

**6.1.2.8 coord\_type\* grid\_type ::lon**

buffer for longitudes (nrows\*ncols elements)

**6.1.2.9 int grid\_type ::ncols**

number of columns of the grid (common to all the buffers lat, lon tim, data, distance\_from\_ref, time\_from\_ref)

**6.1.2.10 int grid\_type ::nrows**

number of rows of the grid buffers (common to all the buffers lat, lon, tim, data, distance\_from\_ref, time\_from\_ref)

**6.1.2.11 float64 grid\_type ::offset**

calibration offset factor to apply to a measure count to convert it into a physical value: phys\_val = slope\*(count - offset)

**6.1.2.12 char grid\_type\_::output\_dataset[STRING\_MAXLEN+1]**

output dataset of the grid, in the output file (relevant for source and target grids), should be equal to input\_dataset by default

**6.1.2.13 int grid\_type\_::rank**

rank of the grid buffers (should always be 2 for the time being)

**6.1.2.14 float64 grid\_type\_::slope**

calibration scale factor (slope) to apply to a measure count to convert it into a physical value:  $\text{phys\_val} = \text{slope} * (\text{count} - \text{offset})$

**6.1.2.15 int\* grid\_type\_::src\_icols**

buffer to store, for each pixel of the target grid, its nearest neighbour's column in the source grid (nrows\*ncols elements); valid only if **grid\_type::is\_target**(p. 15) == true; currently not used (intended as a future optimization of the reprojection code, in order to reuse precomputed source rows and cols instead of recomputing them again and again)

**6.1.2.16 int\* grid\_type\_::src\_irows**

buffer to store, for each pixel of the target grid, its nearest neighbour's row in the source grid (nrows\*ncols elements); valid only if **grid\_type::is\_target**(p. 15) == true; currently not used (intended as a future optimization of the reprojection code, in order to reuse precomputed source rows and cols instead of recomputing them again and again)

**6.1.2.17 time\_type\* grid\_type\_::tim**

buffer for times of acquisition (nrows\*ncols elements)

**6.1.2.18 time\_type\* grid\_type\_::time\_from\_ref**

buffer for time differences between pixels in the target (reference) grid and their nearest neighbour in the source grid (nrows\*ncols elements)

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

- `src/grid.h`

## 6.2 Nearer\_from< Pixel\_type > Class Template Reference

```
#include <Pixel.h>
```

### Public Member Functions

- **Nearer\_from** (const Pixel\_type &**this\_pixel**)
- **bool operator()** (const Pixel\_type &pixel1, const Pixel\_type &pixel2) const

### Private Attributes

- const Pixel\_type **this\_pixel**

```
template<typename Pixel_type> class Nearer_from< Pixel_type >
```

#### 6.2.1 Constructor & Destructor Documentation

**6.2.1.1** `template<typename Pixel_type> Nearer_from< Pixel_type >::Nearer_from (const Pixel_type & this_pixel) [inline]`

#### 6.2.2 Member Function Documentation

**6.2.2.1** `template<typename Pixel_type> bool Nearer_from< Pixel_type >::operator() (const Pixel_type & pixel1, const Pixel_type & pixel2) const [inline]`

#### 6.2.3 Member Data Documentation

**6.2.3.1** `template<typename Pixel_type> const Pixel_type Nearer_from< Pixel_type >::this_pixel [private]`

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

- `src/Pixel/Pixel.h`

## 6.3 Pixel\_base< T, V > Class Template Reference

```
#include <Pixel.h>
```

### Public Types

- typedef Pixel\_base< T, V > Pixel\_type
- typedef T coord\_type
- typedef T distance\_type
- typedef V value\_type
- enum unit\_type { RADIANS, DEGREES }

### Public Member Functions

- Pixel\_base ()
- Pixel\_base (const coord\_type lat, const coord\_type lon, const value\_type &val, const unit\_type unit=RADIANS)
- bool operator< (const Pixel\_type &other) const
- bool operator== (const Pixel\_type &other) const
- coord\_type lat () const
- coord\_type lon () const
- value\_type val () const
- distance\_type distance (const Pixel\_type &other) const
- void get\_neighbours (std::vector< Pixel\_type > &neighbours, std::multiset< Pixel\_type > &pixels, distance\_type resolution, bool sorted=false) const

### Static Public Member Functions

- coord\_type get\_resolution ()
- void set\_resolution (distance\_type new\_resolution)

### Static Public Attributes

- const coord\_type DEFAULT\_RESOLUTION = 1.
- const coord\_type R\_EARTH = 6371.005076
- const distance\_type DEG2RAD = M\_PI/180.
- const distance\_type RAD2DEG = 180.\*M\_1\_PI

### Private Attributes

- coord\_type lat\_
- coord\_type lon\_
- value\_type val\_
- int ilat
- int ilon

### Static Private Attributes

- distance\_type resolution = DEFAULT\_RESOLUTION

## Friends

- `std::ostream & operator<< (std::ostream &os, const Pixel_type &pixel)`

```
template<typename T, typename V> class Pixel_base< T, V >
```

### 6.3.1 Member Typedef Documentation

6.3.1.1 `template<typename T, typename V> typedef T Pixel_base< T, V >::coord_type`

6.3.1.2 `template<typename T, typename V> typedef T Pixel_base< T, V >::distance_type`

6.3.1.3 `template<typename T, typename V> typedef Pixel_base<T, V> Pixel_base< T, V >::Pixel_type`

6.3.1.4 `template<typename T, typename V> typedef V Pixel_base< T, V >::value_type`

### 6.3.2 Member Enumeration Documentation

6.3.2.1 `template<typename T, typename V> enum Pixel_base::unit_type`

Enumeration values:

***RADIANS***

***DEGREES***

### 6.3.3 Constructor & Destructor Documentation

**6.3.3.1** `template<typename T, typename V> Pixel_base< T, V >::Pixel_base ()`  
[inline]

**6.3.3.2** `template<typename T, typename V> Pixel_base< T, V >::Pixel_base`  
(const coord\_type *lat*, const coord\_type *lon*, const value\_type & *val*,  
const unit\_type *unit* = RADIANS) [inline]

### 6.3.4 Member Function Documentation

**6.3.4.1** `template<typename T, typename V> distance_type Pixel_base< T, V`  
>::distance (const Pixel\_type & *other*) const [inline]

**6.3.4.2** `template<typename T, typename V> void Pixel_base< T, V`  
>::get\_neighbours (std::vector< Pixel\_type > & *neighbours*, std::multiset<  
Pixel\_type > & *pixels*, distance\_type *resolution*, bool *sorted* = false)  
const [inline]

**6.3.4.3** `template<typename T, typename V> coord_type Pixel_base< T, V`  
>::get\_resolution () [inline, static]

**6.3.4.4** `template<typename T, typename V> coord_type Pixel_base< T, V >::lat`  
( ) const [inline]

**6.3.4.5** `template<typename T, typename V> coord_type Pixel_base< T, V >::lon`  
( ) const [inline]

**6.3.4.6** `template<typename T, typename V> bool Pixel_base< T, V >::operator<`  
(const Pixel\_type & *other*) const [inline]

**6.3.4.7** `template<typename T, typename V> bool Pixel_base< T, V`  
>::operator== (const Pixel\_type & *other*) const [inline]

**6.3.4.8** `template<typename T, typename V> void Pixel_base< T, V`  
>::set\_resolution (distance\_type *new\_resolution*) [inline, static]

**6.3.4.9** `template<typename T, typename V> value_type Pixel_base< T, V >::val`  
( ) const [inline]

### 6.3.5 Friends And Related Function Documentation

**6.3.5.1** `template<typename T, typename V> std::ostream& operator<<`  
(std::ostream & *os*, const Pixel\_type & *pixel*) [friend]

### 6.3.6 Member Data Documentation

**6.3.6.1** `template<typename T, typename V> const Pixel_base< T, V`  
>::coord\_type Pixel\_base< T, V >::DEFAULT\_RESOLUTION = 1.  
[static]

**6.3.6.2** `template<typename T, typename V> const Pixel_base< T, V`  
>::distance\_type Pixel\_base< T, V >::DEG2RAD = M\_PI/180. [static]

**6.3.6.3** `template<typename T, typename V> int Pixel_base< T, V >::ilat`  
[private]

**6.3.6.4** `template<typename T, typename V> int Pixel_base< T, V >::ilon`  
[private]

**6.3.6.5** `template<typename T, typename V> coord_type Pixel_base< T, V`



- src/Pixel/Pixel.h

## 6.4 PRODUCT Struct Reference

```
#include <VModis_interpol.h>
```

### Public Attributes

- struct {  
    int **Nl**  
    int **Np**  
} **sds**
- struct {  
    **PROJTYPE** **type**  
} **projparam**
- int **Nl\_geo**
- int **Np\_geo**
- float **rowoffset**
- float **rowstep**
- float **coloffset**
- float **colstep**

### 6.4.1 Member Data Documentation

6.4.1.1 float **PRODUCT::coloffset**

6.4.1.2 float **PRODUCT::colstep**

6.4.1.3 int **PRODUCT::Nl**

6.4.1.4 int **PRODUCT::Nl\_geo**

6.4.1.5 int **PRODUCT::Np**

6.4.1.6 int **PRODUCT::Np\_geo**

6.4.1.7 struct { ... } **PRODUCT::projparam**

6.4.1.8 float **PRODUCT::rowoffset**

6.4.1.9 float **PRODUCT::rowstep**

6.4.1.10 struct { ... } **PRODUCT::sds**

6.4.1.11 **PROJTYPE** **PRODUCT::type**

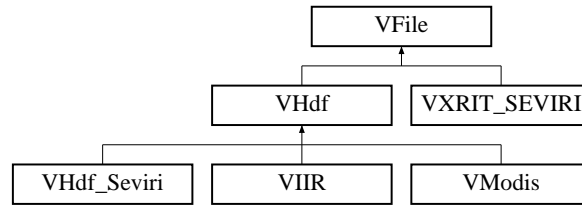
The documentation for this struct was generated from the following file:

- `src/VFiles/VModis_interpol.h`

## 6.5 VFile Class Reference

```
#include <VFile.h>
```

Inheritance diagram for VFile::



### Public Member Functions

- **VFile** (const char \*filename, const char \*dataset, int ichannel=0, const char \*sds\_time=NULL, const char \*sds\_latitude="Latitude", const char \*sds\_longitude="Longitude")
- virtual std::string **filename** () const
- virtual std::string **dataset** () const
- virtual ~**VFile** ()
- virtual void **get\_calibration** (double &slope, double &offset) const =0
- virtual **coord\_type** \* **lat** () const =0
- virtual **coord\_type** \* **lon** () const =0
- virtual **time\_type** \* **time** () const =0
- virtual **data\_type** \* **data** () const =0
- virtual std::vector< int > **dimensions** () const
- virtual int **rank** () const
- virtual int **dimension** (int idim) const

### Protected Attributes

- char **latlon\_filename\_** [STRING\_MAXLEN+1]
- char **time\_filename\_** [STRING\_MAXLEN+1]
- char **data\_filename\_** [STRING\_MAXLEN+1]
- char **sds\_lat\_** [STRING\_MAXLEN+1]
- char **sds\_lon\_** [STRING\_MAXLEN+1]
- char **sds\_time\_** [STRING\_MAXLEN+1]
- char **sds\_data\_** [STRING\_MAXLEN+1]
- int **ichannel\_**
- std::vector< int > **dimensions\_**

## 6.5.1 Constructor & Destructor Documentation

**6.5.1.1** `VFile::VFile (const char * filename, const char * dataset, int ichannel = 0, const char * sds_time = NULL, const char * sds_latitude = "Latitude", const char * sds_longitude = "Longitude")` [inline]

**6.5.1.2** `virtual VFile::~~VFile ()` [inline, virtual]

## 6.5.2 Member Function Documentation

**6.5.2.1** `virtual data_type* VFile::data () const` [pure virtual]

Implemented in `VHdf` (p.27), `VModis` (p.31), and `VXRIT_SEVIRI` (p.34).

**6.5.2.2** `virtual std::string VFile::dataset () const` [inline, virtual]

**6.5.2.3** `virtual int VFile::dimension (int idim) const` [inline, virtual]

**6.5.2.4** `virtual std::vector<int> VFile::dimensions () const` [inline, virtual]

**6.5.2.5** `virtual std::string VFile::filename () const` [inline, virtual]

**6.5.2.6** `virtual void VFile::get_calibration (double & slope, double & offset) const` [pure virtual]

Implemented in `VHdf` (p.27), `VHdf_Seviri` (p.29), `VIIR` (p.30), `VModis` (p.32), and `VXRIT_SEVIRI` (p.34).

**6.5.2.7** `virtual coord_type* VFile::lat () const` [pure virtual]

Implemented in `VHdf` (p.27), `VModis` (p.32), and `VXRIT_SEVIRI` (p.34).

**6.5.2.8** `virtual coord_type* VFile::lon () const` [pure virtual]

Implemented in `VHdf` (p.27), `VModis` (p.32), and `VXRIT_SEVIRI` (p.34).

**6.5.2.9** `virtual int VFile::rank () const` [inline, virtual]

**6.5.2.10** `virtual time_type* VFile::time () const` [pure virtual]

Implemented in `VHdf` (p.28), `VModis` (p.32), and `VXRIT_SEVIRI` (p.34).

### 6.5.3 Member Data Documentation

**6.5.3.1** `char VFile::data_filename_[STRING_MAXLEN+1]` [protected]

**6.5.3.2** `std::vector<int> VFile::dimensions_` [protected]

**6.5.3.3** `int VFile::ichannel_` [protected]

**6.5.3.4** `char VFile::latlon_filename_[STRING_MAXLEN+1]` [protected]

**6.5.3.5** `char VFile::sds_data_[STRING_MAXLEN+1]` [protected]

**6.5.3.6** `char VFile::sds_lat_[STRING_MAXLEN+1]` [protected]

**6.5.3.7** `char VFile::sds_lon_[STRING_MAXLEN+1]` [protected]

**6.5.3.8** `char VFile::sds_time_[STRING_MAXLEN+1]` [protected]

**6.5.3.9** `char VFile::time_filename_[STRING_MAXLEN+1]` [protected]

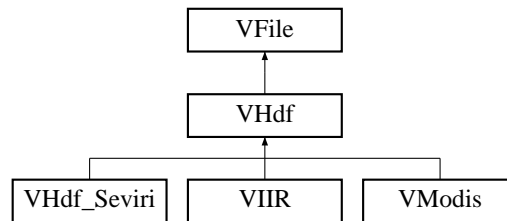
The documentation for this class was generated from the following file:

- `src/VFiles/VFile.h`

## 6.6 VHdf Class Reference

```
#include <VHdf.h>
```

Inheritance diagram for VHdf::



### Public Member Functions

- **VHdf** (const char \*filename, const char \*dataset, int ichannel=0, const char \*sds\_time=NULL, const char \*sds\_latitude="Latitude", const char \*sds\_longitude="Longitude")
- int **ichannel** ()
- virtual void **get\_calibration** (double &slope, double &offset) const
- **coord\_type** \* **lat** () const
- **coord\_type** \* **lon** () const
- **time\_type** \* **time** () const
- **data\_type** \* **data** () const
- virtual ~**VHdf** ()

### Protected Member Functions

- virtual void **read\_data\_2D** (const char \*dataset)
- virtual void **read\_data\_3D** (const char \*dataset, int ichannel)
- virtual void **read\_lat\_lon\_time** ()=0
- virtual void **destroy** ()

### Protected Attributes

- float32 \* **lat\_**
- float32 \* **lon\_**
- **time\_type** \* **time\_**
- uint16 \* **data\_**

## 6.6.1 Constructor & Destructor Documentation

**6.6.1.1** `VHdf::VHdf (const char * filename, const char * dataset, int ichannel = 0, const char * sds_time = NULL, const char * sds_latitude = "Latitude", const char * sds_longitude = "Longitude")`

**6.6.1.2** `virtual VHdf::~~VHdf ()` [virtual]

## 6.6.2 Member Function Documentation

**6.6.2.1** `data_type* VHdf::data () const` [inline, virtual]

Implements **VFile** (p. 24).

Reimplemented in **VModis** (p. 31).

**6.6.2.2** `virtual void VHdf::destroy ()` [protected, virtual]

Reimplemented in **VModis** (p. 32).

**6.6.2.3** `virtual void VHdf::get_calibration (double & slope, double & offset) const` [virtual]

Implements **VFile** (p. 24).

Reimplemented in **VHdf\_Seviri** (p. 29), **VIIR** (p. 30), and **VModis** (p. 32).

**6.6.2.4** `int VHdf::ichannel ()` [inline]

Reimplemented in **VModis** (p. 32).

**6.6.2.5** `coord_type* VHdf::lat () const` [inline, virtual]

Implements **VFile** (p. 24).

Reimplemented in **VModis** (p. 32).

**6.6.2.6** `coord_type* VHdf::lon () const` [inline, virtual]

Implements **VFile** (p. 24).

Reimplemented in **VModis** (p. 32).

**6.6.2.7** `virtual void VHdf::read_data_2D (const char * dataset)` [protected, virtual]

**6.6.2.8** `virtual void VHdf::read_data_3D (const char * dataset, int ichannel)` [protected, virtual]

**6.6.2.9** `virtual void VHdf::read_lat_lon_time ()` [protected, pure virtual]

Implemented in **VHdf\_Seviri** (p. 29), **VIIR** (p. 30), and **VModis** (p. 32).

**6.6.2.10** `time_type* VHdf::time () const` `[inline, virtual]`

Implements **VFile** (p. 24).

Reimplemented in **VModis** (p. 32).

### 6.6.3 Member Data Documentation

**6.6.3.1** `uint16* VHdf::data_` `[protected]`

**6.6.3.2** `float32* VHdf::lat_` `[protected]`

**6.6.3.3** `float32* VHdf::lon_` `[protected]`

**6.6.3.4** `time_type* VHdf::time_` `[protected]`

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

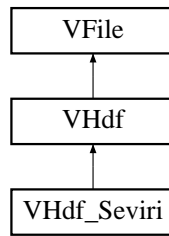
- `src/VFiles/VHdf.h`



## 6.7 VHdf\_Seviri Class Reference

```
#include <VHdf_Seviri.h>
```

Inheritance diagram for VHdf\_Seviri::



### Public Member Functions

- **VHdf\_Seviri** (const char \*filename, const char \*dataset)
- void **get\_calibration** (double &slope, double &offset) const

### Protected Member Functions

- void **read\_lat\_lon\_time** ()

#### 6.7.1 Constructor & Destructor Documentation

**6.7.1.1** VHdf\_Seviri::VHdf\_Seviri (const char \* *filename*, const char \* *dataset*)

#### 6.7.2 Member Function Documentation

**6.7.2.1** void VHdf\_Seviri::get\_calibration (double & *slope*, double & *offset*) const  
[virtual]

Reimplemented from **VHdf** (p. 27).

**6.7.2.2** void VHdf\_Seviri::read\_lat\_lon\_time () [protected, virtual]

Implements **VHdf** (p. 27).

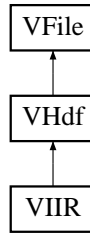
The documentation for this class was generated from the following file:

- src/VFiles/**VHdf\_Seviri.h**

## 6.8 VIIR Class Reference

```
#include <VIIR.h>
```

Inheritance diagram for VIIR::



### Public Member Functions

- **VIIR** (const char \*filename, const char \*dataset, int ichannel=0)
- void **get\_calibration** (double &slope, double &offset) const

### Protected Member Functions

- void **read\_lat\_lon\_time** ()

#### 6.8.1 Constructor & Destructor Documentation

**6.8.1.1** **VIIR::VIIR** (const char \* *filename*, const char \* *dataset*, int *ichannel* = 0)

#### 6.8.2 Member Function Documentation

**6.8.2.1** void **VIIR::get\_calibration** (double & *slope*, double & *offset*) const  
[virtual]

Reimplemented from **VHdf** (p. 27).

**6.8.2.2** void **VIIR::read\_lat\_lon\_time** () [protected, virtual]

Implements **VHdf** (p. 27).

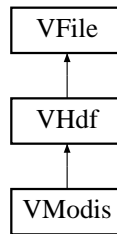
The documentation for this class was generated from the following file:

- src/VFiles/**VIIR.h**

## 6.9 VModis Class Reference

```
#include <VModis.h>
```

Inheritance diagram for VModis:



### Public Member Functions

- **VModis** (const char \*filename, const char \*dataset, int ichannel=0)
- int **ichannel** ()
- void **get\_calibration** (double &slope, double &offset) const
- **coord\_type** \* **lat** () const
- **coord\_type** \* **lon** () const
- **time\_type** \* **time** () const
- **data\_type** \* **data** () const
- ~VModis ()

### Protected Member Functions

- void **read\_lat\_lon\_time** ()
- void **destroy** ()

### Private Member Functions

- void **interpol** (int sample\_nrows, int sample\_ncols, const **coord\_type** \*sample\_lat, const **coord\_type** \*sample\_lon, int full\_nrows, int full\_ncols, **coord\_type** \*full\_lat, **coord\_type** \*full\_lon)

#### 6.9.1 Constructor & Destructor Documentation

**6.9.1.1** VModis::VModis (const char \* *filename*, const char \* *dataset*, int *ichannel* = 0)

**6.9.1.2** VModis::~VModis ()

#### 6.9.2 Member Function Documentation

**6.9.2.1** data\_type\* VModis::data () const [inline, virtual]

Reimplemented from **VHdf** (p. 27).

**6.9.2.2 void VModis::destroy () [protected, virtual]**

Reimplemented from **VHdf** (p. 27).

**6.9.2.3 void VModis::get\_calibration (double & *slope*, double & *offset*) const [virtual]**

Reimplemented from **VHdf** (p. 27).

**6.9.2.4 int VModis::ichannel () [inline]**

Reimplemented from **VHdf** (p. 27).

**6.9.2.5 void VModis::interpol (int *sample\_nrows*, int *sample\_ncols*, const coord\_type \* *sample\_lat*, const coord\_type \* *sample\_lon*, int *full\_nrows*, int *full\_ncols*, coord\_type \* *full\_lat*, coord\_type \* *full\_lon*) [private]****6.9.2.6 coord\_type\* VModis::lat () const [inline, virtual]**

Reimplemented from **VHdf** (p. 27).

**6.9.2.7 coord\_type\* VModis::lon () const [inline, virtual]**

Reimplemented from **VHdf** (p. 27).

**6.9.2.8 void VModis::read\_lat\_lon\_time () [protected, virtual]**

Implements **VHdf** (p. 27).

**6.9.2.9 time\_type\* VModis::time () const [inline, virtual]**

Reimplemented from **VHdf** (p. 28).

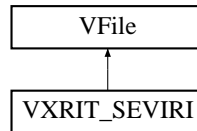
The documentation for this class was generated from the following file:

- `src/VFiles/VModis.h`

## 6.10 VXRIT\_SEVIRI Class Reference

```
#include <VXRIT_SEVIRI.h>
```

Inheritance diagram for VXRIT\_SEVIRI::



### Public Member Functions

- **VXRIT\_SEVIRI** (const char \*filename, const char \*dataset)
- void **get\_calibration** (double &slope, double &offset) const
- **coord\_type \* lat** () const
- **coord\_type \* lon** () const
- **time\_type \* time** () const
- **data\_type \* data** () const
- **~VXRIT\_SEVIRI** ()

### Private Member Functions

- void **read\_seviri\_latlon** (const char \*filename, size\_t nbuffer, float \*buffer)
- void **allocate** (size\_t n)
- void **destroy** ()

### Private Attributes

- float \* **lat\_**
- float \* **lon\_**
- **time\_type \* time\_**
- uint16\_t \* **data\_**
- MSG\_Prologue **prologue\_**

### Static Private Attributes

- const int **SEVIRI\_NROWS** = 3712
- const int **SEVIRI\_NCOLS** = 3712

### 6.10.1 Constructor & Destructor Documentation

**6.10.1.1** `VXRIT_SEVIRI::VXRIT_SEVIRI (const char * filename, const char * dataset)`

**6.10.1.2** `VXRIT_SEVIRI::~~VXRIT_SEVIRI ()`

### 6.10.2 Member Function Documentation

**6.10.2.1** `void VXRIT_SEVIRI::allocate (size_t n)` [private]

**6.10.2.2** `data_type* VXRIT_SEVIRI::data () const` [inline, virtual]

Implements **VFile** (p. 24).

**6.10.2.3** `void VXRIT_SEVIRI::destroy ()` [private]

**6.10.2.4** `void VXRIT_SEVIRI::get_calibration (double & slope, double & offset) const` [virtual]

Implements **VFile** (p. 24).

**6.10.2.5** `coord_type* VXRIT_SEVIRI::lat () const` [inline, virtual]

Implements **VFile** (p. 24).

**6.10.2.6** `coord_type* VXRIT_SEVIRI::lon () const` [inline, virtual]

Implements **VFile** (p. 24).

**6.10.2.7** `void VXRIT_SEVIRI::read_seviri_latlon (const char * filename, size_t nbuffer, float * buffer)` [private]

**6.10.2.8** `time_type* VXRIT_SEVIRI::time () const` [inline, virtual]

Implements **VFile** (p. 24).

### 6.10.3 Member Data Documentation

**6.10.3.1**   `uint16_t* VXRIT_SEVIRI::data_`   [private]

**6.10.3.2**   `float* VXRIT_SEVIRI::lat_`   [private]

**6.10.3.3**   `float* VXRIT_SEVIRI::lon_`   [private]

**6.10.3.4**   `MSG_Prologue VXRIT_SEVIRI::prologue_`   [private]

**6.10.3.5**   `const int VXRIT_SEVIRI::SEVIRI_NCOLS = 3712`   [static, private]

**6.10.3.6**   `const int VXRIT_SEVIRI::SEVIRI_NROWS = 3712`   [static, private]

**6.10.3.7**   `time_type* VXRIT_SEVIRI::time_`   [private]

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

- `src/VFiles/VXRIT_SEVIRI.h`





## Chapter 7

# Remap File Documentation

### 7.1 src/allocation.hpp File Reference

```
#include <new>
#include <cassert>
```

#### Defines

- `#define PDEBUG`

#### Functions

- `template<typename data_type> void allocate (data_type **&data, int nrows, int ncols)`
- `template<typename data_type> void deallocate (data_type **&data)`

#### 7.1.1 Define Documentation

##### 7.1.1.1 `#define PDEBUG`

#### 7.1.2 Function Documentation

##### 7.1.2.1 `template<typename data_type> void allocate (data_type **& data, int nrows, int ncols)`

##### 7.1.2.2 `template<typename data_type> void deallocate (data_type **& data)`

## 7.2 src/common.h File Reference

```
#include <iostream>
#include <cassert>
#include <cstring>
#include <cstdio>
#include <cstdlib>
#include <stdint.h>
#include <ctime>
#include "Hdf.hpp"
#include "debug.h"
```

### Defines

- `#define STRING_MAXLEN 255`
- `#define APPNAME "remap"`
- `#define OPT_VERBOSE 0x00000001`
- `#define DATA_TYPE_MIN 0`
- `#define DATA_TYPE_MAX 65535`
- `#define HDF_COORD_TYPE DFNT_FLOAT32`
- `#define HDF_DISTANCE_TYPE DFNT_FLOAT32`
- `#define HDF_TIME_TYPE DFNT_FLOAT64`
- `#define HDF_DATA_TYPE DFNT_UINT16`
- `#define DATA_TYPE uint16_t`
- `#define FLOAT_TYPE float`

### Typedefs

- `typedef FLOAT_TYPE float_type`
- `typedef float_type distance_type`
- `typedef float_type coord_type`
- `typedef float64 time_type`
- `typedef DATA_TYPE data_type`

### Variables

- `const float FLOAT_NAN = 0./0.`
- `const double DOUBLE_NAN = 0./0.`
- `const distance_type DEFAULT_DISTANCE_FILL_VALUE = FLOAT_NAN`
- `const coord_type DEFAULT_COORD_FILL_VALUE = FLOAT_NAN`
- `const time_type DEFAULT_TIME_FILL_VALUE = DOUBLE_NAN`
- `const data_type DEFAULT_DATA_FILL_VALUE = DATA_TYPE_MAX`
- `bool g_verbose`

### 7.2.1 Define Documentation

- 7.2.1.1 `#define APPNAME "remap"`
- 7.2.1.2 `#define DATA_TYPE uint16_t`
- 7.2.1.3 `#define DATA_TYPE_MAX 65535`
- 7.2.1.4 `#define DATA_TYPE_MIN 0`
- 7.2.1.5 `#define FLOAT_TYPE float`
- 7.2.1.6 `#define HDF_COORD_TYPE DFNT_FLOAT32`
- 7.2.1.7 `#define HDF_DATA_TYPE DFNT_UINT16`
- 7.2.1.8 `#define HDF_DISTANCE_TYPE DFNT_FLOAT32`
- 7.2.1.9 `#define HDF_TIME_TYPE DFNT_FLOAT64`
- 7.2.1.10 `#define OPT_VERBOSE 0x00000001`
- 7.2.1.11 `#define STRING_MAXLEN 255`

### 7.2.2 Typedef Documentation

- 7.2.2.1 `typedef float_type coord_type`
- 7.2.2.2 `typedef DATA_TYPE data_type`
- 7.2.2.3 `typedef float_type distance_type`
- 7.2.2.4 `typedef FLOAT_TYPE float_type`
- 7.2.2.5 `typedef float64 time_type`

### 7.2.3 Variable Documentation

- 7.2.3.1 `const coord_type DEFAULT_COORD_FILL_VALUE = FLOAT_NAN`
- 7.2.3.2 `const data_type DEFAULT_DATA_FILL_VALUE = DATA_TYPE_MAX`
- 7.2.3.3 `const distance_type DEFAULT_DISTANCE_FILL_VALUE = FLOAT_NAN`
- 7.2.3.4 `const time_type DEFAULT_TIME_FILL_VALUE = DOUBLE_NAN`
- 7.2.3.5 `const double DOUBLE_NAN = 0./0.`
- 7.2.3.6 `const float FLOAT_NAN = 0./0.`
- 7.2.3.7 `bool g_verbose`

## 7.3 src/debug.h File Reference

```
#include <iostream>
```

```
#include <cassert>
```

### Defines

- `#define Debug(code)`
- `#define PDEBUG`
- `#define Bench(code) do { code } while (0)`

#### 7.3.1 Define Documentation

**7.3.1.1** `#define Bench(code) do { code } while (0)`

**7.3.1.2** `#define Debug(code)`

**7.3.1.3** `#define PDEBUG`

## 7.4 src/Pixel/debug.h File Reference

```
#include <iostream>
#include <cassert>
#include <ctime>
```

### Defines

- `#define Debug(code)`
- `#define Bench(code) do { code } while (0)`

#### 7.4.1 Define Documentation

**7.4.1.1** `#define Bench(code) do { code } while (0)`

**7.4.1.2** `#define Debug(code)`

## 7.5 src/VFiles/debug.h File Reference

```
#include <iostream>
```

```
#include <cassert>
```

### Defines

- `#define Debug(code)`
- `#define PDEBUG`
- `#define Bench(code) do { code } while (0)`

#### 7.5.1 Define Documentation

**7.5.1.1** `#define Bench(code) do { code } while (0)`

**7.5.1.2** `#define Debug(code)`

**7.5.1.3** `#define PDEBUG`

## 7.6 src/filetypes.h File Reference

### Enumerations

- enum `filetype__type` {  
**FILETYPE\_MODIS\_AQUA\_1KM**, **FILETYPE\_MODIS\_TERRA\_1KM**,  
**FILETYPE\_CAL\_IIR\_L1**, **FILETYPE\_HDF\_SEVIRI**,  
**FILETYPE\_XRIT\_SEVIRI**, **FILETYPE\_UNKNOWN** }

### Functions

- char \* `filetype__to__cstring` (`filetype__type` filetype)
- `filetype__type` `get__filetype` (const char \*filename)
- void `print__supported__filetypes` ()

### 7.6.1 Enumeration Type Documentation

#### 7.6.1.1 enum filetype\_\_type

Enum type for filetypes (files identifiers). Once a file is resolved into a filetype (or file identifier), the latter will be used to instantiate a specific file object able to handle its contents

#### Enumeration values:

**FILETYPE\_MODIS\_AQUA\_1KM** enum code identifying MYD021KM files (Modis over Aqua products)

**FILETYPE\_MODIS\_TERRA\_1KM** enum code identifying MOD021KM files (Modis over Terra products)

**FILETYPE\_CAL\_IIR\_L1** enum code identifying CAL\_IIR\_L1 files (IIR over CALIPSO products)

**FILETYPE\_HDF\_SEVIRI** enum code identifying SEVIRI mosaic files (SEVIRI hdf products, from Icare)

**FILETYPE\_XRIT\_SEVIRI** enum code identifying SEVIRI original files (SEVIRI XRIT products)

**FILETYPE\_UNKNOWN** enum code for not known or not supported products (should always be the last field of the enum)

### 7.6.2 Function Documentation

#### 7.6.2.1 char\* filetype\_\_to\_\_cstring (filetype\_\_type *filetype*)

resolves a filetype into a static C-style (char \*) string (one of the enum codes from `filetype__type`).

cautious: as the returned value is a pointer to a static zone of memory, the function should never be called more than once in the same expression, nor in a multithread application (in these cases its behaviour is undefined)

#### Parameters:

*filetype* one of the filetypes owned by the `filetype__type` enumeration

**Returns:**

a pointer addressing a static array of chars containing a human readable description of the filetype

**7.6.2.2 filetype\_type get\_filetype (const char \* *filename*)**

resolves a filename into one of the filetypes from the filetype\_type enumeration

**Parameters:**

*filename* the name of one of the software-supported products

**Returns:**

one of the filetypes from the filetype\_type enumeration, or FILETYPE\_UNKNOWN is the argument is invalid or not supported

**7.6.2.3 void print\_supported\_filetypes ()**

an helper function that prints every supported types of files in the application

it simply calls filetype\_to\_cstr for each field of the filetype\_type enumeration (except FILETYPE\_UNKNOWN) and prints the result on the standard error

**See also:**

**filetype\_type**(p. 43)

**filetype\_to\_cstr(filetype\_type filetype)**(p. 43)

usage()



## 7.7 src/grid.h File Reference

```
#include <ctime>
#include "Hdf.hpp"
#include "common.h"
#include "VFiles.h"
```

### Classes

- struct `grid_type`

### Typedefs

- typedef `grid_type` `grid_type`

### Functions

- void `create_grid` (`grid_type` \*grid, int nrows=0, int ncols=0, float64 slope=1., float64 offset=0., `data_type` data\_fill\_value=DEFAULT\_DATA\_FILL\_VALUE, `coord_type` coord\_fill\_value=DEFAULT\_COORD\_FILL\_VALUE, `distance_type` distance\_fill\_value=DEFAULT\_DISTANCE\_FILL\_VALUE, `time_type` time\_fill\_value=DEFAULT\_TIME\_FILL\_VALUE)
- void `reset_grid` (`grid_type` \*grid, `data_type` data\_fill\_value, `coord_type` coord\_fill\_value, `distance_type` distance\_fill\_value, `time_type` time\_fill\_value)
- void `reset_grid` (`grid_type` \*grid, `data_type` data\_fill\_value, `distance_type` distance\_fill\_value, `time_type` time\_fill\_value)
- void `destroy_grid` (`grid_type` \*grid)
- int `load_grid` (`grid_type` \*grid, `data_type` data\_fill\_value=DEFAULT\_DATA\_FILL\_VALUE, `coord_type` coord\_fill\_value=DEFAULT\_COORD\_FILL\_VALUE, `distance_type` distance\_fill\_value=DEFAULT\_DISTANCE\_FILL\_VALUE, `time_type` time\_fill\_value=DEFAULT\_TIME\_FILL\_VALUE)
- int `save_grid` (`grid_type` \*grid, bool record\_delta, bool record\_latlontime)
- void `copy_grid` (`grid_type` \*target\_grid, `grid_type` \*src\_grid)
- void `copy_grid_footprint` (`grid_type` \*target\_grid, `grid_type` \*src\_grid)
- bool `have_same_grid_footprint` (`grid_type` \*grid1, `grid_type` \*grid2)

### 7.7.1 Typedef Documentation

#### 7.7.1.1 typedef struct `grid_type` `grid_type`

Grids are the main data structures handled by the software. They should have been designed as a class instead of a simple struct with functions to handle it, but by lack of time the software had to be delivered as is. A rewritten code with a grid class instead of a struct should be much clearer and easier to maintain, but will need a substantial amount of time to reimplement (and of course redocument !), that is not available today.

Grids are abstracts for reprojection. Their main purpose is to handle 2-dimensional buffers of geolocated data (measures along with their latitudes, longitudes and times of acquisition). The reprojection algorithm remaps a grid of data (from one instrument product) into another one. For

convenience, origin and target of the data (files and datasets) are also maintained in the structure (although this is not a very clever design, I have to confess, I hope I'll have a chance to change this if more time is given to this project)

## 7.7.2 Function Documentation

### 7.7.2.1 void copy\_grid (grid\_type \* target\_grid, grid\_type \* src\_grid)

intended to copy src\_grid into target\_grid (does correct allocation and copies) this function is not used in the current implementation

### 7.7.2.2 void copy\_grid\_footprint (grid\_type \* target\_grid, grid\_type \* src\_grid)

intended to copy the footprint of a grid (the footprint includes lightweight data, namely scalar fields, but not buffers) this function is not used in the current implementation

### 7.7.2.3 void create\_grid (grid\_type \* grid, int nrows = 0, int ncols = 0, float64 slope = 1., float64 offset = 0., data\_type data\_fill\_value = DEFAULT\_DATA\_FILL\_VALUE, coord\_type coord\_fill\_value = DEFAULT\_COORD\_FILL\_VALUE, distance\_type distance\_fill\_value = DEFAULT\_DISTANCE\_FILL\_VALUE, time\_type time\_fill\_value = DEFAULT\_TIME\_FILL\_VALUE)

the grid constructor

#### Parameters:

**grid** the address of a grid structure to initialize. All fields will be allocated (for field pointers) and initialized with the values of the subsequent parameters.

**nrows** the number of rows of the grid (common number of rows of all the buffers in the grid)

**ncols** the number of columns of the grid (common number of columns of all the buffers in the grid)

**slope** the initialization value for the slope field of the grid

**offset** the initialization value for the offset field of the grid

**data\_fill\_value** the initialization value for the **grid\_type::data**(p.14) buffer of the grid (after allocation of `nrows*ncols` elements)

**coord\_fill\_value** the initialization value the **grid\_type::lat**(p.15) and **grid\_type::lon**(p.15) buffers of the grid (after allocation of `nrows*ncols` elements)

**distance\_fill\_value** the initialization value for the **grid\_type::distance\_from\_ref**(p.15) buffer of the grid (after allocation of `nrows*ncols` elements)

**time\_fill\_value** the initialization value for the **grid\_type::tim**(p.16) and **grid\_type::time\_from\_ref**(p.16) buffers of the grid (after allocation of `nrows*ncols` elements)

#### See also:

**reset\_grid**(grid\_type \*, data\_type, coord\_type, distance\_type, time\_type)(p.48)

cautious: create\_grid currently does not initialize the **grid\_type::is\_target**(p.15) field (this one had to be added to meet some user's new requirements, only after the interface of **create\_grid**(p.46) was designed. So for the time being, the **grid\_type::is\_target**(p.15) field must be

set directly before `create_grid`(p. 46) is called, in order to allow `create_grid`(p. 46) to allocate and initialize correctly the `grid_type::src_irows`(p. 16) and `grid_type::src_icols`(p. 16) fields. This should be fixed in a future release (this will imply to break the interface of `create_grid`(p. 46), in order to pass the initialization value of `grid_type::is_target`(p. 15) as an argument).

#### 7.7.2.4 void destroy\_grid (grid\_type \* grid)

the destructor of a grid

it resets all the scalars fields and deallocates (and nullifies) all the pointer fields of a grid structure

##### Parameters:

*grid* a pointer to the grid to destroy

##### See also:

`create_grid`(p. 46)

`load_grid`(p. 47)

#### 7.7.2.5 bool have\_same\_grid\_footprint (grid\_type \* grid1, grid\_type \* grid2)

intended to compare two grids footprints (the footprint includes lightweight data, namely scalar fields, but not buffers) this function is not used in the current implementation

#### 7.7.2.6 int load\_grid (grid\_type \* grid, data\_type data\_fill\_value = DEFAULT\_DATA\_FILL\_VALUE, coord\_type coord\_fill\_value = DEFAULT\_COORD\_FILL\_VALUE, distance\_type distance\_fill\_value = DEFAULT\_DISTANCE\_FILL\_VALUE, time\_type time\_fill\_value = DEFAULT\_TIME\_FILL\_VALUE)

loads data from the dataset `grid_type::input_dataset`(p. 15) of the file `grid_type::file`(p. 15) into a grid. The function calls `create_grid`(p. 46), so don't do it yourself. Grids allocated and filled by `load_grid` must be destroyed by the caller with `destroy_grid`(p. 47)

##### Parameters:

*grid* a pointer to the `grid_type`(p. 45) structure to allocate and fill with the file data

*data\_fill\_value* the new initialization value of the `grid_type::data`(p. 14) buffer elements

*coord\_fill\_value* the new initialization value of the `grid_type::lat`(p. 15), `grid_type::lon`(p. 15) buffer elements

*distance\_fill\_value* the new initialization value of the `grid_type::distance_from_ref`(p. 15) buffer elements

*time\_fill\_value* the new initialization value of the `grid_type::tim`(p. 16) and `grid_type::time_from_ref`(p. 16) buffer elements

##### See also:

`create_grid`(p. 46) (called by this one)

`destroy_grid`(p. 47) (to call when the grid is not used anymore)

### 7.7.2.7 void reset\_grid (grid\_type \* *grid*, data\_type *data\_fill\_value*, distance\_type *distance\_fill\_value*, time\_type *time\_fill\_value*)

resets all the fields of a grid, except **grid\_type::lat**(p.15), **grid\_type::lon**(p.15), **grid\_type::tim**(p.16)

It is used for the target grid whose geolocation fields never change. On the other hand, **grid\_type::data**(p.14) (measures), **grid\_type::distance\_from\_ref**(p.15) and **grid\_type::time\_from\_ref**(p.16) buffers must be reset with each new source grid (as source will be remapped into target). This function is essentially called from the main loop in the `::main` function.

#### Parameters:

*grid* the pointer to the grid to reset

*data\_fill\_value* the new initialization value of the **grid\_type::data**(p.14) buffer elements

*distance\_fill\_value* the new initialization value of the **grid\_type::distance\_from\_ref**(p.15) buffer elements

*time\_fill\_value* the new initialization value of the **grid\_type::time\_from\_ref**(p.16) buffer elements

### 7.7.2.8 void reset\_grid (grid\_type \* *grid*, data\_type *data\_fill\_value*, coord\_type *coord\_fill\_value*, distance\_type *distance\_fill\_value*, time\_type *time\_fill\_value*)

resets all the fields of a grid. It is essentially called by **create\_grid**(p.46) after allocations of pointers have been done.

#### Parameters:

*grid* the pointer to the grid to reset

*data\_fill\_value* the new initialization value of the **grid\_type::data**(p.14) buffer elements

*coord\_fill\_value* the new initialization value of the **grid\_type::lat**(p.15), **grid\_type::lon**(p.15) buffer elements

*distance\_fill\_value* the new initialization value of the **grid\_type::distance\_from\_ref**(p.15) buffer elements

*time\_fill\_value* the new initialization value of the **grid\_type::tim**(p.16) and **grid\_type::time\_from\_ref**(p.16) buffer elements

#### See also:

**create\_grid**(p.46)

### 7.7.2.9 int save\_grid (grid\_type \* *grid*, bool *record\_delta*, bool *record\_latlon*)

saves **grid\_type::data**(p.14) buffer (with its slope and offset) into the **grid\_type::output\_dataset**(p.16) of the file **grid\_type::file**(p.15) recorded in the grid structure.

#### Parameters:

*grid* a pointer to the grid to save into a file

*record\_delta* if `true`, saves **grid\_type::distance\_from\_ref**(p.15) and **grid\_type::time\_from\_ref**(p.16) (namely 'delta' buffers) along with the **grid\_type::data**(p.14) buffer (this parameter may be controled by a user option, see `::main`)

*record\_latlon**time* if true, saves `grid_type::lat`(p.15), `grid_type::lon`(p.15) and `grid_type::tim`(p.16) buffers along with the `grid_type::data`(p.14) buffer (this is currently only used to save latitudes, longitudes and times of acquisition of the reference grid, see `::main`)

## 7.8 src/hdf\_utils.h File Reference

```
#include "Hdf.hpp"
```

### Functions

- `int hdf_create_empty_file (const char *file)`
- `int hdf_add_empty_sds (const char *file, const char *sds_name, int32 sds_type, int32 rank, int32 *dimensions, const float64 cal=1., const float64 offset=0., const float64 cal_err=0., const float64 off_err=0.)`

### 7.8.1 Function Documentation

**7.8.1.1** `int hdf_add_empty_sds (const char * file, const char * sds_name, int32 sds_type, int32 rank, int32 * dimensions, const float64 cal = 1., const float64 offset = 0., const float64 cal_err = 0., const float64 off_err = 0.)`

creates a new dataset in an existing hdf file (will fail if a dataset with the same name already exists)

#### Parameters:

*file* the hdf file to update

*sds\_name* the name of the new dataset (sds: scientific dataset) to add

*sds\_type* the hdf type code of the new dataset, e.g. DFNT\_UINT16, DFNT\_FLOAT32... (see HDF documentation for valid type codes)

*rank* the rank (number of dimensions) of the new dataset to create

*dimensions* a pointer to the array of dimensions of the new dataset

*cal* the calibration factor linked to the new dataset (also named slope); relation between calibrated and uncalibrated data is:  $cal\_data = cal * (uncal\_data - offset)$

*offset* the offset factor linked to the new dataset; relation between calibrated and uncalibrated data is:  $cal\_data = cal * (uncal\_data - offset)$

*cal\_err* error on the calibration factor (unused for the time being)

*off\_err* error on the offset (unused for the time being)

#### Returns:

0 on success, -1 on failure

**7.8.1.2** `int hdf_create_empty_file (const char * file)`

creates a new (empty) hdf file

#### Parameters:

*file* the name of the file to create

#### Returns:

0 on success, -1 on failure

## 7.9 src/parse\_argument.h File Reference

```
#include "grid.h"
```

### Functions

- `int parse_argument (int iarg, char *argv[], grid_type *grid)`

#### 7.9.1 Function Documentation

##### 7.9.1.1 `int parse_argument (int iarg, char * argv[], grid_type * grid)`

parses a user argument in order to extract some useful values for a `grid_type` structure. Expected argument must match the regular expression `^([^\@/]+)(\[([0-9]+)\])?(/[^\@]+)?@(\.)*$` which means in clear one of the four alternatives :

- `input_dataset@input_file`
- `input_dataset[ichannel]@input_file`
- `input_dataset/output_dataset@input_file`
- `input_dataset[ichannel]/output_dataset@input_file`

(with : `ichannel` an integer, `input_dataset` a char \*, `output_dataset` a char \*, and `input_file` a char \*)

#### Parameters:

***iarg*** the index of the argument to parse (between 1 and `argc - 1`)

***argv*** the list of the command arguments, from `::main`

***grid*** a pointer to a `grid_type` structure to update with infos found in the `i`-th argument

#### Returns:

0 on success, -1 on failure; the current implementation actually exits in a brutal way when a invalid argument is found, simply sending an error message on the standard error as a side effect. Indeed, an invalid argument was seen as a fatal error in the application context (no processing can be done), so it was judged useless to continue.

The fields of `grid_type` updated by `parse_argument` include:

- `grid_type::input_dataset`(p. 15)
- `grid_type::output_dataset`(p. 16) (set to `grid_type::input_dataset`(p. 15) if not found in the regexp)
- `grid_type::file`(p. 15)
- `grid_type::ichannel`(p. 15) (set to 0 if not found in the regexp)

## 7.10 src/Pixel/Pixel.h File Reference

```
#include <cassert>
#include <set>
#include <algorithm>
#include <utility>
#include <functional>
#include <cmath>
#include <vector>
#include <iostream>
#include "debug.h"
```

### Classes

- class `Nearer_from< Pixel_type >`
- class `Pixel_base< T, V >`

### Defines

- `#define BENCH`

### Functions

- `template<typename T> T sqr (T x)`

#### 7.10.1 Define Documentation

##### 7.10.1.1 `#define BENCH`

#### 7.10.2 Function Documentation

##### 7.10.2.1 `template<typename T> T sqr (T x) [inline]`



## 7.11 src/reproject.h File Reference

```
#include "common.h"
```

```
#include "grid.h"
```

### Functions

- void **reproject** (**grid\_type** \*src\_grid, **grid\_type** \*target\_grid, **distance\_type** distance, **time\_type** max\_dtime)

#### 7.11.1 Function Documentation

**7.11.1.1 void reproject** (**grid\_type** \* *src\_grid*, **grid\_type** \* *target\_grid*, **distance\_type** *distance*, **time\_type** *max\_dtime*)

reprojects src\_grid over target\_grid

The current implementation uses a *"nearest neighbour"* policy, an optional argument might be added to choose the reprojection policy (with this one by default)

Source and target grids must meet some requirements :

- their **grid\_type::lat**(p. 15), **grid\_type::lon**(p. 15) and **grid\_type::tim**(p. 16) fields must be filled
- **grid\_type::lat**(p. 15) must contain values in *degrees* between MIN\_LAT and MAX\_LAT
- **grid\_type::lon**(p. 15) must contain values in *degrees* between MIN\_LON and MAX\_LON
- **grid\_type::tim**(p. 16) must contain values in **time\_type**(p. 39) (TAI93, Temps Atomique International 1993), e.g. a number of seconds since the 1st Jan 1993, 0h00 UTC

#### Parameters:

*src\_grid* the grid whose content is to reproject

*target\_grid* the grid over which reprojection is to be done

*distance* the maximal search distance for the nearest neighbour

*max\_dtime* the maximal time difference for the nearest neighbour

**reproject**(p. 53) updates the following fields of target\_grid :

- **grid\_type::data**(p. 14)
- **grid\_type::distance\_from\_ref**(p. 15)
- **grid\_type::time\_from\_ref**(p. 16)
- **grid\_type::src\_irows**(p. 16) (currently not used)
- **grid\_type::src\_icols**(p. 16) (currently not used)

## 7.12 src/tokenize.h File Reference

### Typedefs

- typedef enum `etok_type_` `etok_type`

### Enumerations

- enum `etok_type_` {  
    `ETOK_SUCCESS`,   `ETOK_REGCOMP`,   `ETOK_MEMORY`,   `ETOK_-`  
    `NOMATCH`,  
    `ETOK_ESPACE` }

### Functions

- `etok_type` `tokenize` (const char \**pattern*, const char \**a\_string*, size\_t \**ntokens*, char \*\*\**ptokens*)
- void `free_tokens` (size\_t *ntokens*, char \*\**tokens*)

#### 7.12.1 Typedef Documentation

##### 7.12.1.1 typedef enum `etok_type_` `etok_type`

#### 7.12.2 Enumeration Type Documentation

##### 7.12.2.1 enum `etok_type_`

Enumeration values:

*`ETOK_SUCCESS`*  
*`ETOK_REGCOMP`*  
*`ETOK_MEMORY`*  
*`ETOK_NOMATCH`*  
*`ETOK_ESPACE`*

#### 7.12.3 Function Documentation

##### 7.12.3.1 void `free_tokens` (size\_t *ntokens*, char \*\* *tokens*)

##### 7.12.3.2 `etok_type` `tokenize` (const char \* *pattern*, const char \* *a\_string*, size\_t \* *ntokens*, char \*\*\* *ptokens*)

## 7.13 src/VFiles/normalize\_cal\_factors.h File Reference

### Functions

- void **normalize\_seviri\_factors** (int *met8\_channel*, double &*slope*, double &*offset*)
- void **normalize\_seviri\_factors** (const char \**dataset*, double &*slope*, double &*offset*)

### 7.13.1 Function Documentation

**7.13.1.1** void **normalize\_seviri\_factors** (const char \* *dataset*, double & *slope*, double & *offset*)

**7.13.1.2** void **normalize\_seviri\_factors** (int *met8\_channel*, double & *slope*, double & *offset*)

## 7.14 src/VFiles/seviri\_latlon/geostat.h File Reference

### Typedefs

- typedef enum **channel\_enum** channel\_t
- typedef enum **geostat\_err\_enum** geostat\_err\_t

### Enumerations

- enum **channel\_enum** { **GEOSTAT\_CHNL\_IR** = 1, **GEOSTAT\_CHNL\_VIS** = 2, **GEOSTAT\_CHNL\_WV** = 3 }
- enum **geostat\_err\_enum** { **GEOSTAT\_ERR\_OK** = 0, **GEOSTAT\_ERR\_OUT\_OF\_RANGE** = -1, **GEOSTAT\_ERR\_BAD\_INIT** = -2, **GEOSTAT\_ERR\_BAD\_INPUT\_COORD** = -3 }

### Functions

- **geostat\_err\_t** **geostat\_init** (const char \*satellite)
- **geostat\_err\_t** **geostat\_latlon\_to\_lincol\_double** (double latitude, double longitude, double \*line, double \*column)
- **geostat\_err\_t** **geostat\_lincol\_to\_latlon\_double** (double line, double column, double \*latitude, double \*longitude)
- **geostat\_err\_t** **geostat\_latlon\_to\_lincol** (const double latitude, const double longitude, unsigned long \*line, unsigned long \*column)
- **geostat\_err\_t** **geostat\_lincol\_to\_latlon** (const unsigned long line, const unsigned long column, double \*latitude, double \*longitude)
- **geostat\_err\_t** **geostat\_get\_azimut\_and\_zenithal** (const double latitude, const double longitude, double \*azimut, double \*zenithal)

#### 7.14.1 Typedef Documentation

##### 7.14.1.1 typedef enum channel\_enum channel\_t

##### 7.14.1.2 typedef enum geostat\_err\_enum geostat\_err\_t

#### 7.14.2 Enumeration Type Documentation

##### 7.14.2.1 enum channel\_enum

Enumeration values:

***GEOSTAT\_CHNL\_IR***  
***GEOSTAT\_CHNL\_VIS***  
***GEOSTAT\_CHNL\_WV***

##### 7.14.2.2 enum geostat\_err\_enum

Enumeration values:

***GEOSTAT\_ERR\_OK***

*GEOSTAT\_ERR\_OUT\_OF\_RANGE*  
*GEOSTAT\_ERR\_BAD\_INIT*  
*GEOSTAT\_ERR\_BAD\_INPUT\_COORD*

### 7.14.3 Function Documentation

- 7.14.3.1 `geostat_err_t geostat_get_azimut_and_zenithal (const double latitude, const double longitude, double * azimut, double * zenithal)`
- 7.14.3.2 `geostat_err_t geostat_init (const char * satellite)`
- 7.14.3.3 `geostat_err_t geostat_latlon_to_lincol (const double latitude, const double longitude, unsigned long * line, unsigned long * column)`
- 7.14.3.4 `geostat_err_t geostat_latlon_to_lincol_double (double latitude, double longitude, double * line, double * column)`
- 7.14.3.5 `geostat_err_t geostat_lincol_to_latlon (const unsigned long line, const unsigned long column, double * latitude, double * longitude)`
- 7.14.3.6 `geostat_err_t geostat_lincol_to_latlon_double (double line, double column, double * latitude, double * longitude)`

## 7.15 src/VFiles/VFile.h File Reference

```
#include <cassert>
#include <stdint.h>
#include <cstring>
#include <vector>
#include "common.h"
#include "debug.h"
```

### Classes

- class **VFile**

### Defines

- `#define` **DEBUG**

#### 7.15.1 Define Documentation

##### 7.15.1.1 `#define` **DEBUG**

## 7.16 src/VFiles/VFiles.h File Reference

```
#include "VFile.h"  
#include "VHdf.h"  
#include "VIIR.h"  
#include "VHdf_Seviri.h"  
#include "VXRIT_SEVIRI.h"  
#include "VModis.h"
```

## 7.17 src/VFiles/VHdf.h File Reference

```
#include <cstring>
#include "VFile.h"
#include "Hdf.hpp"
```

### Classes

- class **VHdf**



## 7.18 src/VFiles/VHdf\_Seviri.h File Reference

```
#include "VHdf.h"
```

### Classes

- class VHdf\_Seviri

## 7.19 src/VFiles/VIIR.h File Reference

```
#include "VHdf.h"
```

### Classes

- class **VIIR**

## 7.20 src/VFiles/VModis.h File Reference

```
#include <cstring>
#include "VHdf.h"
#include "Hdf.hpp"
```

### Classes

- class **VModis**

## 7.21 src/VFiles/VModis\_interpol.h File Reference

```
#include "common.h"
```

### Classes

- struct **PRODUCT**

### Enumerations

- enum **PROJTYPE** { **INTERP\_SWATH** }
- enum **INTERP\_OUTPUT** { **INTERP\_BOUNDS**, **INTERP\_CENTERS** }

### Functions

- void **get\_step\_and\_offset** (int n1, int n2, float \*offset, float \*step, int rowsperscan, int georowsamplingstep, unsigned char crosstrack)
- void **resample\_row\_geolocation** (**INTERP\_OUTPUT** output, double \*oldlon, double \*oldlat, **PRODUCT** \*product, **coord\_type** \*newlon, **coord\_type** \*newlat)
- void **get\_virtual\_row** (const **coord\_type** \*lon1, const **coord\_type** \*lat1, const **coord\_type** \*lon2, const **coord\_type** \*lat2, double t, int ncols, double \*lon3, double \*lat3)
- void **get\_georows** (double fractrow, **PRODUCT** \*input, int iscan, int \*georow1, int \*georow2, double \*t)

### 7.21.1 Enumeration Type Documentation

#### 7.21.1.1 enum **INTERP\_OUTPUT**

Enumeration values:

***INTERP\_BOUNDS***

***INTERP\_CENTERS***

#### 7.21.1.2 enum **PROJTYPE**

Enumeration values:

***INTERP\_SWATH***

## 7.21.2 Function Documentation

- 7.21.2.1** void get\_georows (double *fractrow*, PRODUCT \* *input*, int *iscan*, int \* *georow1*, int \* *georow2*, double \* *t*)
- 7.21.2.2** void get\_step\_and\_offset (int *n1*, int *n2*, float \* *offset*, float \* *step*, int *rowperscan*, int *georowsamplingstep*, unsigned char *crosstrack*)
- 7.21.2.3** void get\_virtual\_row (const coord\_type \* *lon1*, const coord\_type \* *lat1*, const coord\_type \* *lon2*, const coord\_type \* *lat2*, double *t*, int *ncols*, double \* *lon3*, double \* *lat3*)
- 7.21.2.4** void resample\_row\_geolocation (INTERP\_OUTPUT *output*, double \* *oldlon*, double \* *oldlat*, PRODUCT \* *product*, coord\_type \* *newlon*, coord\_type \* *newlat*)

## 7.22 src/VFiles/VModis\_latlon\_resolve.h File Reference

### Functions

- char \* VModis\_latlon\_resolve (char \*modis\_file, char \*modis\_latlon\_path=NULL)

### 7.22.1 Function Documentation

**7.22.1.1** char\* VModis\_latlon\_resolve (char \* *modis\_file*, char \* *modis\_latlon\_path* = NULL)

## 7.23 src/VFiles/VXRIT\_SEVIRI.h File Reference

```
#include <vector>
#include <stdint.h>
#include "VFile.h"
#include "libxrit.h"
```

### Classes

- class VXRIT\_SEVIRI

# Index

- ~VFile
  - VFile, 24
- ~VHdf
  - VHdf, 27
- ~VModis
  - VModis, 31
- ~VXRIT\_SEVIRI
  - VXRIT\_SEVIRI, 34
- allocate
  - allocation.hpp, 37
  - VXRIT\_SEVIRI, 34
- allocation.hpp
  - allocate, 37
  - deallocate, 37
  - PDEBUG, 37
- APPNAME
  - common.h, 39
- BENCH
  - Pixel.h, 52
- Bench
  - debug.h, 40
  - Pixel/debug.h, 41
  - VFiles/debug.h, 42
- channel\_enum
  - geostat.h, 56
- channel\_t
  - geostat.h, 56
- coloffset
  - PRODUCT, 22
- colstep
  - PRODUCT, 22
- common.h
  - APPNAME, 39
  - coord\_type, 39
  - DATA\_TYPE, 39
  - data\_type, 39
  - DATA\_TYPE\_MAX, 39
  - DATA\_TYPE\_MIN, 39
  - DEFAULT\_COORD\_FILL\_VALUE, 39
  - DEFAULT\_DATA\_FILL\_VALUE, 39
  - DEFAULT\_DISTANCE\_FILL\_VALUE, 39
  - DEFAULT\_TIME\_FILL\_VALUE, 39
  - distance\_type, 39
  - DOUBLE\_NAN, 39
  - FLOAT\_NAN, 39
  - FLOAT\_TYPE, 39
  - float\_type, 39
  - g\_verbose, 39
  - HDF\_COORD\_TYPE, 39
  - HDF\_DATA\_TYPE, 39
  - HDF\_DISTANCE\_TYPE, 39
  - HDF\_TIME\_TYPE, 39
  - OPT\_VERBOSE, 39
  - STRING\_MAXLEN, 39
  - time\_type, 39
- coord\_type
  - common.h, 39
  - Pixel\_base, 19
- copy\_grid
  - grid.h, 46
- copy\_grid\_footprint
  - grid.h, 46
- create\_grid
  - grid.h, 46
- data
  - grid\_type\_, 14
  - VFile, 24
  - VHdf, 27
  - VModis, 31
  - VXRIT\_SEVIRI, 34
- data\_
  - VHdf, 28
  - VXRIT\_SEVIRI, 35
- data\_filename\_
  - VFile, 25
- DATA\_TYPE
  - common.h, 39
- data\_type
  - common.h, 39
- DATA\_TYPE\_MAX
  - common.h, 39
- DATA\_TYPE\_MIN
  - common.h, 39
- dataset
  - VFile, 24



- deallocate
  - allocation.hpp, 37
- DEBUG
  - VFile.h, 58
- Debug
  - debug.h, 40
  - Pixel/debug.h, 41
  - VFiles/debug.h, 42
- debug.h
  - Bench, 40
  - Debug, 40
  - PDEBUG, 40
- DEFAULT\_COORD\_FILL\_VALUE
  - common.h, 39
- DEFAULT\_DATA\_FILL\_VALUE
  - common.h, 39
- DEFAULT\_DISTANCE\_FILL\_VALUE
  - common.h, 39
- DEFAULT\_RESOLUTION
  - Pixel\_base, 20
- DEFAULT\_TIME\_FILL\_VALUE
  - common.h, 39
- DEG2RAD
  - Pixel\_base, 20
- DEGREES
  - Pixel\_base, 19
- destroy
  - VHdf, 27
  - VModis, 31
  - VXRIT\_SEVIRI, 34
- destroy\_grid
  - grid.h, 47
- dimension
  - VFile, 24
- dimensions
  - VFile, 24
- dimensions\_
  - VFile, 25
- distance
  - Pixel\_base, 20
- distance\_from\_ref
  - grid\_type\_, 14
- distance\_type
  - common.h, 39
  - Pixel\_base, 19
- DOUBLE\_NAN
  - common.h, 39
- ETOK\_ESPACE
  - tokenize.h, 54
- ETOK\_MEMORY
  - tokenize.h, 54
- ETOK\_NOMATCH
  - tokenize.h, 54
- ETOK\_REGCOMP
  - tokenize.h, 54
- ETOK\_SUCCESS
  - tokenize.h, 54
- etok\_type
  - tokenize.h, 54
- etok\_type\_
  - tokenize.h, 54
- file
  - grid\_type\_, 15
- filename
  - VFile, 24
- FILETYPE\_CAL\_IIR\_L1
  - filetypes.h, 43
- FILETYPE\_HDF\_SEVIRI
  - filetypes.h, 43
- FILETYPE\_MODIS\_AQUA\_1KM
  - filetypes.h, 43
- FILETYPE\_MODIS\_TERRA\_1KM
  - filetypes.h, 43
- filetype\_to\_cstr
  - filetypes.h, 43
- filetype\_type
  - filetypes.h, 43
- FILETYPE\_UNKNOWN
  - filetypes.h, 43
- FILETYPE\_XRIT\_SEVIRI
  - filetypes.h, 43
- filetypes.h
  - FILETYPE\_CAL\_IIR\_L1, 43
  - FILETYPE\_HDF\_SEVIRI, 43
  - FILETYPE\_MODIS\_AQUA\_1KM, 43
  - FILETYPE\_MODIS\_TERRA\_1KM, 43
  - filetype\_to\_cstr, 43
  - filetype\_type, 43
  - FILETYPE\_UNKNOWN, 43
  - FILETYPE\_XRIT\_SEVIRI, 43
  - get\_filetype, 44
  - print\_supported\_filetypes, 44
- FLOAT\_NAN
  - common.h, 39
- FLOAT\_TYPE
  - common.h, 39
- float\_type
  - common.h, 39
- free\_tokens
  - tokenize.h, 54
- g\_verbose
  - common.h, 39
- geostat.h
  - channel\_enum, 56
  - channel\_t, 56

- GEOSTAT\_CHNL\_IR, 56
- GEOSTAT\_CHNL\_VIS, 56
- GEOSTAT\_CHNL\_WV, 56
- GEOSTAT\_ERR\_BAD\_INIT, 57
- GEOSTAT\_ERR\_BAD\_INPUT\_COORD, 57
- geostat\_err\_enum, 56
- GEOSTAT\_ERR\_OK, 56
- GEOSTAT\_ERR\_OUT\_OF\_RANGE, 56
- geostat\_err\_t, 56
- geostat\_get\_azimut\_and\_zenithal, 57
- geostat\_init, 57
- geostat\_latlon\_to\_lincol, 57
- geostat\_latlon\_to\_lincol\_double, 57
- geostat\_lincol\_to\_latlon, 57
- geostat\_lincol\_to\_latlon\_double, 57
- GEOSTAT\_CHNL\_IR
  - geostat.h, 56
- GEOSTAT\_CHNL\_VIS
  - geostat.h, 56
- GEOSTAT\_CHNL\_WV
  - geostat.h, 56
- GEOSTAT\_ERR\_BAD\_INIT
  - geostat.h, 57
- GEOSTAT\_ERR\_BAD\_INPUT\_COORD
  - geostat.h, 57
- geostat\_err\_enum
  - geostat.h, 56
- GEOSTAT\_ERR\_OK
  - geostat.h, 56
- GEOSTAT\_ERR\_OUT\_OF\_RANGE
  - geostat.h, 56
- geostat\_err\_t
  - geostat.h, 56
- geostat\_get\_azimut\_and\_zenithal
  - geostat.h, 57
- geostat\_init
  - geostat.h, 57
- geostat\_latlon\_to\_lincol
  - geostat.h, 57
- geostat\_latlon\_to\_lincol\_double
  - geostat.h, 57
- geostat\_lincol\_to\_latlon
  - geostat.h, 57
- geostat\_lincol\_to\_latlon\_double
  - geostat.h, 57
- get\_calibration
  - VFile, 24
  - VHdf, 27
  - VHdf\_Seviri, 29
  - VIIR, 30
  - VModis, 32
  - VXRIT\_SEVIRI, 34
- get\_filetype
  - filetypes.h, 44
- get\_georows
  - VModis\_interpol.h, 65
- get\_neighbours
  - Pixel\_base, 20
- get\_resolution
  - Pixel\_base, 20
- get\_step\_and\_offset
  - VModis\_interpol.h, 65
- get\_virtual\_row
  - VModis\_interpol.h, 65
- grid.h
  - copy\_grid, 46
  - copy\_grid\_footprint, 46
  - create\_grid, 46
  - destroy\_grid, 47
  - grid\_type, 45
  - have\_same\_grid\_footprint, 47
  - load\_grid, 47
  - reset\_grid, 47, 48
  - save\_grid, 48
- grid\_type
  - grid.h, 45
- grid\_type\_, 13
  - data, 14
  - distance\_from\_ref, 14
  - file, 15
  - ichannel, 15
  - input\_dataset, 15
  - is\_target, 15
  - lat, 15
  - lon, 15
  - ncols, 15
  - nrows, 15
  - offset, 15
  - output\_dataset, 15
  - rank, 16
  - slope, 16
  - src\_icols, 16
  - src\_irows, 16
  - tim, 16
  - time\_from\_ref, 16
- have\_same\_grid\_footprint
  - grid.h, 47
- hdf\_add\_empty\_sds
  - hdf\_utils.h, 50
- HDF\_COORD\_TYPE
  - common.h, 39
- hdf\_create\_empty\_file
  - hdf\_utils.h, 50
- HDF\_DATA\_TYPE
  - common.h, 39

- HDF\_DISTANCE\_TYPE
  - common.h, 39
- HDF\_TIME\_TYPE
  - common.h, 39
- hdf\_utils.h
  - hdf\_add\_empty\_sds, 50
  - hdf\_create\_empty\_file, 50
- ichannel
  - grid\_type\_, 15
  - VHdf, 27
  - VModis, 32
- ichannel\_
  - VFile, 25
- ilat
  - Pixel\_base, 20
- ilon
  - Pixel\_base, 20
- input\_dataset
  - grid\_type\_, 15
- INTERP\_BOUNDS
  - VModis\_interpol.h, 64
- INTERP\_CENTERS
  - VModis\_interpol.h, 64
- INTERP\_OUTPUT
  - VModis\_interpol.h, 64
- INTERP\_SWATH
  - VModis\_interpol.h, 64
- interpol
  - VModis, 32
- is\_target
  - grid\_type\_, 15
- lat
  - grid\_type\_, 15
  - Pixel\_base, 20
  - VFile, 24
  - VHdf, 27
  - VModis, 32
  - VXRIT\_SEVIRI, 34
- lat\_
  - Pixel\_base, 20
  - VHdf, 28
  - VXRIT\_SEVIRI, 35
- latlon\_filename\_
  - VFile, 25
- load\_grid
  - grid.h, 47
- lon
  - grid\_type\_, 15
  - Pixel\_base, 20
  - VFile, 24
  - VHdf, 27
  - VModis, 32
  - VXRIT\_SEVIRI, 34
- lon\_
  - Pixel\_base, 20
  - VHdf, 28
  - VXRIT\_SEVIRI, 35
- ncols
  - grid\_type\_, 15
- Nearer\_from, 17
  - Nearer\_from, 17
  - operator(), 17
  - this\_pixel, 17
- Nl
  - PRODUCT, 22
- Nl\_geo
  - PRODUCT, 22
- normalize\_cal\_factors.h
  - normalize\_seviri\_factors, 55
- normalize\_seviri\_factors
  - normalize\_cal\_factors.h, 55
- Np
  - PRODUCT, 22
- Np\_geo
  - PRODUCT, 22
- nrows
  - grid\_type\_, 15
- offset
  - grid\_type\_, 15
- operator()
  - Nearer\_from, 17
- operator<
  - Pixel\_base, 20
- operator<<
  - Pixel\_base, 20
- operator==
  - Pixel\_base, 20
- OPT\_VERBOSE
  - common.h, 39
- output\_dataset
  - grid\_type\_, 15
- parse\_argument
  - parse\_argument.h, 51
- parse\_argument.h
  - parse\_argument, 51
- PDEBUG
  - allocation.hpp, 37
  - debug.h, 40
  - VFiles/debug.h, 42
- Pixel.h
  - BENCH, 52
  - sqr, 52
- Pixel/debug.h

- Bench, 41
- Debug, 41
- Pixel\_base, 18
  - coord\_type, 19
  - DEFAULT\_RESOLUTION, 20
  - DEG2RAD, 20
  - DEGREES, 19
  - distance, 20
  - distance\_type, 19
  - get\_neighbours, 20
  - get\_resolution, 20
  - ilat, 20
  - ilon, 20
  - lat, 20
  - lat\_, 20
  - lon, 20
  - lon\_, 20
  - operator<, 20
  - operator<=, 20
  - operator==, 20
  - Pixel\_base, 20
  - Pixel\_type, 19
  - R\_EARTH, 20
  - RAD2DEG, 20
  - RADIANS, 19
  - resolution, 20
  - set\_resolution, 20
  - unit\_type, 19
  - val, 20
  - val\_, 20
  - value\_type, 19
- Pixel\_type
  - Pixel\_base, 19
- print\_supported\_filetypes
  - filetypes.h, 44
- PRODUCT, 22
  - coloffset, 22
  - colstep, 22
  - Nl, 22
  - Nl\_geo, 22
  - Np, 22
  - Np\_geo, 22
  - projparam, 22
  - rowoffset, 22
  - rowstep, 22
  - sds, 22
  - type, 22
- projparam
  - PRODUCT, 22
- PROJTYPE
  - VModis\_interpol.h, 64
- prologue\_
  - VXRIT\_SEVIRI, 35
- R\_EARTH
  - Pixel\_base, 20
- RAD2DEG
  - Pixel\_base, 20
- RADIANS
  - Pixel\_base, 19
- rank
  - grid\_type\_, 16
  - VFile, 24
- read\_data\_2D
  - VHdf, 27
- read\_data\_3D
  - VHdf, 27
- read\_lat\_lon\_time
  - VHdf, 27
  - VHdf\_Seviri, 29
  - VIIR, 30
  - VModis, 32
- read\_seviri\_latlon
  - VXRIT\_SEVIRI, 34
- reproject
  - reproject.h, 53
- reproject.h
  - reproject, 53
- resample\_row\_geolocation
  - VModis\_interpol.h, 65
- reset\_grid
  - grid.h, 47, 48
- resolution
  - Pixel\_base, 20
- rowoffset
  - PRODUCT, 22
- rowstep
  - PRODUCT, 22
- save\_grid
  - grid.h, 48
- sds
  - PRODUCT, 22
- sds\_data\_
  - VFile, 25
- sds\_lat\_
  - VFile, 25
- sds\_lon\_
  - VFile, 25
- sds\_time\_
  - VFile, 25
- set\_resolution
  - Pixel\_base, 20
- SEVIRI\_NCOLS
  - VXRIT\_SEVIRI, 35
- SEVIRI\_NROWS
  - VXRIT\_SEVIRI, 35
- slope

- grid\_type\_, 16
- sqr
  - Pixel.h, 52
- src/ Directory Reference, 11
- src/allocation.hpp, 37
- src/common.h, 38
- src/debug.h, 40
- src/filetypes.h, 43
- src/grid.h, 45
- src/hdf\_utils.h, 50
- src/parse\_argument.h, 51
- src/Pixel/ Directory Reference, 9
- src/Pixel/debug.h, 41
- src/Pixel/Pixel.h, 52
- src/reproject.h, 53
- src/tokenize.h, 54
- src/VFiles/ Directory Reference, 12
- src/VFiles/debug.h, 42
- src/VFiles/normalize\_cal\_factors.h, 55
- src/VFiles/seviri\_latlon/ Directory Reference, 10
- src/VFiles/seviri\_latlon/geostat.h, 56
- src/VFiles/VFile.h, 58
- src/VFiles/VFiles.h, 59
- src/VFiles/VHdf.h, 60
- src/VFiles/VHdf\_Seviri.h, 61
- src/VFiles/VIIR.h, 62
- src/VFiles/VModis.h, 63
- src/VFiles/VModis\_interpol.h, 64
- src/VFiles/VModis\_latlon\_resolve.h, 66
- src/VFiles/VXRIT\_SEVIRI.h, 67
- src\_icols
  - grid\_type\_, 16
- src\_irows
  - grid\_type\_, 16
- STRING\_MAXLEN
  - common.h, 39
- this\_pixel
  - Nearer\_from, 17
- tim
  - grid\_type\_, 16
- time
  - VFile, 24
  - VHdf, 27
  - VModis, 32
  - VXRIT\_SEVIRI, 34
- time\_
  - VHdf, 28
  - VXRIT\_SEVIRI, 35
- time\_filename\_
  - VFile, 25
- time\_from\_ref
  - grid\_type\_, 16
- time\_type
  - common.h, 39
- tokenize
  - tokenize.h, 54
- tokenize.h
  - ETOK\_ESPACE, 54
  - ETOK\_MEMORY, 54
  - ETOK\_NOMATCH, 54
  - ETOK\_REGCOMP, 54
  - ETOK\_SUCCESS, 54
  - etok\_type, 54
  - etok\_type\_, 54
  - free\_tokens, 54
  - tokenize, 54
- type
  - PRODUCT, 22
- unit\_type
  - Pixel\_base, 19
- val
  - Pixel\_base, 20
- val\_
  - Pixel\_base, 20
- value\_type
  - Pixel\_base, 19
- VFile, 23
  - ~VFile, 24
  - data, 24
  - data\_filename\_, 25
  - dataset, 24
  - dimension, 24
  - dimensions, 24
  - dimensions\_, 25
  - filename, 24
  - get\_calibration, 24
  - ichannel\_, 25
  - lat, 24
  - latlon\_filename\_, 25
  - lon, 24
  - rank, 24
  - sds\_data\_, 25
  - sds\_lat\_, 25
  - sds\_lon\_, 25
  - sds\_time\_, 25
  - time, 24
  - time\_filename\_, 25
  - VFile, 24
- VFile.h
  - DEBUG, 58
- VFiles/debug.h
  - Bench, 42
  - Debug, 42
  - PDEBUG, 42

---

VHdf, 26  
   ~VHdf, 27  
   data, 27  
   data\_, 28  
   destroy, 27  
   get\_calibration, 27  
   ichannel, 27  
   lat, 27  
   lat\_, 28  
   lon, 27  
   lon\_, 28  
   read\_data\_2D, 27  
   read\_data\_3D, 27  
   read\_lat\_lon\_time, 27  
   time, 27  
   time\_, 28  
   VHdf, 27  
 VHdf\_Seviri, 29  
   get\_calibration, 29  
   read\_lat\_lon\_time, 29  
   VHdf\_Seviri, 29  
 VIIR, 30  
   get\_calibration, 30  
   read\_lat\_lon\_time, 30  
   VIIR, 30  
 VModis, 31  
   ~VModis, 31  
   data, 31  
   destroy, 31  
   get\_calibration, 32  
   ichannel, 32  
   interpol, 32  
   lat, 32  
   lon, 32  
   read\_lat\_lon\_time, 32  
   time, 32  
   VModis, 31  
 VModis\_interpol.h  
   get\_georows, 65  
   get\_step\_and\_offset, 65  
   get\_virtual\_row, 65  
   INTERP\_BOUNDS, 64  
   INTERP\_CENTERS, 64  
   INTERP\_OUTPUT, 64  
   INTERP\_SWATH, 64  
   PROJTYPE, 64  
   resample\_row\_geolocation, 65  
 VModis\_latlon\_resolve  
   VModis\_latlon\_resolve.h, 66  
 VModis\_latlon\_resolve.h  
   VModis\_latlon\_resolve, 66  
 VXRIT\_SEVIRI, 33  
   ~VXRIT\_SEVIRI, 34  
   allocate, 34  
   data, 34  
   data\_, 35  
   destroy, 34  
   get\_calibration, 34  
   lat, 34  
   lat\_, 35  
   lon, 34  
   lon\_, 35  
   prologue\_, 35  
   read\_seviri\_latlon, 34  
   SEVIRI\_NCOLS, 35  
   SEVIRI\_NROWS, 35  
   time, 34  
   time\_, 35  
   VXRIT\_SEVIRI, 34

---